Tesis doctoral

Universidad Nacional de Educación a Distancia

Escuela Técnica Superior de Ingeniería Informática

Departamento de Lenguajes y Sistemas Informáticos

Pelican. Una plataforma para el diseño y desarrollo de escenarios de aprendizaje colaborativo. Soporte a los aspectos dinámicos

Javier Vélez Reyes
Licenciado en Informática por la UPM

2009

Tesis doctoral

Universidad Nacional de Educación a Distancia

Escuela Técnica Superior de Ingeniería Informática

Departamento de Lenguajes y Sistemas Informáticos

Pelican. Una plataforma para el diseño y desarrollo de escenarios de aprendizaje colaborativo. Soporte a los aspectos dinámicos

Javier Vélez Reyes
Licenciado en Informática por la UPM

2009

Tesis doctoral

Universidad Nacional de Educación a Distancia

Escuela Técnica Superior de Ingeniería Informática

Departamento de Lenguajes y Sistemas Informáticos

Pelican. Una plataforma para el diseño y desarrollo de escenarios de aprendizaje colaborativo. Soporte a los aspectos dinámicos

Javier Vélez Reyes
Licenciado en informática por la UPM

Dirigido por Mª Felisa Verdejo

2009

Agradecimientos

A Felisa Verdejo, mi directora de tesis, por su paciencia, indicaciones y dedicación. A Beatriz Barros porque empezó esta andadura conmigo y sin ella hubiera sido imposible llegar hasta donde estoy hoy. A Emilio Julio Lorenzo y Carlos Celorrio por su implicación activa en cada proceso de desarrollo circunscrito directa o indirectamente a este trabajo. A José Ignacio Mayorga y Yolanda Calero, por su apoyo incondicional en lo emocional y humano. A todos mis compañeros de departamento que fueron testigos día a día de este desarrollo. Y finalmente a mi familia por estar siempre ahí. A todos, Gracias.

Resumen

El aprendizaje colaborativo es un paradigma pedagógico de relativa reciente aparición. De acuerdo a esta nueva corriente, para desarrollar experiencias de instrucción significativas es conveniente abandonar los clásicos esquemas de transmisión de conocimientos entre alumno y profesor en favor de otros donde los estudiantes adquieren un rol más participativo. El cambio fundamental consiste en organizar a los estudiantes en pequeños grupos de colaboración para que realicen conjuntamente una serie de actividades de aprendizaje. Inmersos en el desarrollo de las mismas, los estudiantes aprenderán debido a la necesidad de reconciliar por consenso los conflictos cognitivos surgidos en la base de las interacciones sociales. La labor del profesor consiste entonces a diseñar y monitorizar las actividades de aprendizaje garantizando que dichos conflictos sucedan efectivamente.

La tecnología, y en particular el campo del aprendizaje colaborativo soportado por computador (CSCL, en inglés), pretende dar soporte a esta problemática. Actualmente se han desarrollado diversas plataformas de aprendizaje que se centran, fundamentalmente, en dar soporte a la interacción colaborativa. Sin embargo, no debemos olvidar que lo más importante para desarrollar este tipo de experiencias es realizar un apropiado diseño de las mismas. Por tanto, los entornos virtuales de aprendizaje colaborativo deberían dar un soporte completo a estas tareas.

En este trabajo de tesis presentamos Pelican, una plataforma de aprendizaje colaborativo que da soporte integral a todos los procesos que intervienen en el diseño y desarrollo de este tipo de experiencias. En Pelican es posible especificar formalmente escenarios de aprendizaje colaborativo y posteriormente ponerlos en práctica orquestados por la propia plataforma. La aportación principal a este respecto radica en las capacidades de intervención que presenta Pelican para adaptar puntualmente tanto la plataforma en si misma como las herramientas externas integradas dentro de ella a las necesidades de cambio de un flujo de trabajo instruccional. Esto permite capturar los aspectos dinámicos que aparecen de forme recurrente inherentemente vinculados a la mayoría de los escenarios de esta naturaleza.

Abstract

Computer supported collaborative learning is a recent pedagogical paradigm. According to this new theoretical tendency, to undertake significative instructional experiences it is convenient to leave the classical transmission of knowledge approaches among teacher and students in favour of others where students can take a more participative role. The main change lies on organizing students in small collaboration groups to let them carry out together a set of learning activities. Students, whilst being immersed in the activities development, will learn due to the necessity of meeting an agreement on cognitive conflicts arisen from social interactions. The teacher's responsibility consists of designing and monitoring learning activities in order to make sure that cognitive conflicts can take place.

Technology and Computer Supported Collaborative Learning (CSCL) in particular, aim to solve this kind of matters. In recent years, several learning environments have been developed mainly focused on supporting collaborative interaction. However, we cannot forget that the most important thing to undertake this kind of experiences is carrying out an appropriate design. Hence, Virtual Learning Environments should support this kind of tasks fully.

In this Ph. D. work, we introduce Pelican, a collaborative learning platform that wholly supports all the processes related with the design and development of this kind of experiences. Pelican allows specifying collaborative learning scenarios in a formal way and then putting them to practice while being managed by the platform. The main contribution of this work is focused on the intervention capabilities of Pelican to punctually adapting both the platform itself and the integrated external tools to the changing necessities of an instructional workflow. These features allow catching the dynamic aspects inherently bounded to most of the collaborative learning scenarios.

Índice general

1 Introducción 1

1.1. Introducción 2

1.2. Motivaciones 4

1.3. Objetivos 5

1.4. Marco de trabajo 6

1.5. Organización del documento 7

2 Estado del arte 9

2.1. Introducción 10

2.2. El aprendizaje colaborativo 11

2.3. Teorías psicológicas de aprendizaje colaborativo 14

2.4. Métodos pedagógicos de aprendizaje colaborativo 22

2.5. Soporte computacional al aprendizaje colaborativo 34

2.5.1. Soporte a la estratificación social 35

2.5.1.1. Fase de diseño 35

2.5.1.2. Fase de desarrollo 36

2.5.2. Soporte a la colaboración 38

2.5.2.1. Fase de diseño 38

2.5.2.2. Fase de desarrollo 46

2.5.3. Soporte a la integración 48

2.5.4. Soporte a la adaptación 53

2.6. Conclusiones 56

3 Solución propuesta 57
3.1. Introducción 58
3.2. Visión arquitectónica de Pelican 60
3.3. Soporte a la estructuración social 63
3.3.1. El subsistema social 68
3.3.2. La interfaz del subsistema social 74
3.4. Soporte a la colaboración 81
3.4.1. El subsistema de colaboración 90
3.4.2. La interfaz del subsistema de colaboración 95
3.5. Soporte a la intervención adaptativa 105
3.5.1. El subsistema de intervención 135
3.5.2. La interfaz del subsistema de intervención 142
3.6. Soporte a integración de herramientas externas 150
3.6.1. El subsistema de integración 161
3.6.2. La interfaz del subsistema de integración 167
3.7. Conclusiones 169
4 Desarrollo de pruebas y resultados 171
4.1. Introducción 172
4.2. Ecosistema tecnológico de ENLACE 173
4.3. Una Librería de scripts de adaptación 181
4.4. Experiencias en escenarios reales 188
4.5. Cinco escenarios colaborativos de ejemplo 191
4.5.1. Procedimiento Jigsaw 191
4.5.2. División colaborativa del trabajo 201
4.5.3. Debate colaborativo 213
4.5.4. Elección de delegados 223
4.5.5. Estratificación social por evaluación de resultados 232
4.6. Conclusiones 238
5 Conclusiones y líneas futuras 243
5.1. Introducción 244
5.2. Conclusiones 245

5.3. Líneas futuras 255

Bibliografía 259

A. Integración de herramientas externas A-1

A.1. Introducción A-2

A.2. Servicios Web de Pelican A-2

A.3. Integración de herramientas de ENLACE A-6

A.3.1. La herramienta LOR A-6

A.3.2. La herramienta AGORA A-10

A.3.3. La herramienta CHAT A-11

A.3.4. La herramienta CARDS A-12

A.3.5. La herramienta COMPO A-13

A.3.6. La herramienta COMET A-14

B. Instalación y configuración de Pelican B-1

B.1. Introducción B-2

B.2. Instalación de la plataforma Pelican B-2

B.3. Configuración de la plataforma Pelican B-3

Índice de figuras

3.1. Arquitectura general de la plataforma Pelican. 62
3.2. Niveles de especificación social en Pelican. 67
3.3. Diagrama conceptual en UML del subsistema social. 73
3.4. Interfaz para el diseño de actores. 76
3.5. Interfaz para el diseño de plantillas de grupo. 77
3.6. Interfaz para la administración de usuarios. 78
3.7. Interfaz para la administración de grupos. 80
3.8. Espacios de trabajo organizados a distintos niveles sociales. 83
3.9. Ciclo de vida de la actividad. 86
3.10. Niveles de especificación de la colaboración en Pelican. 88
3.11. Modelo conceptual en UML del subsistema de colaboración. 94
3.12. Interfaz para el diseño de plantillas de proyecto. 96
3.13. Interfaz para el diseño de plantillas de actividad. 97
3.14. Interfaz para la administración de proyectos. 99
3.15. Interfaz para la administración de actividades. 101
3.16. Interfaz del espacio de trabajo. 102
3.17. Interfaz de proyectos desplegados en el espacio de trabajo. 103
3.18. Interfaz de actividades desplegadas en espacio de trabajo. 104
3.19. Modelo de objetos de las variables implícitas de dominio. 117
3.20. Proceso de ejecución de scripts dentro de un contexto. 129
3.21. Proceso de ejecución de scripts basada en reglas. 133
3.22. Niveles de especificación del subsistema de intervención. 134

3.23. Modelo en UML de la ejecución de scripts de adaptación. 138
3.24. Modelo en UML de ejecución de scripts basada en reglas. 142
3.25. Interfaz para la administración de scripts de adaptación. 143
3.26. Interfaz del explorador de contextos de usuario. 144
3.27. Interfaz de la consola de depuración de scripts. 145
3.28. Interfaz de la consola para la ejecución de scripts. 146
3.29. Interfaz para la administración de políticas y reglas políticas. 147
3.30. Interfaz para la consola de políticas. 149
3.31. Ejemplo de integración de herramientas en Pelican. 152
3.32. Mecanismos de integración de Pelican. 158
3.33. Niveles de especificación del subsistema de integración. 160
3.34. Modelo en UML del subsistema de integración. 166
4.1. Ecosistema tecnológico del proyecto ENLACE. 180
4.2. Fases del escenario de Jigsaw. 191
4.3. Modelo de sociedad y comunidad del escenario de Jigsaw. 193
4.4. Flujo de trabajo instruccional del escenario de Jigsaw. 194
4.5. Fases del escenario de división colaborativa del trabajo. 202
4.6. Flujo de trabajo del escenario de división del trabajo. 206
4.7. Fases del escenario de debate colaborativo. 214
4.8. Flujo de trabajo del escenario de debate colaborativo. 216
4.9. Fases del escenario de elección de delegados. 224
4.10. Flujo de trabajo del escenario de elección de delegados. 226
4.11. Fases de desarrollo del escenario de estratificación social. 233
4.12. Flujo de trabajo del escenario de estratificación social. 234
A.1. Comunicación entre la plataforma y las herramientas. A-3

Índice de tablas

2.1. Tareas de las fases y procesos de aprendizaje colaborativo. 14
2.2. Resumen los métodos pedagógicos aprendizaje colaborativo. 34
2.3. Relación comparativa de los entornos de aprendizaje. 37
2.4. Clasificación de los medios de interacción más populares. 47
3.1. Vocabulario de permisos asociados a la entidad Actor. 70
3.2. Colección de variables implícitas de dominio. 116
3.3. Familia de tareas de dominio. 122
3.4. Colección de tipos de eventos gestionados por Pelican. 124
3.5. Aportación de cada subsistema a especificación de escenario. 170
5.1. Relación entre subsistemas de Pelican y tipos de procesos 254
A.1. Servicios Web de Pelican. A-6
A.2. Servicios proporcionados por la herramienta LOR. A-7
A.3. Familia de eventos de la herramienta LOR. A-8
A.4. Familia de eventos de la herramienta LOR. A-9
A.5. Servicios de la herramienta AGORA. A-10
A.6. Servicios Web de la herramienta AGORA. A-10
A.7. Familia de eventos de la herramienta AGORA. A-11
A.8. Servicios de la herramienta CHAT. A-11
A.9. Familia de eventos de la herramienta CHAT. A-12
A.10. Servicios de la herramienta CARDS. A-12
A.11. Servicios de la herramienta COMPO. A-13
A.12. Servicios de la herramienta COMET. A-14

Índice de listados

3.1. Ejemplo de script en P# para dar soporte a la evaluación. 125
3.2. Regla que aplicar el script 3.1 al terminar cada actividad. 126
4.1. Regla de creación de la sesión de votación de temas expertos. 196
4.2. Regla de creación de grupos expertos. 196
4.3. Regla de creación de sesión de votación de grupos expertos. 197
4.4. Regla de enrolado de estudiantes en grupos expertos. 198
4.5. Regla de despliegue de actividades expertas. 199
4.6. Regla de creación de equipos de trabajo. 200
4.7. Regla de despliegue de actividades Jigsaw. 200
4.8. Especificación inicial de la lógica de secuenciamiento 204
4.9. Especificación transformada de la lógica de secuenciamiento. 205
4.10. Regla para el despliegue de las actividades. 207
4.11. Regla de creación del motor de workflow. 208
4.12. Regla de arranque del motor de workflow. 208
4.13. Regla de transición de actividad 209
4.14. Regla de creación de subproyectos 210
4.15. Regla de formación de equipos colaborativos. 211
4.16. Regla de formación de equipos colaborativos. 211
4.17. Regla de transformación del flujo de trabajo. 212
4.18. Regla de creación de votación inicial y final. 218
4.19. Regla de formación de colectivos de debate. 219
4.20. Regla de arranque último turno para participantes. 220

4.21. Regla de cambio de turno al siguiente participante. 221
4.22. Regla de arranque del turno de palabra de los anotadores. 222
4.23. Regla de arranque del turno de palabra de los anotadores. 223
4.24. Especificación de la lógica de secuenciamiento. 225
4.25. Regla de creación del motor de workflow. 227
4.26. Regla para el cálculo de indicador nCandidaturas. 228
4.27. Regla de transición de actividad. 229
4.28. Regla para el cálculo del indicador nMessages. 230
4.29. Regla de denegación del permiso de intervención. 230
4.30. Regla de concesión del permiso de intervención. 231
4.31. Regla de concesión del permiso de intervención. 231
4.32. Regla para el inicio de la sesión de debate. 235
4.33. Regla de recolección de test de estudiantes. 235
4.34. Regla de formación de pares. 236
B.1. Extracto del fichero de configuración hibernate.properties. B-4
B.2. Extracto del fichero de configuración society-config.xml. B-4
B.3. Extracto del fichero de configuración society-config.xml. B-6
B.4. Extracto del fichero de configuración society-config.xml. B-7

1 Objetivos y motivaciones

En los últimos años, el aprendizaje colaborativo está cobrando gran aceptación. Este nuevo paradigma pedagógico promueve la realización de actividades formativas en grupo. De esta manera, se suelen obtener mejores resultados en el aprendizaje que en el modelo clásico de transmisión de conocimiento entre el profesor y los alumnos ya que éstos últimos interactúan entre sí para alcanzar una solución consensuada al problema planteado. Pero para que el aprendizaje tenga lugar de forma significativa, se ha señalado que es necesario prestar especial atención al diseño de las experiencias. En efecto, para cada situación formativa particular, éstas deben ser ideadas de tal forma que se maximice la probabilidad de provocar los tipos de interacción colaborativa que son identificados como pedagógicamente beneficiosos.

Dentro de este campo, el aprendizaje colaborativo soportado por computador proporciona soluciones para facilitar el desarrollo de este tipo de experiencias mediante el uso de tecnología. Sin embargo, la mayoría de sus esfuerzos han ido dirigidos, generalmente, a proporcionar medios de interacción colaborativa que permitan el desarrollo de las actividades. Aunque esto es conveniente y necesario, no debemos olvidar que el aprendizaje significativo en colaboración sólo surge cuando se realiza un adecuado diseño de la experiencia. Por tanto, la tecnología debería proporcionar mecanismos de especificación formales y completos para la descripción de este tipo de escenarios y en particular dar cobertura a los aspectos dinámicos inherentes a la mayoría de las experiencias. En este capítulo presentamos esta necesidad como objetivo primordial de nuestro trabajo de tesis.

1.1. Introducción

En los últimos tiempos, los expertos en educación han advertido la necesidad de inyectar, dentro del sistema educativo, nuevas tendencias y métodos pedagógicos. Éstas proponen un cambio en la manera de concebir la educación ya que confieren al estudiante un rol mucho más activo dentro de las aulas. Las propuestas son de diversa índole pero todas han terminado confluyendo en los beneficios que supone articular sesiones de trabajo en grupo para realizar actividades formativas de manera conjunta. Estas actividades, aunque a la vez demandan una mayor cantidad de tiempo para su desarrollo, están demostrando grandes resultados en el aprendizaje.

En el **modelo pedagógico clásico** [Skinner, 1982] [Watson, 1913], caracterizado por la transmisión de conocimientos directa del profesor al alumno, el mayor inconveniente radica en que el estudiante es un agente pasivo que se limita a recibir las explicaciones de un profesor. La falta de participación del estudiante hace que éste no se vea implicado en su propio proceso de instrucción y rápidamente se pierde el interés y la motivación por aprender. Además, este método de enseñanza puede resultar apropiado para transmitir un conocimiento tácito, explicito y fáctico. Pero cuando se trata de adquirir otro tipo de habilidades, bien sea cognitivas, como la resolución de problemas complejos y mal definidos, o sociales, como por ejemplo la capacidad de encontrar soluciones consensuadas a determinados problemas en grupo, esta aproximación resulta ineficaz.

Haciéndose eco de esta problemática se ha señalado la necesidad de permitir que sea el estudiante el que construya su propio conocimiento a través de procesos de interacción con el entorno. El **aprendizaje constructivista** [Ausubel, 1983] [Bruner, 1997] [Piaget, 1928] [Piaget, 1985] constituye un primer cambio paradigmático en la forma de concebir la educación. Este modelo pedagógico estipula que los procesos educativos deben enmarcarse dentro de escenarios donde el estudiante participe en actividades formativas que le permitan adquirir proactivamente el conocimiento. Esto permite rescatar el contexto al que éste se circunscribe y que había sido perdido en el enfoque clásico, lo cual es una importante mejora como han apuntado diversos autores [Winograd & Flores, 1986] [Clancey, 1997].

Pero además, dentro del enfoque constructivista, el nivel de participación del profesor y el alumno se ha invertido con respecto al modelo pedagógico clásico. Ahora, la responsabilidad del profesor se centra en diseñar, monitorizar y gestionar el desarrollo de actividades formativas que obliguen al estudiante a seguir un proceso cognitivo de razonamiento lógico hasta adquirir el conocimiento referido dentro de los objetivos de la actividad. El nivel de participación conferido a los estudiantes en este paradigma viene a resolver serios problemas de la aproximación pedagógica clásica. En primer lugar, los problemas de falta de motivación quedan considerablemente mitigados ya que ahora el estudiante se sabe responsable de su propio aprendizaje. En segundo lugar, el alumno aprende más significativamente ya que este proceso de aprendizaje permite consolidar el nuevo conocimiento enlazándolo a las estructuras cognitivas del sujeto.

En efecto, [Ausubel, 1983] explica este hecho afirmando que el aprendizaje es un proceso que reside en el anclaje de nuevas estructuras cognitivas – aquellas que se están aprendiendo – sobre otras ya existentes que forman parte del conocimiento previo del alumno. Cada estudiante tendrá un nivel de conocimientos previo potencialmente distinto al de los demás alumnos. Por tanto, debe ser éste el que dirija su instrucción porque sólo él conoce conjuntamente lo nuevo – el nuevo conocimiento presentado en la actividad – y lo previo – el conocimiento ya adquirido. Sólo él sabe el camino que debe recorrer para vincular las nuevas estructuras cognitivas a las anteriores. Este vínculo, articulado a través de la interacción con el entorno, ofrece mejores resultados que los del enfoque clásico, de manera que lo aprendido por este procedimiento consolida un conocimiento mucho más estable.

De manera similar, [Piaget, 1985] incide en que el aprendizaje debe entenderse como un proceso de equilibrio entre el conocimiento del estudiante, que le permite entender la realidad que le rodea y el nuevo conocimiento que le permitirá interpretar un contexto desconocido, presentado a través de actividades formativas. El estudiante debe reconciliar las estructuras cognitivas anteriores con las nuevas para adquirir un conocimiento integrador. En este sentido, el aprendizaje surge en el momento en que se le presenta al alumno una situación de conflicto cognitivo. Según esto, el trabajo del profesor consistiría en diseñar las actividades de manera que se garantice la existencia de dicho conflicto.

Los investigadores de la nueva corriente neopiagetiada de Genova conocida bajo el nombre de **constructivismo social** [Doyse y Mugny, 1984] [Hogan et al. 2000] afirman que el conflicto cognitivo puede buscarse no tanto dentro del propio individuo sino, adicionalmente, en las interacciones sociales entre los estudiantes realizando actividades de forma conjunta. Estos autores, proponen que las actividades sean realizadas entre pares de estudiantes de manera que el conflicto no provenga tanto del conocimiento descubierto durante el transcurso de la actividad sino también de las colisiones naturales que se darán al confrontar estructuras cognitivas potencialmente distintas de varios estudiantes trabajando juntos.

El modelo de interacción horizontal, entre pares de igual a igual, facilita el aprendizaje. Los estudiantes involucrados en este tipo de actividades formativas tendrán menos reparo en participar activamente en los procesos instructivos diseñados de esta forma que en aquéllos donde se establece una interacción caracterizada por una marcada dependencia jerárquica entre profesor y alumno. Esto vuelve a redundar en un factor beneficioso frente al enfoque convencional ya que se fomenta aun más la participación de los estudiantes implicándoles de una manera más activa en la instrucción.

[Vigotsky, 1978], fundador de la **teoría histórico cultural soviética**, incide igualmente en los beneficios encontrados al desarrollar actividades en grupo. Este autor postula que el aprendizaje tiene lugar en dos planos, primero en el interno del individuo, y sólo después en el plano social. Las conclusiones adquiridas por el estudiante en el plano individual son contrastadas con las perspectivas de los demás en el discurso

social antes de ser interiorizadas definitivamente por éste. Dicho de otra forma, la consolidación de las reflexiones mentales están supeditadas al plano social. La propuesta de Vigotsky es formar pares heterogéneos de estudiantes con un nivel cognitivo distinto para garantizar el aprendizaje. De este esquema, típicamente, cabría esperar que el aprendizaje solamente se produjese en una dirección: el que sabe menos aprende del que sabe más. Sin embargo algunos autores [Renkl, 1997] han señalado que el mero hecho de tener que explicar un concepto a alguien que no lo conoce supone más beneficios pedagógicos para el que lo explica que para el que recibe la explicación.

Estas dos últimas tendencias – el constructivismo social y la teoría histórico cultural soviética – han sido las principales precursoras del **aprendizaje colaborativo**, un nuevo paradigma pedagógico que goza actualmente de cierta aceptación debido a sus considerables ventajas con respecto a sus predecesores. Desde el punto de vista metodológico, esta aproximación comparte los principios constructivistas. Sin embargo, ahora la dimensión social le confiere nuevas posibilidades. Dentro de este paradigma, el conocimiento puede y debe ser construido socialmente a partir de procesos de negociación. Además la realización de actividades colaborativas fomenta la capacidad de trabajo en grupo, la responsabilidad social, la capacidad de organización, etc.

1.2. Motivaciones

El aprendizaje colaborativo ha estado apoyado, casi desde sus inicios, por la tecnología. El CSCL, siglas en inglés de Aprendizaje Colaborativo Soportado por Computador, es un área de investigación multidisciplinar orientada a dar soporte computacional a las experiencias de esta naturaleza [Krochmann, 1996].

En este sentido, la mayoría de los esfuerzos han ido dirigidos a proporcionar diversos medios de interacción para facilitar la colaboración entre los participantes de los procesos de instrucción. En efecto, los entornos virtuales de aprendizaje (VLE en inglés) tales como [Blackboard] [Moodle] o [Alf] son las soluciones más consolidadas a este respecto. Estos sistemas permiten articular experiencias formativas sobre un entramado social subyacente y proporcionan una colección de herramientas que ofrecen diversos mecanismos de interacción tales como repositorios de recursos compartidos, sistemas de mensajería, foros de discusión, herramientas de votación, etc.

Pero, como apuntábamos en la presentación de este capítulo, para realizar con éxito experiencias de aprendizaje colaborativo se debe prestar especial atención al diseño de las mismas con el fin de garantizar que se produzcan las interacciones sociales esperadas. En este sentido, cabe destacar dos aspectos fundamentales que, a nuestro juicio deberían atender las herramientas de soporte al aprendizaje colaborativo:

- **El diseño de escenarios de aprendizaje colaborativo.** Dado que el diseño es una parte integral del desarrollo de experiencias de aprendizaje colaborativo, los entornos deberían dar soporte a la especificación formal y computable de escenarios de aprendizaje para constituir artefactos que puedan ser aplicados a distintos contextos socio-cognitivos y reutilizados en diferentes situaciones.

- **El desarrollo de experiencias de aprendizaje colaborativo.** Los sistemas de soporte al aprendizaje colaborativo deben proporcionar los mecanismos tecnológicos oportunos para poder poner en práctica experiencias instructivas de acuerdo a tales diseños.

En relación al primer punto, durante nuestro estudio, hemos encontrado una gran deficiencia por parte de las herramientas. Como se analizará en el siguiente capítulo, los entornos VLE dan soporte, en mayor o menor medida, a la estratificación social y a la estructuración de actividades descuidando la monitorización, evaluación y control de las mismas de acuerdo al enfoque colaborativo. Pero, al respecto de la especificación, lo que ningún sistema proporciona son mecanismos eficaces para capturar los aspectos dinámicos que aparecen de forma inherente al desarrollo de la mayoría de las experiencias instruccionales. Esta es, sin embargo, una cuestión importante que constituye una aportación fundamental de nuestro trabajo de tesis.

Con respecto al segundo de los aspectos – el soporte al desarrollo de las experiencias de aprendizaje – nosotros afirmamos que los entornos colaborativos, más allá de ser meros proveedores de una colección canónica, cerrada y, por lo general insuficiente, de medios de interacción, deben constituir plataformas de integración donde herramientas externas de soporte a la interacción colaborativa, potencialmente desarrolladas por terceras partes, se incorporen de manera sencilla y eficaz al entorno de aprendizaje. La sencillez significa aquí que la integración no debe imponer fuertes restricciones tecnológicas aunque éstas estén respaldadas por estándares consolidados. La eficacia se refiere a que la integración debe permitir un grado de interoperabilidad y cohesión suficiente como para que ofrezca al usuario una sensación de continuidad en el uso de las herramientas, adaptándose éstas puntualmente a los requerimientos, potencialmente cambiantes, del flujo de trabajo instruccional.

1.3. Objetivos

Una vez establecidos las motivaciones que nos han traído hasta aquí, estamos en disposición de exponer de manera extensiva y precisa cuáles son los objetivos que perseguimos en este trabajo de tesis:

- **Estudio del arte del aprendizaje colaborativo soportado por computador.** En primer lugar, pretendemos hacer un estudio del paradigma de aprendizaje colaborativo centrándonos en describir la dimensión psico–pedagógica del mismo para después, desde un punto de vista más tecnológico, presentar las aportaciones que se han hecho dentro del campo del CSCL enfocadas en el soporte al diseño social, colaborativo, de integración y de adaptación, temas centrales de nuestro trabajo de tesis.

- **Obtención de un mecanismo de especificación de escenarios colaborativos.** Una vez realizado este estudio, se pretende proponer un mecanismo de especificación completo, formal y computable para describir los escenarios de esta naturaleza. El propósito perseguido en este sentido es obtener una colección cohesionada de artefactos software que definan experiencias colaborativas.

- **Desarrollo de una plataforma de aprendizaje de soporte a la colaboración.** Finalmente, se plantea el desarrollo de Pelican, una plataforma de aprendizaje orientada a la colaboración que de soporte a la especificación de escenarios de aprendizaje de acuerdo al formalismo anterior y que proporcione los mecanismos tecnológicos oportunos para poner en práctica las experiencias de aprendizaje colaborativo. En este sentido, la plataforma podrá entenderse como: 1) un entorno VLE orientado a la colaboración, 2) una herramienta de autor para la definición de escenarios de aprendizaje y 3) una arquitectura de integración donde incorporar herramientas de soporte a la colaboración desarrolladas por terceras partes.

1.4. Marco de trabajo

El desarrollo de este trabajo, y en particular la implementación de la plataforma de aprendizaje colaborativo Pelican, se inscribe dentro de dos proyectos de investigación vinculados al grupo LTCS del Departamento de Lenguajes y Sistemas Informáticos de la UNED. Sendos proyectos han proporcionado a lo largo de 4 y 5 años de duración un excelente marco de evaluación para poner a prueba la misma en diferentes escenarios pedagógicos y tecnológicos.

Muchas de las características que hoy hacen de Pelican una plataforma diferente de otras propuestas de soporte a la colaboración existentes han sido el resultado de un minucioso estudio de las necesidades que surgieron en el seno de estos dos proyectos. A continuación describimos en detalle cada uno de ellos:

- **El proyecto COLDEX.** COLDEX (Collaborative Learning and Distributed Experimentation) fue un proyecto liderado por el grupo COLLIDE de la universidad de Duisburg Essen (Alemania) que se desarrolló durante los años 2001 a 2005 [COLDEX]. En él se planteaban una colección de escenarios pedagógicos para que los estudiantes llevasen a cabo actividades colaborativas con ayuda de una colección de dispositivos de experimentación digital. Nuestra responsabilidad dentro de este proyecto fue doble. Por un lado proporcionar un mecanismo de soporte a la construcción de comunidades virtuales de aprendizaje con una estructura social fija formada por comunidades y grupos. Por otro, articular un medio de interacción colaborativa basada en la compartición de recursos. Como fruto del primer esfuerzo germinó una versión preliminar del subsistema social que luego formaría parte de la plataforma Pelican.

- **El proyecto ENLACE.** El proyecto ENLACE (Escuela y Naturaleza, Lugares para Aprender, Colaborar y Experimentar) comenzó en el año 2004 y terminó a finales del pasado año 2008 [ENLACE]. Este proyecto conjuga dos espacios para desarrollar experiencias de aprendizaje colaborativo: el aula y la naturaleza. Los escenarios pedagógicos planteados en el proyecto proponen actividades a los estudiantes para desarrollarlas en entornos naturales tales como la reserva natural de Doñana o los ecosistemas de El Monte del Pardo que se complementan con otras desarrolladas en el aula o en el laboratorio. Los estudiantes realizan las actividades haciendo uso de ordenadores, y de dispositivos PDA para tener acceso a un

entorno tecnológico constituido por un colectivo de herramientas de soporte a la colaboración. Dentro de este marco nació la plataforma Pelican como un elemento mediador que permitiese centralizar los aspectos de organización social y colaborativa de los escenarios pedagógicos y orquestar el uso transversal de diversas herramientas integradas dentro de la plataforma. Durante este tiempo, ha madurado el modelo social de Pelican, originalmente surgido en COLDEX, para ofrecer mayor flexibilidad en la definición de modelos sociales y se han desarrollado el modelo colaborativo, el modelo de integración y el modelo de intervención de la plataforma que serán presentados en el capítulo 3.

1.5. Organización del documento

La memoria del trabajo de tesis que presentamos en este documento ha sigo organizada en cinco capítulos y dos apéndices. A continuación describimos brevemente el contenido de cada una de estas partes:

- **Capítulo 1 Introducción.** El presente capítulo ha proporcionado una introducción orientada a posicionar al lector en el contexto pedagógico dentro del cual se ubica este trabajo. Asimismo, se ha discutido la problemática que motiva su desarrollo y los objetivos perseguidos.

- **Capítulo 2 Estado del arte.** En el segundo capítulo de este trabajo se realiza un estudio de las principales aportaciones llevadas a cabo por la comunidad científica dentro del aprendizaje colaborativo soportado por computador. Este estudio sirve para establecer un marco de trabajo teórico para la tesis que permite determinar los principales elementos que deben ser considerados cuando se diseñan este tipo de experiencias formativas.

- **Capítulo 3 Solución propuesta.** A lo largo del capítulo tercero presentamos la plataforma de aprendizaje colaborativo Pelican. En concreto, describimos en detalle la arquitectura interna de la misma así como los servicios, facilidades y características que ésta proporciona para articular experiencias de aprendizaje de esta naturaleza.

- **Capítulo 4 Desarrollo de pruebas y resultados.** En el capítulo cuarto presentamos la colección de pruebas que han sido llevadas a cabo para probar la viabilidad de nuestras propuestas tanto en entornos controlados como en escenarios de experimentación reales.

- **Capítulo 5 Conclusiones y líneas futuras.** En este último capítulo se exponen, a modo de resumen, las conclusiones obtenidas como resultado de la realización de nuestro trabajo y se proponen nuevas líneas de investigación e innovación tecnológica que darán continuidad a esta tesis.

- **Bibliografía.** En esta parte pueden encontrarse las referencias a las aportaciones científicas que constituyen el sustrato teórico sobre el que se apoya este trabajo, así como las direcciones de interés en internet a sistemas en línea.

- **Apéndice A Integración de herramientas externas.** En este apéndice describimos en detalle algunos de los mecanismos tecnológicos que presenta la plataforma Pelican para facilitar la integración de herramientas externas. El discurso se centra en la familia de herramientas de soporte desarrolladas dentro del contexto del proyecto [ENLACE] y que han sido utilizadas a lo largo del desarrollo de nuestras pruebas descritas en el capítulo 4.
- **Apéndice B Instalación y configuración de Pelican.** El apéndice B, por su parte, describe las tareas de instalación y configuración que es necesario realizar para poder llevar a explotación la plataforma de aprendizaje Pelican en situaciones reales. Aquí se detallan los pasos que es necesario realizar sobre una arquitectura cliente – servidor a partir del material proporcionado en el CD que acompaña a este libro así como los ficheros que pueden ser editados previamente al arranque para ajustar determinados aspectos de la plataforma.

2 Estado del arte

Como se describió a lo largo del capítulo primero de este trabajo, para poner en práctica experiencias de aprendizaje colaborativo es necesario prestar especial atención al diseño de las mismas. Atendiendo a esta preocupación, en este capítulo haremos una revisión de las principales aportaciones que se han realizado desde la pedagogía, la psicología y la informática al área multidisciplinar del CSCL (Aprendizaje Colaborativo Soportado por Computador). Nuestro objetivo en este sentido, es encontrar una caracterización lo más precisa posible de este paradigma pedagógico y entender mejor cómo se producen este tipo de experiencias, qué elementos intervienen en su desarrollo y cuáles son sus fundamentos teóricos.

En concreto, comenzaremos ofreciendo una caracterización del significado que encierra el término aprendizaje colaborativo si bien somos conscientes de que el uso del mismo ha sido empleado en numerosas ocasiones dentro de la literatura especializada de forma poco rigurosa. A continuación, trataremos de encontrar respuesta a dos preguntas fundamentales y complementarias. Desde un punto de vista psicológico describiremos cuáles son los fundamentos teóricos que justifican los beneficios de este paradigma, mientras que pedagógicamente, nos preocuparemos de ofrecer distintos métodos para inducir situaciones colaborativas. Por último, abordaremos las principales soluciones tecnológicas que, desde el campo de la informática, han sido desarrolladas para dar soporte computacional a este tipo de experiencias, centrándonos en aquéllas especialmente relacionadas con la propuesta que hacemos nosotros en este trabajo y que se discutirá en el siguiente capítulo.

2.1. Introducción

Tal y como se señaló en las motivaciones de este trabajo de tesis, cuando se ponen en práctica experiencias de aprendizaje colaborativo es preciso dedicar una especial atención al diseño de las mismas para garantizar que se produzcan los mecanismos de aprendizaje pertinentes durante su desarrollo. En las experiencias mediadas por computador, la tecnología no sólo debe dar soporte a la interacción entre estudiantes sino, adicionalmente, a su especificación formal. Precisamente para intentar caracterizar cómo es esta especificación formal es preciso dar respuesta de forma preliminar a una serie de cuestiones que constituyen el hilo conductual de este capítulo y que exponemos de forma resumida a continuación:

- **Qué es el aprendizaje colaborativo.** Quizá la cuestión más prioritaria a la que debe darse respuesta es relativa a qué debe entenderse por aprendizaje colaborativo. En efecto, para poder llevar a cabo experiencias instruccionales de este tipo, es necesario tener bien claro a qué nos estamos refiriendo cuando hablamos de aprendizaje colaborativo. Esto es importante por dos motivos. En primer lugar, para poder dirigir nuestros esfuerzos a alcanzar situaciones colaborativas. Y en segundo, para reconocer el aprendizaje en colaboración cuando éste se produzca.

- **Cómo se produce el aprendizaje colaborativo.** En torno a esta cuestión se circunscriben todas las teorías psicológicas que proporcionan los fundamentos en los que se apoya este paradigma de aprendizaje. Saber cómo tiene lugar el aprendizaje dentro de las estructuras cognitivas de los estudiantes cuando éstos realizan experiencias colaborativas, e incluso más aún, por qué el aprendizaje así obtenido es más significativo y más apropiado que el alcanzado con otros paradigmas, es importante ya que, de esta manera, podemos enfocar el diseño de nuestras experiencias instruccionales de forma más adecuada.

- **Cómo se induce el aprendizaje colaborativo.** Desde el punto de vista pedagógico, lo que interesa es saber inducir situaciones colaborativas en las cuales el aprendizaje tenga lugar de acuerdo a nuestros objetivos. En este sentido, se han establecido diversos métodos pedagógicos para articular experiencias colaborativas que fomenten determinados tipos de aprendizaje, algunos de ellos exclusivos de este paradigma, como puede ser la adquisición de habilidades de interacción social o la responsabilidad individual y grupal.

- **Cómo se soporta computacionalmente el aprendizaje colaborativo.** De cara al desarrollo de experiencias de aprendizaje colaborativo mediadas por computador en necesario discutir cuáles son los mecanismos y soluciones tecnológicas existentes en la actualidad para articular este tipo de experiencias instruccionales. La descripción de los artefactos descritos dentro de este aspecto será realizada alineándose con la de aquéllos que hemos desarrollado a lo largo de este trabajo y que, como tales, serán descritos en el capítulo tercero de esta tesis y puestos a prueba durante el capítulo cuarto. En este sentido, nuestro objetivo es proporcionar el marco de trabajo relacionado en el que se inscribe nuestra propuesta.

2.2. El aprendizaje colaborativo

El primer paso para obtener una especificación formal de los escenarios de aprendizaje colaborativo consiste en establecer una definición precisa que describa a qué nos estamos refiriendo cuando hablamos de este tipo de experiencias. Lamentablemente esta definición es difícil de encontrar y no existe consenso dentro de la comunidad científica acerca de qué significa exactamente este término. Incluso en ocasiones son utilizados otros vocablos para hacer referencia a situaciones de aprendizaje con iguales o similares características y propiedades tales como aprendizaje por equipos, aprendizaje grupal, aprendizaje basado en proyectos etc. Una definición común de aprendizaje colaborativo es la que se expone a continuación:

> **Aprendizaje colaborativo.** Aprendizaje colaborativo es toda situación en la que un grupo de individuos establecen sesiones de trabajo mediante las cuales intentan aprender algo en colaboración [Dillenbourg, 1999].

Aunque [Dillenbourg, 1999] señala que definiciones de este tipo no resultan de gran utilidad dado su carácter ambiguo e impreciso – no queda claro cuál es la estructura y dimensión de los grupos ni la duración de las sesiones de trabajo ni qué se entiende exactamente por colaboración – lo cierto es que sí es un punto de partida para caracterizar este tipo de situaciones. En este punto, es conveniente recordar una distinción común entre aprendizaje cooperativo y colaborativo. Nosotros sin embargo utilizaremos el término colaborativo para referirnos indistintamente a ambas aproximaciones:

- **Aprendizaje cooperativo.** En las experiencias de aprendizaje cooperativo, el profesor tiene el rol de experto en la materia y autoridad en el aula y es el encargado de formar los grupos, diseñar las actividades, realizar la asignación de las mismas, gestionar el tiempo y los recursos y monitorizar el desarrollo [Smith, 1996]. Según [Panitz, 1997], en estos casos el foco de atención es el profesor, la fuente de conocimiento es éste mismo y la motivación suele ser extrínseca al propio estudiante.

- **Aprendizaje colaborativo.** En las experiencias colaborativas por el contrario, son los estudiantes de los dirigen su propio proceso de instrucción. En este sentido [Panitz, 1997] afirma que el foco de atención está centrado en el estudiante, que la motivación es intrínseca y que éste, junto con las actividades que desarrolla, son la fuente de conocimiento.

Sin embargo, para caracterizar el diseño y desarrollo de experiencias de aprendizaje colaborativo desde una perspectiva computacional, que es lo que nos interesa en este trabajo, es preciso pararse a reflexionar acerca de cuáles son los elementos que entran en juego. En este sentido podemos destacar los siguientes:

- **Escenario de aprendizaje colaborativo.** El elemento central de las experiencias colaborativas es el escenario. Un escenario de aprendizaje es un contexto físico, social y tecnológico donde tiene lugar el desarrollo de una experiencia colaborativa que implica a un grupo de participantes y donde existen ciertos objetivos pedagógicos de carácter social y epistemológico que se pretende alcanzar.

- **Actividad de aprendizaje colaborativo.** Una actividad de aprendizaje colaborativo, o actividad colaborativa, es una especificación que describe un trabajo que debe ser realizado por un grupo de estudiantes dentro de un escenario. Los elementos y el nivel de detalle en la especificación de las actividades de aprendizaje pueden variar de una situación a otra ya que, por lo general, éstos dependen del contexto pedagógico donde se desarrolle la experiencia.

- **Flujo de trabajo instruccional.** El desarrollo de las actividades colaborativas de un escenario siempre se realiza de acuerdo a un flujo de trabajo instruccional. En ocasiones, los escenarios prescriben dicho flujo y en otras, por el contrario, son los estudiantes los que deciden el orden de realización de las actividades. Esto da pie a distinguir entre dos tipos de flujos de instrucción:

  – *Flujo de instrucción estático.* Un flujo de instrucción estático es aquél que no será alterado durante el desarrollo de la experiencia de aprendizaje para adaptarse a las necesidades cambiantes del escenario pedagógico.

  – *Flujo de instrucción dinámico.* Un flujo de instrucción dinámico es aquél que puede adaptarse durante el tiempo de desarrollo para atender a los requerimientos de cambio del escenario. Por ejemplo, ciertas decisiones tomadas por los estudiantes durante la experiencia pueden alterar el número y orden de desarrollo de las actividades.

- **Actor.** El diseño y desarrollo de las actividades colaborativas de un escenario involucran a una serie de participantes. Para hacer referencia a cada uno de ellos, utilizaremos el concepto de actor. Evidentemente, la colección de actores que aparecen en una experiencia particular es una decisión de diseño y por tanto depende de la naturaleza de la misma y de los objetivos pedagógicos perseguidos. Sin embargo nosotros, a lo largo de este texto, distinguiremos cinco actores que suelen aparecer comúnmente, en la mayoría de los escenarios:

  – *Estudiante.* El estudiante representa a cualquier alumno que aparece implicado en las actividades de aprendizaje colaborativo con el fin de adquirir conocimientos epistemológicos y habilidades sociales relacionadas, directa o indirectamente, con los objetivos pedagógicos establecidos en las mismas.

  – *Profesor.* En las experiencias colaborativas, el profesor es un agente encargado de dar soporte al proceso formativo dirigido por los estudiantes y una autoridad que gestiona el tiempo y los recursos en el caso de las experiencias cooperativas. Generalmente los participantes que encarnan este actor, también suelen ser monitores y diseñadores instruccionales. Sin embargo en este trabajo mantendremos esta distinción para mayor flexibilidad.

  – *Monitor.* Los monitores se encargan de observar y analizar el desarrollo de las experiencias de aprendizaje de uno o varios grupos a lo largo del tiempo así como de proporcionar retroalimentación a los mismos sobre su evolución y articular, si procede, los mecanismos de regulación previstos.

  - *Diseñador instruccional.* El diseñador instruccional se encarga de diseñar las experiencias de aprendizaje, determinando la colección de actividades que deben realizarse, especificando un flujo de instrucción si procede, caracterizándolas apropiadamente y estableciendo las estrategias y tácticas de monitorización y los mecanismos para la evaluación.
  - *Administrador.* El administrador tiene una responsabilidad organizativa en la experiencia de aprendizaje. En escenarios soportados por computador, éste es el encargado de gestionar el sistema de aprendizaje y realizar tareas de administración que faciliten la labor de los profesores, monitores y diseñadores.

- **Procesos colaborativo.** En las experiencias de aprendizaje colaborativo, las tareas realizadas por alguno, o varios, de los actores anteriores aparecen vinculadas a algún tipo de proceso de diferente naturaleza. En concreto, hemos identificado, dentro de la literatura, cuatro tipos o familias de procesos diferentes:
  - *Procesos de estratificación social.* Los procesos de estratificación social dan cobertura a todos los aspectos sociales propios del aprendizaje colaborativo.
  - *Procesos de colaboración y cooperación.* El trabajo colaborativo que tiene lugar cuando se desarrollan experiencias de aprendizaje se engloba dentro de los procesos de colaboración y cooperación.
  - *Procesos de monitorización y control.* Los procesos de monitorización y control tienen que ver con el análisis del desarrollo de las experiencias de aprendizaje y la articulación de mecanismos para la regulación de las mismas.
  - *Procesos de evaluación.* Los procesos de evaluación tienen por objeto llevar a cabo la evaluación del proceso de instrucción tanto en su dimensión sumativa como formativa.
- **Fase.** El trabajo de estos cuatro tipos de procesos se distribuye a lo largo del tiempo a través de dos fases que transcurren potencialmente en paralelo.
  - *Fase de diseño.* Durante la fase de diseño, los actores implicados en la especificación de experiencias de aprendizaje colaborativo – típicamente profesores, administradores y diseñadores instruccionales – definen el escenario con respecto a los cuatro procesos anteriores.
  - *Fase de desarrollo.* Durante la fase de desarrollo, los actores implicados en la experiencia – profesores, monitores y estudiantes – llevan a cabo el control y participan en la realización de las actividades colaborativas.
- **Producto colaborativo.** Las actividades de aprendizaje colaborativo requieren, potencialmente, de una serie de recursos de entrada para su desarrollo y generan, como resultado, una colección de productos de salida. Habitualmente, los recursos de entrada suelen coincidir con productos previos generados por tareas o actividades anteriores. Esto permite contemplar las experiencias de aprendizaje como procesos de producción donde las entidades constituyentes (actividades

colaborativas e individuales) tienen por objetivo (condición de terminación) generar ciertos productos. El tipo de productos obtenidos depende de la naturaleza y dominio de la experiencia. No obstante, en el capítulo 3 se discutirán cuatro productos que aparecen siempre que se diseñan escenarios en Pelican (modelo de sociedad, modelo de colaboración, modelo de integración y modelo de intervención).

	Estratificación social	*Colaboración y cooperación*	*Monitorización y control*	*Evaluación*
Diseño	- Elección de tipos de grupos - Determinación del tamaño - Selección de miembros - Decisión tipos de interacción social	- Selección de objetivos - Identificación de actividades - Decisión del nivel de estructuración - Selección del prompt - Diseño de scripts de colaboración	- Estudio de las cualidades a medir - Selección indicadores de análisis - Determinación métodos de cálculo	- Decidir qué y como evaluar - Decidir el propósito dela evaluación - Decidir a quién repercute y cómo - Decidir quién evalúa - Decidir cuándo se evalúa
Desarrollo	- Formación de grupos emergentes - Movimientos grupales - Movimientos individuales - Evolución de grupos - Establecimiento de reglas sociales	- Interacción colaborativa - Negociación colaborativa - Coordinación de tareas y recursos - Construcción colaborativa de significados	- Observación de la interacción - Análisis de los datos registrados - Toma de decisiones - Mediación en la experiencia	- Realizar evaluación formativa - Realizar evaluación sumativa

Tabla 2.1. Tareas frecuentes de las fases y procesos de aprendizaje colaborativo.

Para ilustrar con mayor claridad lo que acabamos de describir, presentamos en la tabla 2.1 un resumen de todas las tareas que se realizan, frecuentemente, a lo largo de las dos fases mencionadas y para cada uno de los cuatro procesos identificados. Lamentablemente una descripción más detallada de las mismas cae fuera de los objetivos de este trabajo.

2.3. Teorías psicológicas de aprendizaje colaborativo

Desde el punto de vista psicológico, el aprendizaje en general, y el colaborativo en particular, es un mecanismo cognitivo que se produce ocasionalmente cuando se coloca al individuo o al grupo en unas condiciones contextuales favorables. Así, el sentido del término es meramente descriptivo: se observa que dos o mas individuos han aprendido colaborando juntos y la colaboración es entendida como el mecanismo necesario para conseguir que el aprendizaje tenga lugar [Dillenbourg, 1999].

En esta dimensión se ponen de relieve los mecanismos cognitivos y sociales que conducen al aprendizaje. Así, el aprendizaje colaborativo puede contemplarse como un continuo entre dos polos [Dillenbourg et al., 1996]. En uno de ellos, el foco de atención es el individuo y las interacciones sociales que con éste se producen permitiendo así transformar su sistema cognitivo. En el otro polo, la unidad de análisis es el grupo y el propósito es entender cómo éste produce un conocimiento compartido del problema propuesto en la experiencia. A lo largo de este continuo, entre la perspectiva individual y grupal, podemos encontrar tres diferentes aproximaciones y posicionamientos teóricos: el constructivismo social, la teoría socio–cultural y la teoría de la cognición compartida. El primero de ellos, situado en el extremo individual, pone énfasis en los mecanismos de interacción social que conducen al aprendizaje. Por su parte, la teoría socio–cultural, más próxima a la perspectiva grupal, se preocupa por los mecanismos cognitivos que construyen conocimiento dentro de un individuo contrastado a nivel social. Finalmente, la cognición compartida, situada a caballo entre ambas aproxima-

ciones, reclama la importancia del contexto dentro del aprendizaje colaborativo. A lo largo de esta sección discutiremos en profundidad estas tres principales teorías que constituyen los fundamentos en los que se sustenta este paradigma y analizaremos los mecanismos psicológicos propios de cada una de ellas.

Teorías socio–constructivistas y mecanismos sociales

Las teorías de aprendizaje socio–constructivista provienen de los postulados del constructivismo expresados por [Piaget, 1985]. A juicio de este autor, los estudiantes construyen su conocimiento a través de un mecanismo de equilibrio, según el cual, intentan mantener consistentes y equilibrados sus modelos cognitivos. Así, el aprendizaje ocurre cuando se logran acomodar sus estructuras cognitivas para entender mejor el contexto que les rodea. A través de este mecanismo, se crean nuevos conceptos y se construye nuevo conocimiento. Desde esta perspectiva, el diseño de experiencias de aprendizaje requiere introducir en el entorno perturbaciones que desequilibren las estructuras cognitivas de los estudiantes con el fin de arrancar los mecanismos de equilibrio que conducen al aprendizaje. Es decir, es preciso diseñar experiencias que sitúen al estudiante ante una situación de conflicto cognitivo para provocar una nueva construcción de conocimiento.

Sin embargo, los investigadores de la corriente neo–piagetiana de Genova [Doyse & Mugny, 1984] [Hogan et al. 2000] consideran que el conflicto no debe buscarse exclusivamente dentro del propio individuo sino adicionalmente en las interacciones entre pares de estudiantes aprendiendo juntos. En efecto, cuando los estudiantes presentan puntos de vista divergentes acerca del problema en estudio, surgen conflictos de carácter ahora socio–cognitivo que fomentan las interacciones sociales. A juicio de estos autores, el aprendizaje es algo que surge en el seno de estas interacciones. En efecto, el desarrollo cognitivo individual, es considerado como una espiral de causalidad: un nivel dado de desarrollo individual, permite la participación en ciertos tipos de interacciones sociales, que produce a su vez nuevos estados de desarrollo en el individuo, lo que nuevamente permite acceder a interacciones sociales más sofisticadas y así sucesivamente [Dillenbourg et al., 1996].

El método experimental que confirma estas hipótesis está formado por dos fases independientes: fase de pre–test y fase de post–test, ambas separadas por una sesión en la cual los estudiantes trabajan bien por parejas (grupo de prueba) bien individualmente (grupo de control). Los resultados muestran que, ante ciertas condiciones, el aprendizaje en los estudiantes trabajando por parejas supera a aquéllos que aprenden de forma individual [Doyse & Mugny, 1984] [Blaye, 1988]. De esta manera, los conflictos socio–cognitivos son considerados, dentro de esta teoría, como uno de los mecanismos psicológicos centrales en la construcción colaborativa del conocimiento. En este sentido, se ha comprobado que tales conflictos surgen, de manera más efectiva, en situaciones simétricas entre pares de estudiantes que en experiencias de aprendizaje convencional, caracterizadas por una fuerte dependencia jerárquica entre profesor y alumno [Hogan et al., 2000] [Hatano & Inagaki, 1991]. Pero en realidad, el conflicto socio–cognitivo no es la condición para un adecuado aprendizaje,

sino más bien, la resolución del mismo a través de un esfuerzo de coordinación de los diferentes puntos de vista de los estudiantes. [Berkowitz & Gibbs, 1983 pag. 402] explican este hecho a través del concepto de discurso transactivo, similar al principio de interactividad de [Dillenbourg, 1999]. Un discurso transactivo se define como aquel razonamiento que opera sobre el razonamiento de los demás. De esta forma, es interesante determinar el grado, cuando menos relativo, de transactividad de las interacciones puesto que se ha comprobado que éste está positivamente correlacionado con los resultados de la construcción colaborativa del conocimiento [Teasley, 1997]. En línea con lo anterior, también se ha comprobado que las situaciones de interacción entre pares resultan en un grado de transactividad mayor que las interacciones entre profesor y alumno [Kruger, 1992] [Kruger & Tomasello, 1986]. De esta manera, a juicio de estos autores, los mecanismos psicológicos en los que debemos centrar nuestra atención son aquellos que fomenten la transactividad. En concreto, podemos destacar los siguientes [Weinberger, 2003]:

- **Externalización.** La externalización del conocimiento es un mecanismo dirigido a los demás miembros del grupo, mediante la cual, un estudiante verbaliza su punto de vista sobre el problema a resolver. Aunque se trate de una tarea de baja transactividad, lo ciento es que éste es considerado como uno de los mecanismos centrales de la construcción colaborativa de conocimiento motivado por la creación de situaciones sociales. Sin embargo, [Webb, 1989] advierte que la externalización puede no ser un mecanismo de aprendizaje colaborativo sino una indicación de que los estudiantes con ciertas habilidades de aprendizaje individual son capaces de verbalizar su propio conocimiento. En este sentido, la externalización puede tener una función mediadora pero no puede ser hecha responsable de los resultados obtenidos en las experiencias de aprendizaje colaborativo.

- **Elicitación.** La elicitación (solicitud de ayuda) es un mecanismo más transactivo que la externalización de conocimiento ya que tiene como propósito provocar una reacción en los estudiantes y fomentar la externalización. Así, un grupo de aprendizaje puede ampliar sus fronteras de conocimiento a través de la elicitación [Dillenbourg, 1995]. Pero más allá de eso, la elicitación permite que los estudiantes descubran *huecos* dentro de sus propias estructuras cognitivas, lo que puede ayudar a resolver conflictos [Renkl, 1997]. De esta forma, este mecanismo resulta beneficioso tanto desde el punto de vista individual como grupal. Sin embargo, a juicio de estos autores, en ocasiones la elicitación puede ser perjudicial puesto que los estudiantes corren el riesgo de hacerse dependientes de ella.

- **Construcción rápida de consenso.** En las experiencias de aprendizaje colaborativo, los estudiantes tienen que coordinarse entre sí para encontrar una conceptualización compartida del problema. Generalmente esto suele comenzar mediante un episodio de negociación conocido como construcción de consenso rápido. A través de él, los estudiantes se limitan a aceptar las contribuciones de sus compañeros en aras a poder avanzar en el discurso colaborativo aunque sin alterar su perspectiva del problema [Fischer, 2001]. Por eso, aunque este mecanismo presenta mayor nivel de transactividad que los anteriores [Teasley, 1997], aún supone

un nivel intermedio. En efecto, los resultados indican que la construcción rápida de consenso no se encuentra correlacionada linealmente con los resultados del aprendizaje colaborativo. De hecho, puede suponer un detrimento de la construcción de conocimiento si los alumnos utilizan este mecanismo en lugar de otros más pedagógicamente significativos para encontrar un consenso.

- **Construcción de consenso orientado a la Integración.** Durante las experiencias de aprendizaje colaborativo, los estudiantes tienen que mantener la conceptualización compartida del problema en estudio, lo cual puede ser descrito como una necesidad de integrar y acumular las perspectivas individuales de cada participante [Fischer et al., 2002]. De esta forma, enriquecen el entendimiento del problema incorporando las perspectivas de los demás. La construcción de consenso orientada a la integración, ha sido considerada como un mecanismo de alta transactividad [Berkowitz & Gibbs, 1983] [Teasley, 1997]. Sin embargo, es necesario advertir que ésta debe ser claramente diferenciada de la construcción rápida de consenso, aunque en la práctica dicha distinción resulte complicada. Al contrario que en esta última, la integración no es un simple acuerdo basado en la repetición o yuxtaposición de las perspectivas individuales, sino una combinación sustitutiva de posicionamientos [Roschelle, 1992]. La integración surge así cuando los estudiantes abandonan sus perspectivas individuales en favor de una nueva conceptualización compartida que emerge a partir de las interacciones con los demás. Un claro indicador de que se está produciendo consenso orientado a la integración es que los participantes muestran una tendencia activa a revisar sus puntos de vista influenciados por los argumentos persuasivos de sus compañeros [Keefer et al., 2000, pag. 77].

- **Construcción de consenso orientada al conflicto.** La construcción de consenso orientada al conflicto es considerada como el mecanismo de aprendizaje principal dentro de esta teoría [Teasley & Roschelle, 1993]. En efecto, como explicábamos al inicio de este apartado, al colocar a los estudiantes en una situación de confrontación con perspectivas divergentes se produce un desequilibrio en sus estructuras cognitivas que les induce, potencialmente, a reconsiderar sus conceptualizaciones con el fin de resolver tales conflictos [Piaget, 1928]. De esta manera, la construcción de consenso orientada al conflicto puede arrancar otros mecanismos transactivos tales como la externalización de conocimiento o la construcción de consenso orientada a la integración.

[Dillenbourg & Schneider, 1995] destacan que algunos resultados empíricos cuestionan la aproximación socio–constructivista. Por un lado, las divergencias entre las perspectivas de algunos estudiantes pueden no constituir verdaderos conflictos que conduzcan a situaciones de aprendizaje [Blaye, 1988]. Por otro, cuando los conflictos no son verbalizados, no se puede predecir cuales serán los resultados [Butterworth, 1982]. De aquí los autores extraen dos conclusiones. En primer lugar que un simple desacuerdo o malentendido puede resultar tan efectivo como un conflicto y en segundo lugar, que la verbalización de las interacciones generadas durante la resolución del conflicto fomenta el aprendizaje, en línea con la externalización del conocimiento.

Teorías socio–culturales y mecanismos cognitivos

En contraste con la perspectiva neo–piagetiana, que considera a las interacciones sociales como el contexto idóneo que beneficia la construcción del conocimiento, las teorías socio–culturales señalan que esta construcción es más bien un proceso social en sí mismo mediado por el lenguaje [Vygotsky, 1978]. La asunción básica aquí es que la construcción del conocimiento se produce en dos planos: primero en el plano interno del individuo y sólo después en el plano social. Así, el discurso social se corresponde con el discurso interno, de manera que éste último puede entenderse una externalización del desarrollo cognitivo propio del estudiante.

El conocimiento previo del estudiante es activado en el discurso social y se convierte en objeto de comparación. Como consecuencia, dicho conocimiento puede ser reorganizado a través de la negociación del significado [Dillenbourg & Baker, 1996] para adaptar las estructuras cognitivas del estudiante. Vygotsky señala, en este sentido, que el estudiante necesita ser orientado por un compañero más competente para ser asistido en la resolución de aquellos problemas que éste no sea capaz de resolver sólo. El autor describe esta orientación a través del concepto de *zona de próximo desarrollo* para caracterizar el potencial de aprendizaje de una persona en presencia de otra que le asiste. Este potencial puede desarrollarse a través de acciones e interacciones mediadas por un conjunto de herramientas entre las que cabe destacar, por su especial relevancia, el lenguaje. Expresado lo anterior en estos términos, el compañero más competente guía al otro estudiante a la zona de próximo desarrollo para aplicar estrategias adecuadas que permitan la resolución del problema. Estas estrategias, más o menos dependientes de dominio, son interiorizadas por el estudiante lo que le capacitará, en lo venidero, para resolver problemas de este tipo por sí mismo. De hecho, la interiorización es considerada el mecanismo psicológico central de esta teoría junto a algunos otros. A continuación repasamos someramente cada uno de ellos [Dillenbourg & Scheneider, 1995]:

- **Interiorización.** De acuerdo a esta teoría, el conocimiento es compartido por el contexto cultural en el que éste se produce y es articulado mediante el lenguaje. Efectivamente, el lenguaje sirve primero para construir una conceptualización a nivel individual y después, en el plano social, para someter a análisis el conocimiento así adquirido. El pensamiento es considerado, de esta manera, como una discusión que uno tiene primero consigo mismo, y después con sus compañeros. Este mecanismo de aprendizaje, a través de la participación en el discurso social, es conocido bajo el nombre de interiorización (internalization) [Vigotsky, 1978]. Los conceptos conferidos mediante la interacción con los compañeros más competentes son progresivamente integrados y usados como parte del razonamiento del propio estudiante. Sin embargo, la interiorización sólo ocurre si se dan ciertas condiciones. Una de ellas es que cada individuo puede sólo desarrollar conceptos que están dentro de su zona de próximo desarrollo. Esto es, en la vecindad del nivel cognitivo actual. Otra condición es que el estudiante menos capacitado no debe ser un agente pasivo sino un participante activo en la estrategia de resolución del problema [Rogoff, 1991].

- **Apropiación.** La apropiación es otro de los elementos psicológicos que conducen a situaciones de aprendizaje colaborativo. Éste puede ser entendido como el mecanismo mediante el cual el estudiante más competente intenta integrar en su propio plan de resolución del problema las acciones previas de su compañero, lo cual permite a este último reinterpretar sus propias acciones a partir de la interpretación que de éstas hizo el primero [Fox, 1987]. [Pea, 1993] expresa el fundamento de este mecanismo psicológico afirmando que a través de las interpretaciones que hacen los demás de nuestras acciones se puede llegar a comprender éstas mejor. De esta forma, el estudiante menos competente aprende progresivamente cómo ensamblar acciones elementales en una estrategia de resolución correcta y coherente.

- **Carga cognitiva compartida.** Cuando dos sujetos colaboran, éstos comparten una carga cognitiva implicada por la actividad formativa que están desarrollando. Espontáneamente el grupo distribuye esta carga entre sus miembros de acuerdo a una estrategia de división del trabajo horizontal [Dillenbourg, 1999]. De acuerdo a ella, aparecen nuevos roles durante el desarrollo de la actividad de manera que algunos miembros del grupo llevan a cabo las operaciones de nivel inferior mientras que otros se encargan de supervisar y monitorizar las actividades de sus compañeros. La distribución persigue en todo momento evitar redundancias y una consumición excesiva de recursos de manera que cada miembro sólo puede hacer uso de los recursos que le han sido asignados. El mecanismo de carga cognitiva compartida fomenta la adquisición de habilidades sociales propia de este paradigma. En efecto, este tipo de tareas no epistemológicas permiten a los estudiantes adquirir destrezas y habilidades en las tareas de gestión del trabajo en grupo tales como la interacción social, el reparto del trabajo, la asignación de responsabilidades, etc.

- **Social grounding.** El social grounding es un mecanismo mediante el cual los estudiantes intentan mantener la creencia de que sus compañeros han entendido lo que ellos quieren decir, al menos en un grado suficiente que permita llevar a cabo la actividad colaborativa que deben realizar. De esta manera, cada estudiante que interviene en el plano social, presta atención a lo que están entendiendo sus compañeros y, en caso de detectar malentendidos, intenta reparar éstos en base a interacciones sociales. Este mecanismo resulta fundamental ya que, como afirman [Roschelle & Teasley, 1995 pag. 70], el propósito último de las experiencias de aprendizaje colaborativo es construir y mantener una conceptualización compartida de un problema. Para [Clark & Brennan, 1991], el coste de mantener el social grounding está estrechamente relacionando con el medio de comunicación utilizado. Entre los factores que estos autores destacan para justificar dicho coste se encuentran el coste en el cambio del hablante y el coste en el display. El primero hace referencia a la dificultad de respetar los turnos de palabra y la sincronización de las interacciones cuando se emplean medios telemáticos no regulados. El segundo destaca la pérdida de la expresión enfática y gestual al utilizar este tipo de medios.

Teorías de la cognición compartida y la importancia del contexto

El concepto de *cognición compartida* está profundamente relacionado con la teoría de la cognición situada [Suchman, 1987] [Lave, 1988]. Para estos investigadores, el entorno en el cual se desarrollan las experiencias de aprendizaje colaborativo es una parte integral de la actividad cognitiva y no meramente un conjunto de circunstancias bajo las cuales se producen ciertos mecanismos cognitivos independientes del contexto. Inicialmente, el entorno al que se refieren los investigadores incluye tanto el contexto físico como el contexto social al que se circunscribe el aprendizaje. Sin embargo, influenciados por la antropología y la sociología, en los últimos años, el foco de atención se ha ubicado a lo largo del contexto social, de manera que ya no se habla de grupos de colaboración puntuales sino de verdaderas comunidades virtuales de aprendizaje donde los estudiantes participan.

Esta teoría ofrece una nueva perspectiva sobre las anteriores desde la cual diversos autores han puesto en tela de juicio los resultados experimentales de las mismas. Así, por ejemplo, [Perret–Clermont et al., 1991] advierten, con respecto al socio–constructivismo, que las respuestas de los estudiantes en los test están condicionadas por lo que éstos entienden que los experimentadores esperan de ellos. En la misma línea, [Wertsch, 1991] critica la aproximación socio–cultural: las interacciones sociales son estudiadas como si tuviesen lugar fuera de una estructura social. Es cierto que a través del lenguaje nosotros adquirimos una cultura específica y propia de una comunidad. Por ejemplo, cambiamos de gramática y vocabulario rápidamente al cambiar de contexto social. Pero sobre todo, más allá del vocabulario y la gramática, adquirimos una estructura de significados y relaciones sociales, que son fundamentales condicionantes para desarrollar las interacciones [Resnik, 1991].

De esta manera, tanto la metodología de estudio como las formulaciones teóricas de sendas aproximaciones previamente descritas son puestas en duda. [Perret–Clermont et al., 1991] vuelven a reclamar que las teorías metodologías de investigación se apoyan en la supuesta existencia de una clara separación entre lo social y lo cognitivo. Sin embargo, tal separación es discutible, puesto que la relación de causalidad de sendas dimensiones es, en realidad, circular y por tanto más compleja.

En este sentido, mientras que las teorías anteriores se centran en el plano inter–individual, cada una desde postulados diferentes, la teoría de la cognición compartida pone su atención en el plano social, donde las conceptualizaciones emergentes son analizadas como un producto del grupo. Por ejemplo, como comentamos con anterioridad, realizar explicaciones conduce a la construcción de conocimiento. Desde la perspectiva individualista, propia de las teorías socio–constructivistas, esto puede ser explicado a través del mecanismo de externalización del conocimiento. Sin embargo, desde la perspectiva de grupo, más cercana a las teorías socio–culturales y de cognición compartida, la explicación no se contempla como algo transferido desde el que explica hacia el que recibe la explicación, sino como un producto de grupo construido conjuntamente en un intento por parte de los participantes de entender las posiciones de sus compañeros [Baker, 1991].

En el campo del CSCL, donde se inscribe nuestro trabajo de tesis, esta teoría sirve para orientar de desarrollo de entornos de aprendizaje colaborativo que faciliten a los estudiantes la realización de actividades en condiciones próximas al contexto real donde éstas se producen. En este sentido, deben tenerse en cuenta ciertos factores relevantes a la hora de diseñar experiencias de aprendizaje situadas. A continuación mencionamos los más relevantes [Renkl & Mandl, 1995]:

- **La organización.** Este primer factor hace referencia al grado en el que la organización en la que se desarrolla el aprendizaje acepta los postulados de la construcción colaborativa del conocimiento. Se ha comprobado que los esfuerzos individuales de algunos profesores por realizar prácticas colaborativas rara vez llevan a los resultados deseados, puesto que la construcción colaborativa del aprendizaje es una disciplina que necesita ser integrada dentro del contexto de la rutina escolar y soportada tanto por los administradores y profesores como por los estudiantes. Estos últimos pueden necesitar experimentar repetidamente los beneficios de este paradigma para aceptarlo completamente y desarrollar competencias propias del mismo. Además, el factor organizacional también considera aspectos tales como el tamaño de los grupos, el tiempo de las sesiones de trabajo, etc.

- **La incentivación.** [Slavin, 1993] destaca la importancia de premiar la responsabilidad individual dentro del grupo como una manera de alcanzar una construcción colaborativa de conocimiento efectiva. El autor considera que los estudiantes necesitan sentirse interdependientes con respecto a la obtención de un premio compartido. A su vez, las contribuciones de cada miembro al grupo deben ser premiadas y reconocidas [Cohen, 1994].

- **La actividad.** Las actividades de aprendizaje colaborativo es otro de los factores centrales a la hora de desarrollar experiencias instruccionales efectivas. A este respecto, se han identificado ciertas condiciones que deben cumplir las actividades para alcanzar una construcción del conocimiento significativa. En primer lugar, las actividades deben ser lo suficientemente complejas como para que los estudiantes requieran compartir una colección de recursos para resolverlas. [Cohen, 1994] argumenta, en este sentido, que la mayor parte de las actividades de aprendizaje, son excesivamente simples y con una única posible solución correcta, de manera que los estudiantes no necesitan construir ni mantener una conceptualización compartida sobre la misma. Además, las tareas deben ser elementos motivadores del aprendizaje y la complejidad es el vehículo para conseguir este objetivo.

- **El estudiante.** El último factor relevante a tener en cuenta señala los prerrequisitos que un estudiante potencial de aprendizaje colaborativo debe presentar. Éstos pueden ser categorizados en cognitivos, tales como el conocimiento previo de los estudiantes o las estrategias y estilos de aprendizaje y emocionales y motivacionales como la ansiedad social (entendida como una percepción de amenaza de la realidad social subyacente), la incertidumbre relativa a la conveniencia del proceso colaborativo o el interés.

2.4. Métodos pedagógicos de aprendizaje colaborativo

Desde una perspectiva pedagógica, el aprendizaje se contempla como algo que puede y debe inducirse a través de la aplicación de métodos y estrategias. En el caso concreto del aprendizaje colaborativo estas estrategias determinan la forma en que debe ser diseñada la experiencia para que se alcancen los objetivos perseguidos. De estas forma, el sentido del término aprendizaje aquí es claramente prescriptivo: se pide que un grupo de estudiantes colaboren de una manera determinada y de acuerdo a unas normas establecidas porque se espera que de esa forma se produzcan ciertos tipos de aprendizaje [Dillenbourg, 1999]. Existen, en la literatura, una gran cantidad de métodos pedagógicos de diferente relevancia y nivel de abstracción. En los siguientes apartados presentamos aquéllos mas relevantes.

Learning Together

Learning together [Johnson & Johnson, 1994] es un método pedagógico que surgió a mediados de los 70 como un intento de romper los esquemas competitivos e individualistas vigentes en la educación formal en favor de una organización cooperativa que promovía cambios estructurales tanto en la forma de organizar las clases y las lecciones como en la concepción del aprendizaje desde el punto de vista organizacional, moviendo el foco de atención del individuo al grupo.

El método propone organizar grupos de trabajo heterogéneos de tamaño medio (de entre cuatro y cinco alumnos) y ponerlos a trabajar juntos para que generen como resultado un producto, que será evaluado al finalizar la experiencia. Las calificaciones que recibirán cada uno de los alumnos como resultado de dicha evaluación deben expresar el rendimiento del grupo en su conjunto. Learning together considera como esenciales los siguientes cinco principios a la hora de definir experiencias de aprendizaje efectivas:

- **Interdependencia positiva.** La interdependencia positiva se refiere al hecho de que la experiencia debe ser diseñada de tal forma que el éxito de cada individuo esté vinculado al éxito del grupo en su conjunto. De esta manera se consigue que los estudiantes se ayuden mutuamente para alcanzar el conjunto de metas compartidas. Los autores proponen diferentes maneras de estructurar la actividad para garantizar la interdependencia:
  - *Interdependencia de metas positiva.* En un diseño guiado por la interdependencia de metas, cada estudiante percibe que puede alcanzar sus objetivos individuales de aprendizaje si el resto de sus compañeros también lo hace. El grupo es así cohesionado en torno a un conjunto de metas en común. Desde el punto de vista del diseño, lo relevante es identificar dicho conjunto de metas y mostrar a los estudiantes la naturaleza compartida de las mismas. Comúnmente, éstas forman parte del material que debe ser aprendido.
  - *Recompensa positiva.* Otra forma de establecer la interdependencia es a través de la asignación de recompensas. De esta manera, cada miembro del gru-

po debe recibir una recompensa cada vez que uno de sus compañeros alcanza sus objetivos. Este principio puede aplicarse de forma complementaria a la interdependencia de metas. Por ejemplo, si todos los miembros de un grupo obtienen al menos un 90% de respuestas correctas en los tests individuales, entonces cada uno de ellos recibe una bonificación. En ocasiones se puede asignar a cada estudiantes tres calificaciones diferentes: 1) una calificación de grupo medida en términos de la productividad global de éste; 2) una calificación individual resultante de la evaluación de los test individuales y 3) puntos adicionales a todos los miembros del grupo si todos ellos alcanzan cierto criterio sobre el resultado de los test individuales.

– *Interdependencia de recursos positiva.* En las experiencias diseñadas con interdependencia de recursos, cada miembro del grupo dispone solamente de una parte de los recursos, información y materiales necesarios para llevar a cabo la actividad colaborativa. Esto obliga a combinarlos si se pretende alcanzar los objetivos perseguidos en la experiencia. A juicio de los autores, existen dos maneras de establecer la interdependencia de recursos. En primer lugar, se puede asignar a los miembros del grupo un número de recursos limitado, lo que les obligará a compartir, colaborativamente, los mismos. En segundo lugar, puede asignarse a cada participante un subconjunto del total de los recursos, lo que de nuevo fomentará la colaboración ya que para completar la actividad es necesario disponer de todos ellos. Esta segunda estrategia es conocida como *jigsaw* y será descrita, como método pedagógico, más adelante.

– *Interdependencia de roles positiva.* En las experiencias de aprendizaje diseñadas de acuerdo a la interdependencia de roles, a cada miembro del grupo se le asigna un rol diferente. Los roles son cuidadosamente definidos de tal manera que sean complementarios entre sí y se encuentren interconectados para que, en su conjunto, den cobertura a las responsabilidades necesarias para llevar a cabo la actividad colaborativa. Algunos ejemplos típicos de roles comúnmente utilizados son el lector, que aporta informaciones al grupo; el anotador, que registra las actuaciones de cada estudiante; el revisor, que se encarga de asegurar un nivel de aprendizaje homogéneo en los miembros del grupo; el animador, que se encarga de implicar a los estudiantes en la actividad o el trabajador, que realiza la construcción de conocimiento.

– *Interdependencia de tareas positiva.* La interdependencia de tareas se produce cuando se realiza una división del trabajo tal que las acciones de un participante tienen que ser completadas para que otro u otros participantes completen las suyas.

– *Interdependencia competitiva positiva.* La interdependencia competitiva se obtiene cuando la experiencia colaborativa es desarrollada de forma paralela y simultánea en varios grupo de estudiantes y éstos son animados para competir entre ellos, por ejemplo asignando una recompensa mayor al grupo que primero complete la actividad o al que mejores resultados saque.

- **Interacción promotora.** La interdependencia positiva conduce, de manera natural, a situaciones de interacción promotora, en las que los estudiantes se ayudan entre sí, compartiendo recursos, información y conocimiento, para alcanzar las metas que tienen en común y completar la actividad. Aunque la interdependencia positiva puede tener algunos efectos positivos sobre el aprendizaje, la interacción promotora es la que se encarga de fomentar la relaciones inter–personales, la adaptación psicológica al entorno y la adquisición de competencias sociales.

- **Responsabilidad individual.** El tercer principio esencial de diseño de las experiencias de aprendizaje colaborativo es la responsabilidad individual, según el cual las actividades deben estructurarse de tal forma que cada miembro del grupo sienta la responsabilidad individual que tiene para alcanzar los objetivos de las mismas. Una manera de conferir esta propiedad a las experiencias colaborativas es a través de la evaluación: las aportaciones de cada estudiante son valoradas y los resultados devueltos tanto al estudiante como al grupo para que éstos establezcan las medidas correctivas oportunas. A juicio de los autores, es importante que el grupo sepa, en todo momento, qué miembro necesita más ayuda e implicación en la actividad para evitar que unos participantes se apoyen en el trabajo de los demás. Esta situación, conocida como *holgazanería social*, puede darse especialmente en casos donde resulta complicado identificar las contribuciones de cada estudiante, cuando éstas son redundantes, o cuando los participantes no son responsables. Para garantizar la responsabilidad individual pueden destacarse las siguientes formas:

  – *Mantener reducido el número de miembros del grupo.* Una de las formas más efectivas de garantizar la responsabilidad individual es la definición de grupos pequeños de estudiantes. En éstos, la participación es mayor ya que la carga de trabajo es significativa para cada uno de sus miembros. En grupos grandes, por el contrario, la probabilidad de encontrar holgazanería social aumenta.

  – *Realizar test individuales.* La realización de test individuales con relativa frecuencia permite mantener controladas las condiciones bajo las cuales se produce el aprendizaje ya que se detectan con facilidad los alumnos que incumplen el principio de responsabilidad individual.

  – *Realizar exámenes orales aleatorios.* Otra forma de mantener controlada la responsabilidad individual consiste en hacer exámenes orales al azar a diversos miembros del grupo en cada sesión de trabajo.

  – *Monitorizar el comportamiento del grupo.* Mediante la observación directa y el registro de la frecuencia con que cada miembro del grupo participa en las actividades colaborativas puede determinarse el nivel de responsabilidad de los estudiantes.

  – *Asignar el rol de revisor.* También es posible definir un rol *revisor* dentro de la actividad y asignárselo a uno o varios miembros del grupo. Estos estudiantes deberán asegurarse de que los demás alumnos participen.

- **Desarrollo de habilidades de equipo.** El cuarto principio de este método pretende garantizar que los estudiantes adquieran ciertas habilidades sociales en la interacción con los demás durante el desarrollo de las experiencias colaborativas. En efecto, el aprendizaje en colaboración requiere de los estudiantes una serie de destrezas tales como la adquisición de confianza por parte de los demás miembros del grupo, la capacidad de comunicación de forma precisa y no ambigua, la aceptación de los compañeros y la resolución de los conflictos socio–cognitivos. Los autores afirman que el nivel de habilidades sociales adquirido por los estudiantes está directamente relacionado con el nivel de productividad del grupo. Así, cuantas más habilidades sociales tienen los estudiantes y cuanta más atención y recompensas por ellas ofrecen los profesores, mayor será el nivel de aprendizaje adquirido por los miembros del grupo.

- **Procesamiento de grupo.** El trabajo efectivo de los grupos de colaboración se encuentra influenciado por su capacidad reflexiva para determinar lo bien que funcionan como equipos. En concreto, para realizar esta determinación, el grupo debe analizarse trabajando junto dentro de la experiencia colaborativa e intentar dar respuesta a dos cuestiones fundamentales. En primer lugar, descubrir qué actuaciones de los participantes fueron útiles durante la experiencia y cuáles no. Y en segundo lugar, tomar decisiones acerca de qué acciones correctivas deben llevarse a cabo para mejorar los resultados en futuras experiencias. El procesamiento del grupo trae consigo varias ventajas. Por un lado, permite establecer y mantener unas buenas relaciones de trabajo entre los miembros del grupo. Por otro, este principio facilita la adquisición de habilidades cooperativas. Además, dicho procesamiento garantiza que los estudiantes reciban una retroalimentación periódica de su nivel de participación en las actividades colaborativas de aprendizaje. De esta manera, los alumnos son inducidos a trabajar reflexivamente a nivel meta–cognitivo. Finalmente, el procesamiento de grupo también constituye una forma de reforzar el comportamiento adecuado de los participantes en la experiencia. Los autores advierten que este principio también puede ser aplicado a nivel de clase por parte del profesor. En este caso, éste debe observar los grupos, analizar los problemas que tienen durante las sesiones de trabajo y proporcionar, a este respecto, cierta retroalimentación a cada uno de ellos.

El método pedagógico Learning Together parte de la premisa de que el tipo de relaciones de interdependencia estructural impuestas por diseño en la experiencia colaborativa determina la manera en que los estudiantes interactúan entre sí, lo que a su vez condiciona los resultados instruccionales obtenidos. Además, la calidad de las relaciones entre los compañeros tiene un fuerte impacto en el desarrollo social y cognitivo de los estudiantes. Los objetivos del método se centran en promover relaciones productivas y actitudes positivas entre alumnos con un marcado carácter heterogéneo entre sí, en transmitir a éstos la habilidad de entender y percibir la situación social en la que se desarrolla las experiencias de aprendizaje en colaboración y en fomentar el pensamiento crítico y creativo y ampliar los niveles de autoestima de los participantes.

Student Team Learning

Student team learning es una familia de métodos pedagógicos genéricos y ampliamente aplicados que tienen por objeto mejorar los resultados de las experiencias de aprendizaje colaborativo [Slavin, 1991 pag. 8-20]. A continuación comentamos someramente los más relevantes:

- **Student Teams Achievement Divisions (STAD).** En este método los estudiantes son organizados en equipos de cuatro o cinco miembros. Cada equipo es un microcosmos de la clase formado por estudiantes heterogéneos con respecto al nivel cognitivo, género, raza, cultura, etc. Cada semana el profesor introduce nuevo material que los equipos deben estudiar de forma conjunta. Aquí no se precisa qué procedimiento especifico deben seguir los estudiantes. Así por ejemplo, éstos pueden realizar lecturas en grupo, resolver problemas por parejas, evaluarse los unos a los otros, etc. Sin embargo, para guiar este estudio, el profesor puede entregar a cada equipo un informe con la relación de contenidos que deben ser aprendidos. Tras varias sesiones de trabajo, los estudiantes son evaluados mediante la realización de cuestionarios individuales que serán corregidos en clase. Finalmente, la puntuación de cada estudiante es transferida por el profesor a una puntuación de equipo que refleja la participación de cada uno de los miembros y posteriormente publicada en clase para estimular la mejora competitiva. La aplicación de este método pedagógico no resulta complicada y sin embargo cambia dramáticamente la dinámica de la clase. En efecto, ahora los estudiantes se ayudan entre sí para adquirir nuevas habilidades, el profesor pasa a ser visto como un recurso que proporciona información de utilidad y además las actividades de aprendizaje son consideradas en su perspectiva social lo que permite fomentar la interactividad y aumentar la cohesión entre los estudiantes .

- **Team Games Tournament (TGT).** Este método de aprendizaje colaborativo utiliza la misma estructura de equipos, formato instruccional y sesiones de trabajo que el anterior y requiere un total de seis semanas para su desarrollo. A lo largo de este tiempo, los estudiantes participan en competiciones semanales que muestran sus habilidades en la materia. En concreto cada estudiante participa en un torneo con otros dos estudiantes de su mismo nivel pero pertenecientes a diferentes equipos. Cada semana la composición social de los torneos se va reajustando para garantizar que la competición siempre se produzca en condiciones de equilibrio cognitivo. Los resultados de los estudiantes en cada competición se convierten en una puntuación para el estudiante que es transferida al grupo y posteriormente publicada semanalmente, como propone el método STAD. En realidad, las diferencias entre este método y el anterior son mínimas. La única novedad aquí consiste en la utilización de juegos académicos para estimular y motivar a los estudiantes.

- **Jigsaw.** En el método de jigsaw original, descrito en [Aronson et al., 1978] los estudiantes se organizan en equipos de 6 miembros y la materia a estudiar se divide en 6 partes iguales, cada una de las cuales se asigna a un estudiante distinto. Primero cada estudiante lee su parte de forma individual. A continuación, cada

miembro se reúne con los estudiantes de los demás equipos a los que se les asignó el mismo material y trabajan conjuntamente durante una sesión. Finalmente, la clase se reorganiza en los equipos originales y cada miembro explica a sus compañeros de equipo el conocimiento experto que ha adquirido. La necesidad de compartir y reestructurar la información segmentada fomenta la colaboración y motiva a los estudiantes a participar. Este método pedagógico ha sido revisado en [Slavin, 1991] produciendo una nueva versión del mismo que se suele referenciar en la literatura como Jigsaw II. El autor propone, en esta nueva versión, que en lugar de que los estudiantes dispongan de una única parte de la información, éstos estudien inicialmente toda la materia y después negocien colaborativamente de qué parte se harán cargo para adquirir conocimiento experto sobre ella. Esto libera al profesor del pesado trabajo de tener que segmentar la información en diferentes partes independientes previamente al desarrollo de la experiencia. Otra mejora es introducida durante la evaluación. Tras la segunda sesión de trabajo en la que los estudiantes vuelven al grupo original para reunir toda la información, se pasa un cuestionario a los miembros y se aplica el sistema utilizado en STAD para fomentar la mejora competitiva.

- **Team Accelerated Instruction (TAI).** El método TAI es una combinación de la instrucción individualizada [Slavin, 1983] y de los tres métodos anteriores aplicado a la enseñanza de las matemáticas. Aquí los estudiantes trabajan en las mismas condiciones que en el método STAD. Sin embargo, mientras que en los métodos anteriores todos los alumnos estudian al mismo ritmo, en TAI se asigna a cada uno de ellos material adicional en función de los resultados obtenidos en un test llamado *de posicionamiento*, para que trabajen individualmente a su propio ritmo. Los miembros del equipo se encargan de comprobar el trabajo de los compañeros confrontando las respuestas de sus test diarios con una hoja de respuestas. La puntuación del equipo es una media de las puntuaciones obtenidas por cada uno de sus miembros, lo que permite a los profesores aplicar estrategias de recompensa para fomentar el aprendizaje. Dado que este es un método de trabajo en grupo orientado al desarrollo del aprendizaje individual, su aplicación resulta apropiada en situaciones de gran asimetría cognitiva entre los estudiantes.

- **Cooperative Integrated Reading and Composition (CIRC).** El método CIRC es una aproximación pedagógica que persigue transmitir habilidades de escritura y comprensión lectora a estudiantes de primaria. En él, los profesores organizan a los estudiantes en grupos de lectura y a su vez cada estudiante es asignado a un equipo formado por un par de compañeros de otros dos grupos de lectura diferentes. Mientras el profesor está trabajando con un grupo de lectura, los estudiantes en los miembros de los otros grupos se dedican a trabajar con sus equipos en una serie de actividades de estimulación cognitiva, tales como la lectura en voz alta, hacer predicciones sobre la evolución de la narración, la síntesis del argumento, la respuesta a preguntas relacionadas con el texto, etc. El trabajo del equipo consiste en escribir borradores y revisar y editar el trabajo de los compañeros lo que permite así adquirir experiencia y habilidades de comprensión lectora.

Todos los métodos descritos en este apartado comparten ciertas características entre sí. En primer lugar, la estructura de los equipos de trabajo es para todos la misma (cuatro o cinco miembros y con un carácter heterogéneo entre ellos). En segundo lugar, la estrategia de evaluación está basada en una puntuación de mejora y recompensa reforzada por la competición. Sin embargo, la secuencia temporal de actividades en los métodos no es la misma. La diferencia entre ellos aparece después de que los estudiantes hayan trabajado en sus respectivos equipos. En STAD y Jigsaw II los participantes realizar un test para mostrar lo que han aprendido, mientras que en TGT los estudiantes compiten con los miembros de otros equipos.

Group Investigation (GI)

Este método pedagógico fue diseñado para su utilización en actividades de aprendizaje cooperativo esencialmente complejas [Sharan & Sharan, 1994]. Sus cuatro principio fundamentales permiten caracterizarlo apropiadamente:

- **Investigación.** De acuerdo a este principio, los estudiantes son organizados en pequeños grupos de investigación, típicamente colectivos de entre tres y cinco individuos, que deben ser orientados por el profesor para la realización de su propia actividad investigadora sobre la materia en estudio. De esta forma, la clase se convierte en una pequeña comunidad de investigadores.

- **Interacción.** Durante la experiencia de investigación puede identificarse una gran cantidad de actividad interactiva entre los miembros del grupo. En efecto, los estudiantes discuten sobre la forma de abordar la investigación, elaboran ideas sobre las opiniones de los compañeros, establecen hipótesis que luego validarán o refutarán, etc. De esta forma, este método ofrece amplias oportunidades para el desarrollo intelectual y social. En este sentido, puede afirmarse que la interpretación de la información a través de la interacción social entre los miembros del grupo es consistente con los postulados constructivistas.

- **Interpretación.** La interpretación hace referencia a la negociación que se produce para reconciliar las perspectivas divergentes de cada uno de los miembros del grupo. Así, este método ofrece la posibilidad de poner en consideración comparativa los diferentes puntos de vista de los estudiantes y de construir, de esta forma, una interpretación cooperativa nueva, de la realidad en estudio.

- **Motivación intrínseca.** Dentro de la clase, los estudiantes disfrutan de un alto nivel de autonomía para determinar qué y cómo quieren aprender. Así, su papel en el aprendizaje es esencialmente activo y tienen una alta responsabilidad y control sobre el conocimiento que ellos adquieren. Este método capitaliza los intereses de los alumnos y les ofrece la posibilidad de dirigir sus propios estudios en grupo, lo cual es una característica esencial para garantizar la motivación

El método de grupos de investigación se articula en seis fases que se desarrollan de forma secuencial a lo largo del tiempo. A continuación entramos a describir cada una de ellas:

- **Fase de identificación.** En esta primera fase, el profesor presenta a la clase un problema complejo, que habitualmente proviene del programa de estudios. Esta presentación es apoyada por diferentes tipos de recursos y medios de soporte, tales como documentos, videos, imágenes, gráficas, estadísticas, etc. Este material persiguen despertar la curiosidad de los estudiantes y estimular el interés por su investigación. A continuación, los estudiantes generan una serie de cuestiones acerca del problema que será objeto de estudio durante la experiencia. Todas estas cuestiones son organizadas en categorías para poder analizarlas en grupos separados. Finalmente, se forman los grupos de investigación.

- **Fase de planificación.** Durante la fase de planificación, los miembros de cada grupo elaboran un plan de acción colaborativo para llevar a cabo la investigación. En concreto, comienzan por escoger un subconjunto de entre todas aquellas cuestiones que fueron identificadas en la fase anterior determinando los recursos que necesitan para abordar dicha investigación. Dependiendo de la complejidad intrínseca de las cuestiones seleccionadas, los estudiantes pueden decidir dividir éstas en tareas más sencillas para tratarlas separadamente.

- **Fase de acción.** Durante la fase de acción se realiza la actividad investigadora dentro de cada grupo. En efecto, los estudiantes arrancan el plan de trabajo definido en la fase anterior y se ponen a trabajar. Este trabajo requiere, entre otras cosas, localizar posibles fuentes de información, extraerla y organizarla, realizar estudios de campo, tomar datos, hacer análisis estadísticos, etc. Los resultados obtenidos deben ser compartidos con los compañeros del grupo para analizar, discutir, interpretar e integrar los mismos y poder preparar un producto final.

- **Fase de finalización.** En la fase de finalización los estudiantes planifican en grupo la presentación de los resultados decidiendo cómo mostrarán a sus compañeros de clase lo que han aprendido. Para ello, deben determinar qué resultados se compartirán con la clase y cómo éstos serán presentados. Esto último puede ser llevado a cabo de muy diversas formas, como por ejemplo mediante la presentación de un informe, la publicación de un poster, la edición de un video, la realización de una sesión de *role playing*, el desarrollo de un experimento, etc.

- **Fase de exposición.** A lo largo de la fase de exposición los grupos de investigación presentan sus resultados al resto de la clase, de acuerdo a la planificación acordada en la fase anterior. En concreto, cada grupo presenta aquellas cuestiones que fueron seleccionadas en la primera fase. De esta manera, los estudiantes descubren las diferentes facetas que aparecían en el problema planteado y aprenden a diferenciar las distintas perspectivas y enfoques posibles para afrontar el estudio de las mismas.

- **Fase de evaluación.** En la fase de evaluación, tanto el profesor como la clase evalúan, para cada grupo, el proyecto de investigación. Dicha evaluación considera el producto final del grupo, el conocimiento de los estudiantes adquirido durante la experiencia de investigación, el desarrollado de dicha experiencia y las situaciones individuales de los estudiantes dentro de la misma.

Como se desprende de lo anterior, este método pedagógico pretende fomentar, no sólo el aprendizaje de la materia sino, en mayor medida, las capacidades de análisis y de síntesis y la búsqueda de situaciones de aplicación de los conocimientos adquiridos. Todo ello se articula a partir de cuatro tipos de actividades básicas: 1) la organización del aula en grupos de trabajo, 2) la realización de actividades de aprendizaje multi–fase para la investigación colaborativa, 3) la comunicación inter–grupal y 4) la comunicación con el profesor que es considerado como un orientador puntual.

Cooperative Learning Structures

En el modelo pedagógico de estructuras cognitivas de [Kagan & Kagan, 1994] los estudiantes se organizan en equipos de un tamaño máximo de cuatro miembros para trabajar durante un largo periodo de tiempo, típicamente un semestre o todo un curso académico. Las técnicas de estratificación utilizadas para formar los equipos son de cuatro tipos. En primer lugar, la estratificación heterogénea, que persigue reunir a estudiantes con diferentes perfiles de acuerdo a algún criterio de caracterización. La estratificación aleatoria, que selecciona a los miembros de cada equipo al azar. La estratificación basada en intereses, que agrupa a los estudiantes en función de ciertas relaciones de afinidad identificadas entre los mismos y, por último, la estratificación basada en el lenguaje, que agrupa a estudiantes de una misma lengua.

En cualquier caso, los equipos deben garantizar cinco principios fundamentales. El principio de familiaridad garantiza que el equipo debe proporcionar cohesión entre sus miembros. El principio de identificación, que los estudiantes de cada equipo se sientan una parte integral del grupo al que pertenecen. El principio de apoyo mutuo, que los miembros del equipo se ayuden entre sí durante el desarrollo de la experiencia colaborativa. El principio de diferenciación, que los miembros sean capaces de reconocer y apreciar las diferencias entre cada uno de ellos. Y por último, el principio de colaboración que asegura que los participantes desarrollarán conjuntamente la actividad. Una vez formados los equipos de trabajo, éstos son implicados en el desarrollo de una colección de actividades diseñadas de acuerdo a unas *estructuras de aprendizaje cooperativo*. Estas estructuras, describen protocolos de interacción y procedimientos de actuación estándar e independientes de dominio que son definidos por el profesor y aprendidos por los estudiantes. De hecho, este método incluye, como parte de los objetivos pedagógicos, el aprendizaje de dichas estructuras. Los autores han identificado más de doscientas estructuras de este tipo que aparecen recurrentemente en las experiencias de aprendizaje colaborativo y cooperativo y que resultan de utilidad para diferentes propósitos [Kagan & Kagan, 1994]. En este trabajo nos limitaremos a describir someramente las más representativas:

- **Three–step Interview.** Esta estructura, utilizada para aumentar la familiaridad entre los miembros del equipo cuando los grupos están recién formados, facilita la identificación de perfiles de personalidad y la asignación de roles de cara al desarrollo de las actividades. En primer lugar, los miembros del grupo son organizados en parejas. Primero uno de ellos entrevista al otro haciéndole diferentes preguntas y tomando notas sobre sus respuestas. En una segunda fase, los papeles se

intercambia de modo que el alumno entrevistado para a ser el entrevistador y recíprocamente. Finalmente, los cuatro miembros del grupo se vuelven a reunir y cada uno de ellos presenta, por turnos, al compañero que entrevistó.

- **Roundtable.** Esta estructura constituye un protocolo de tormenta de ideas para generar un gran número de respuestas a una o varias cuestiones. Típicamente el profesor arranca esta estructura presentando una cuestión y proporcionando a cada equipo material para escribir. El primer estudiante escribe una respuesta para la cuestión y la dice en voz alta, y a continuación, pasa el papel al estudiante de su derecha que procede de manera similar. El procedimiento continua circularmente entre los miembros del grupo hasta que el tiempo se acaba.

- **Focused Listing.** Esta estructura, con similares objetivos al Roundtable, presenta la diferencia de que aquí el propósito es encontrar definiciones o descripciones en torno a un concepto. Una vez que los estudiantes hayan realizado esta actividad los resultados pueden utilizarse para comenzar una discusión de clase. El procedimiento comienza cuando el profesor pide a los estudiantes que escriban entre cinco y siete palabras describiendo un concepto. Después los estudiantes deciden en grupo cuáles de todas las definiciones resultan más apropiada.

- **Structured Problem–solving.** En esta estructura cooperativa el profesor presenta un problema a los estudiantes. Después se asigna un número a cada miembro del equipo que comienza a trabajar sobre el problema. Finalmente el profesor escoge un número al azar y los miembros de los equipos identificados con ese número deben responder a las preguntas de éste sin ayuda de sus compañeros.

- **Paired Annotations.** En esta estructura los estudiantes son organizados en parejas. Cada pareja se encarga de revisar un artículo, texto o área de contenido y utilizan un diario de doble entrada para que cada uno de ellos pueda anotar sus impresiones sobre el tema. Después, los estudiantes revisan sus anotaciones y discuten las discrepancias encontradas entre las perspectivas de cada uno de ellos. Finamente, los estudiantes elaboran un informe que resume la nueva perspectiva fruto del consenso.

- **Structured Learning Team Group Roles.** Comúnmente cuando se desarrollan actividades de aprendizaje colaborativo de una elevada complejidad resulta conveniente definir diferentes tipos de roles que se responsabilicen de distintos aspectos de la misma. En concreto los autores proponen en esta estructura la asignación de cinco tipos de roles de uso frecuente: el líder, que dirige el trabajo del grupo, el anotador que se encarga de mantener y custodiar los documentos, el portavoz, que es un representante del grupo a nivel de clase, el monitor, que regula el trabajo del equipo y cronometra las actividades y el comodín, que actúa de asistente.

- **Send A Problem.** Esta estructura puede ser utilizada para que los equipos de la clase discutan y revisen posibles soluciones a un problema de naturaleza compleja. De acuerdo a la misma, cada grupo empieza describiendo un problema y redactando una posible solución para el mismo. Transcurrido un tiempo de entre

tres y cinco minutos, cada equipo entrega el problema y la solución al equipo de su derecha, circularmente, momento en el cual comienza una nueva sesión de tormenta de ideas para criticar la solución del problema recibido y aportar posibles mejoras o soluciones alternativas. Cuando todos los equipos han revisado todos los problemas se comienza una discusión en clase para determinar entre todos cuál es la mejor solución para cada problema.

- **Value Line.** Esta estructura describe una estrategia para llevar a cabo la formación de equipos de trabajo con carácter heterogéneo. El profesor idea un test con pregunta de tipo likert sobre un conjunto de cuestiones. Es decir, un cuestionario con preguntas cuyas posibles respuestas se organizan de acuerdo a un rango de valores discretos comprendidos, típicamente, entre 1 (muy malo) y 10 (excelente). Después entrega este test a los estudiantes para que lo rellenen. Cuando los estudiantes ya han respondido, el profesor organiza a éstos en un ranking en función de sus valoraciones. Para formar cada grupo de cuatro miembros, el profesor sólo debe coger a un estudiante de cada extremo del ranking y a otros dos del centro.

- **Team Expectations.** Cuando comienza una experiencia de aprendizaje es frecuente que los estudiantes no crean disponer de un nivel lo suficientemente alto como para participar y ponen en duda el valor del carácter colaborativo de la misma. Para dar solución a este problema, esta estructura propone a los alumnos rellenar un cuestionario para reflejar el comportamiento esperado de cada uno de los miembros del grupo, de cada pareja y del propio grupo como unidad durante la experiencia. Finalizada ésta, los estudiantes utilizan dicho cuestionario para comprobar si el comportamiento real se correspondió con el esperado.

- **Double Entry Journal.** Esta estructura permite a los estudiantes preparar el material de estudio de una clase. El profesor les entrega un texto para que lo lean individualmente y extraigan las cuestiones principales de éste respondiendo a las mismas en un cuestionario. Cuando los estudiantes terminan, se reúnen por parejas para discutir y comentar los resultados obtenidos.

- **Guided Reciprocal Peer Questioning.** El objetivo de esta estructura es provocar una discusión acerca de un determinado contenido. El profesor hace una introducción breve de unos quince minutos sobre un texto y proporciona una serie de cuestiones a los estudiantes que giran en torno al mismo. Éstos trabajan individualmente para escribir una reflexión – no necesariamente una respuesta – sobre estas cuestiones basándose en el material proporcionado. Finalmente los estudiantes se organizan en grupos y discuten las cuestiones que consideran mas interesantes. Estas preguntas suelen tener un pie de la forma: ¿qué pasaría sí...?, ¿Cómo afectaría esto a...?, ¿Por qué parece que...?, etc.

El método pedagógico estructural genera actividades de aprendizaje diseñadas de tal forma que se garantiza la interacción simultanea entre los estudiantes, su participación igualitaria, una apropiada interdependencia positiva y una responsabilidad individual de éstos para con la actividad en la que están involucrados. Además, como ocurría en GI, la organización de la clase ha cambiado relegando el papel del profesor

a un segundo plano, encargado de proporcionar ahora los recursos apropiados para el desarrollo de la experiencia y controlar el comportamiento de los equipos a lo largo de ésta. En cuanto a los objetivos pedagógicos perseguidos se suman a los propio de adquisición de procedimientos de actuación que mencionamos antes, la integración de destrezas de carácter social.

Complex Instruction (CI)

Complex Instruction es un método pedagógico que tiene por objetivo fomentar la adquisición de habilidades de razonamiento de orden superior en clases de estudiantes con niveles cognitivos e intelectuales diferentes [Cohen, 1994]. Este método consiste en organizar a los alumnos en grupos de cuatro miembros y asignar a cada uno de ellos un papel diferente: el capitán del equipo, el facilitador, el administrador de recursos y el reportero. El profesor asigna a los grupos un problema complejo para que resuelvan juntos y pide a los alumnos que se aseguren de que todos sus compañeros también aprenderán. Así, si un miembro del grupo tiene dificultades con el problema, es responsabilidad de los demás ayudarle. Al final de la lección, uno de los miembros de cada grupo será seleccionado para presentar sus conclusiones a la clase. Debido a que los alumnos no saben de antemano qué miembro del grupo hará la presentación, éstos deben asegurarse de que cualquiera será capaz de hacerla llegado el momento. La autora de este método destaca tres principios fundamentales para caracterizar el desarrollo del mismo. A continuación describimos cada uno de ellos:

- **Definición de un curriculum con múltiples habilidades.** El programa de estudios de los estudiantes debe ser rediseñado de manera que se fomente la adquisición de habilidades de razonamiento de orden superior a través de actividades formativas desarrolladas en grupos. Estas actividades deben ser poco estructuradas y pobremente definidas de manera que requieran una colaboración interdependiente para su resolución por parte de estudiantes con diferentes habilidades intelectuales y niveles académicos y que permitan la contribución significativa de todos los miembros del grupo.

- **Uso de estrategias instruccionales.** Los profesores deben orientar a los estudiantes para establecer ciertas normas cooperativas y definir roles en aras a gestionar el desarrollo de las actividades. El profesor observa dicho desarrollo y proporciona, puntualmente, retroalimentación sobre el mismo a los estudiantes.

- **Tratamiento de problemas de estatus.** El profesor debe aprender a identificar los problemas potenciales que surjan en el seno del desarrollo de las actividades colaborativas motivados por las diferencias entre los estudiantes y ayudarles a resolver estos problemas cuando aparezcan.

Para terminar con la descripción de los métodos pedagógicos que se dirigen a inducir situaciones de aprendizaje presentamos, en la tabla 2.2, un resumen adaptado de [Tan el al., 2006]. En ella, se esquematizan, para cada método analizado a lo largo de esta sección, los objetivos pedagógicos en su dimensión social y cognitiva, los pasos del proceso instruccional y las características del tipo de actividades utilizadas.

	Learning Together	Jigsaw	Student Team Learning	Group Investigation	Structural Approach	Complex Instruction
Objetivos cognitivos	Mejorar los resultados académicos de los estudiantes	Aprender un tópico especifico Monitorizar trabajo en parejas Mejorar los resultados académicos de los estudiantes	Adquirir habilidades académicas sobre para todos los miembros del grupo. Cada uno progresa de manera individual.	Estimular la curiosidad de los estudiantes Adquirir habilidades de razonamiento de orden superior Adquirir un método científico de trabajo	Compartir información, expresar ideas y opiniones y mejorar el pensamiento crítico y los resultados académicos	Dar acceso a todos los estudiantes al desarrollo de actividades complejas Mejorar los resultados académicos
Objetivos sociales	Crear relaciones interdependencia positivas y ayuda mutua Fomentar la responsabilidad individual y grupal	Aprender un tópico especifico Monitorizar trabajo en parejas Mejorar los resultados académicos de los estudiantes	Incrementar la motivación de los estudiantes a través de la ayuda mutual. Cada estudiante tiene iguales posibilidades y es responsable de los demás	Desarrollar la responsabilidad de los estudiantes en el aprendizaje Promover la autonomía en la relación de actividad	Promover la interacción simultanea, la participación equilibrada y la cohesión social en el grupo	Fomentar la igualdad en el estatus social y en las oportunidades de aprendizaje para obtener mejores relaciones sociales
Proceso instruccional	1. El profesor prepara una unidad de estudio 2. El profesor divide a los estudiantes en grupos 3. Los miembros se ayudan a hacer la actividad	1. Cada miembro se reúne con los estudiantes de otros grupos para formar grupos expertos 2. Los grupos se vuelven a reunir y cada experto enseña a sus compañeros	1. El profesor presenta la lección 2. Se realiza el trabajo en grupo 3. Se realizan test y cuestionarios o competiciones	1. El profesor presenta un tópico 2. Los estudiantes trabajan de acuerdo a un plan 3. Se presentan los resultados 4. se evalúa en clase el trabajo	1. El profesor selecciona una estructura apropiada para los objetivos pedagógicos 2. Los estudiantes trabajan para completar la actividad	1. El profesor presenta una actividad poco estructurada y mal definida 2. Los estudiantes definen roles 3. Los estudiantes desarrollan en grupo la actividad
Características de la actividad	Las actividades, relativamente sencillas, son definidas por profesores (elaboración de resúmenes, escritura de informes, etc.)	Las actividad son idénticas para cada grupo Cada estudiante se especializa en adquirir conocimiento sobre una parte de la actividad	Las actividades son relativamente cortas y simples Los grupos se centran en las actividades presentadas por el profesor	Las actividades son complejas, poco estructuradas y presentan múltiples facetas. Se enfatiza el desarrollo autónomo y la adquisición de un método de trabajo	Algunas actividades son cortas y simples mientras que otras son más largas y complejas	Las actividades son poco estructuradas y mal definidas y sin una única solución El desarrollo requiere la aplicación de habilidades cognitivas de orden superior

Tabla 2.2. Resumen de los métodos pedagógicos de aprendizaje colaborativo.

2.5. Soporte computacional al aprendizaje colaborativo

El último de los aspectos a los que debemos prestar atención es el relativo a las soluciones tecnológicas que existen en la actualidad para dar soporte computacional al diseño y desarrollo de experiencias de aprendizaje colaborativo. A lo largo de esta sección pretendemos alinear la discusión con cada uno de los subsistemas que forman parte de la arquitectura general de la plataforma Pelican que presentamos en este trabajo, para que sirva de marco comparativo (consulte el capítulo 3 para detallas sobre la arquitectura de Pelican). Por ello dividiremos esta sección en cuatro subsecciones donde discutiremos el soporte a la estratificación social, a la colaboración, a la integración de herramientas externas y a la adaptación del entorno.

2.5.1. Soporte a la estratificación social

En la dimensión social del aprendizaje colaborativo, la problemática desde el punto de vista tecnológico, es articular mecanismos para definir y dar soporte a colectivos de usuarios organizados en grupos. La primera preocupación se asocia a la fase de diseño mientras que la segunda aparece vinculada a la fase de desarrollo. A continuación abordamos en detalle cada una de ellas.

2.5.1.1. Fase de diseño

Muchas de las soluciones de diseño social encontradas en la literatura son modelos o lenguajes de especificación que se encuentran formando parte del diseño de escenarios colaborativos [IMS–LD] [Miao et al., 2005] por ello serán descritos en la siguiente subsección. No obstante en este punto si existen dos especificaciones estándares focalizadas en la dimensión social que merece la pena destacar.

IMS Enterprise Information Model

[IMS–EM] es una especificación estándar que proporciona un modelo de información para representar estructuras organizacionales formadas por grupos y personas. El modelo es pretendidamente sencillo. Según la especificación, una estructura social puede representarse como una colección de grupos potencialmente relacionados entre sí. Cada grupo constituyen una estructura agregativa jerárquica formada por otros grupos y personas. La prescripción proporciona un modelo de usuario para las personas formado por información de identificación y localización geográfica básicamente mientras que el modelo de grupo se articula en términos del tipo de grupo, la organización a la que pertenece o el tiempo que permanece activo, entre otros. El mayor inconveniente de esta propuesta radica precisamente en su sencillez. Modelos como estos pueden resultar satisfactorios para representar determinados tipos de colectivos sociales. Sin embargo imposibilitan la distinción de tipos de usuarios y la definición estructural de los tipos de grupos en términos de éstos.

IMS Learning Information Package

La propuesta descrita en [IMS–LIP] ofrece un modelo de usuario más rico que el anterior. El propósito de la misma es definir un portafolio de aprendizaje para los usuarios de los sistemas instruccionales que contenga información sobre el estudiante y sus conocimientos a modo de curriculum. En concreto, el modelo propuesto está organizado en varias categorías para almacenar la información de identificación del usuario (datos biográficos y geográficos), objetivos y aspiraciones del mismo, cualificaciones y certificaciones obtenidas, actividades relacionadas con el aprendizaje en las que ha participado, datos curriculares, intereses y aficiones, competencias y habilidades, datos de afiliación a organizaciones profesionales e información de accesibilidad (idiomas, estilos de aprendizaje etc.). Este modelo es considerablemente más rico que el anterior e incluso proporciona mecanismos de extensión. No obstante, tal como se verá en el capítulo 3, nuestra propuesta es considerablemente más sencilla ya que manejar este volumen de datos es complejo y cae fuera de nuestros objetivos.

2.5.1.2. Fase de desarrollo

En la fase de desarrollo, los entornos virtuales de aprendizaje constituyen, en la actualidad, las herramientas software más apropiadas para dar soporte a las experiencias de aprendizaje colaborativo. Sin embargo, no todos ellos se encuentran centrados en este paradigma. La mayoría de estos entornos conciben el soporte a la colaboración como una característica de valor añadido marginal representada por la presencia de medios de interacción en el entorno. Nosotros en este sentido entendemos que un entorno virtual de aprendizaje orientado a la colaboración puro debe estar centrado en la actividad, orientado al trabajo colaborativo y dirigido por la adaptación continua de los escenarios, como se discutirá en los siguientes capítulos. En nuestro estudio sobre los entornos de aprendizaje no hemos encontrado ningún sistema que se ajuste exactamente a esta caracterización, motivo por el cual se ha desarrollado Pelican. Antes de pasar a caracterizar los entornos estudiados en este sentido, resulta conveniente clasificarlos en función del enfoque que hacen del soporte instruccional:

- **Sistemas de gestión del aprendizaje.** Los sistemas de gestión del aprendizaje (LMS en inglés) son entornos diseñados para dar soporte a los procesos instruccionales de un determinado contexto educativo, típicamente una universidad, un instituto o un colegio. A través de ellos, los administradores pueden gestionar el colectivo de usuarios de la organización, controlar el seguimiento de cada estudiante o planificar e informar de eventos y sucesos relacionados con el aprendizaje. Los entornos de este tipo que dan soporte a la colaboración lo hacen proporcionando consciencia social y un conjunto de herramientas de soporte a la interacción estándar. [Alf], basado en [dotLRN], es un ejemplo de este tipo de sistemas.

- **Sistemas de gestión de contenidos de aprendizaje.** Los sistemas de gestión de contenidos de aprendizaje son sistemas LMS que ademas incorporan una serie de facilidades para la edición, publicación y entrega de contenidos pedagógicos digitales en línea típicamente en forma de objetos de aprendizaje [Willey, 2000] y de acuerdo al estándar [IMS–CP]. De hecho, muchos de estos sistemas se implementan como una extensión de sistemas LMS. De ahí una frecuente confusión entre éstos y los primeros. Los entornos de este tipo con facilidades colaborativas son aquellos que presentan las mismas características de soporte a la colaboración que los sistemas LMS. Un típico ejemplo de este tipo de entornos es [Moodle].

- **Sistemas de gestión de cursos de aprendizaje.** Los sistemas de gestión de cursos de aprendizaje ofrecen capacidades para la gestión de los aspectos educativos de un contexto y dan soporte a la edición y entrega de contenidos pedagógicos en línea. Pero adicionalmente, organizan la estructura de los mismos, y los servicios que ofrecen a profesores y alumnos a partir de una estructura de cursos. Es decir, su uso se encuentra orientado exclusivamente a contextos de educación formal estructurados en forma de años académicos, cursos, semestres, asignaturas y prerrequisitos para cursar las mismas. La orientación a la colaboración en este tipo de sistemas también suele limitarse a proporcionar una serie de servicios de interacción. [Web–CT] y [Blackboard] son los dos ejemplos más representativos.

El marco comparativo que utilizaremos para hacer un análisis de los diferentes entornos de aprendizaje estudiados con alguna propiedad colaborativa se centra en una serie de factores que quedan organizados conceptualmente en dos categorías: el soporte a los perfiles de usuario y el soporte a la gestión de grupos. A continuación describimos brevemente estos factores:

- **Soporte a perfiles de usuario.** Un primer aspecto a considerar al abordar el estudio de estos sistemas es el relativo a la capacidad de los mismos para trabajar con diferentes perfiles de usuarios que jueguen un papel diferente dentro de la estructura social de la comunidad virtual. A este respecto, cabe formular las siguientes cuestiones: (P1) ¿Es el sistema capaz de reconocer diferentes tipos de usuarios? (P2) en tal caso, ¿pueden ser estos perfiles definidos por el administrador de la plataforma o están establecidos ad-hoc? (P3) si es así, ¿se puede asignar una familia de permisos sobre los recursos de la plataforma para cada perfil definido? Y respecto al uso de los perfiles, (P4) ¿puede un usuario tener asignados, aunque no simultáneamente, varios perfiles dentro de la comunidad?

- **Soporte a grupos.** En lo relativo a la gestión de grupos cabe discutir (G1) si la comunidad permite organizar a los estudiantes en grupos, (G2) si es posible definir distintos tipos de grupos caracterizados en términos de los tipos de usuario que agrega con el fin de identificar colectivos con diferentes responsabilidades dentro de la comunidad, (G3) si un usuario puede pertenecer simultáneamente a diferentes grupos, (G4) si puede un usuario tener un perfil diferente en cada grupo y, finalmente, (G5) si puede un mismo usuario tener diferentes perfiles dentro de un mismo grupo.

VLE	Soporte a Perfil de Usuario				Soporte a Grupos				
	P1	P2	P3	P4	G1	G2	G3	G4	G5
Web CT	*Sí*	*Fijo*	*No*	*No*	*Sí*	*No*	*Sí*	*No*	*No*
Blackboard	*Sí*	*Fijo*	*No*	*Sí*	*Sí*	*No*	*Sí*	*No*	*No*
Moodle	*Sí*	*Flexible*	*Sí*	*Sí*	*Sí*	*Sí*	*Sí*	*No*	*No*
Sakai	*Sí*	*Flexible*	*Sí*	*No*	*No*	*Sí*	*No*	*Sí*	*Sí*
aLF	*Sí*	*Flexible*	*Sí*	*No*	*Sí*	*Sí*	*Sí*	*No*	*No*

Tabla 2.3. Relación comparativa de los entornos de aprendizaje.

La tabla 2.3, resume los resultados de nuestro estudio. Como se ve, todos los sistemas distinguen tipos de usuarios. Sin embargo en Web–CT y Blackboard éstos están establecidos de forma estática. En aLF y Sakai no es sencillo asignar diferentes perfiles a un mismo usuario. En cuanto a la gestión de grupos parece que el soporte más completo lo proporcionan Moodle y aLF aunque en general ninguno soporta excepto Sakai cambio dinámico de perfiles entre grupos.

2.5.2. Soporte a la colaboración

El soporte al trabajo colaborativo que requiere este paradigma de aprendizaje consiste, por un largo, en proporcionar mecanismos para la especificación formal y computable de escenarios de aprendizaje colaborativo y, por otro en ofrecer medios de soporte a la interacción colaborativa. Los primeros se vinculan a la fase de diseño de la experiencia mientras que los segundos lo hacen a la fase de desarrollo. A continuación abordamos las propuestas en sendas fases:

2.5.2.1. Fase de diseño

El diseño de las experiencias de aprendizaje colaborativo es un labor extremadamente complicada si tenemos en cuenta que debe ser realizado para garantizar la existencia de ciertos tipos de interacción que se suponen pedagógicamente beneficiosos para el estudiantes. Algunos autores [Dillenbourg & Jermann, 2004] reconocen en este sentido que pueden distinguirse dos aproximaciones complementarias:

- **Diseño instruccional convencional.** El diseño convencional se realiza a través del ajuste sistemático de una serie de parámetros tales como el tamaño del grupo, los tipos de interacción permitidos, la naturaleza y complejidad de las actividades, el tiempo de las sesiones, las estrategias de evaluación, etc. Lamentablemente todos estos factores se encuentran estrechamente relacionados entre sí y resulta complejo encontrar las condiciones óptimas para que se produzcan los mecanismos de aprendizaje perseguidos.

- **Diseño basado en scripts.** El diseño de escenarios basado en scripts de colaboración [O'Donnell & Dansereau, 1992] propone realizar una definición de un guión de actuación para los estudiantes – el script – ideado de tal forma que fomente la existencia de conflictos socio–cognitivos entre los miembros del grupo y la elaboración de explicaciones, justificaciones y conceptualizaciones compartidas y consensuadas del problema en estudio. Este enfoque se basa en el principio de que la colaboración puede ser influenciada anticipadamente estructurando el proceso colaborativo con el fin de favorecer el surgimiento de interacciones productivas o retroactivamente, regulando las interacciones como hacen profesores y monitores en el enfoque convencional.

Esta segunda opción es la que goza de más aceptación en la actualidad. [Dillenbourg, 2002] y [Dillenbourg & Jermann, 2004] afirman que el diseño de script de colaboración debe hacerse prestando especial atención a cuatro ejes dimensionales:

- **Granularidad.** Los scripts de colaboración pueden variar en las escala de tiempo en la que se aplican, yendo desde 20 minutos a todo un curso académico y en el tamaño del los grupos implicados en cada actividad. En scripts de grano grueso, cada fase es una actividad pedagógica significativa que consume al menos unas cuantas semanas. En scripts de grano fino, por el contrario, las fases pueden desarrollarse de forma instantánea en pocos segundos simplemente por la interacción de un par de estudiantes. Esto permite hablar de dos tipos de scripts:

  - *Macro–scripts.* Un macro–script es un script de grano grueso que sigue una determinada orientación pedagógica y se centra en la orquestación de las actividades que constituyen la experiencia de aprendizaje colaborativo.
  - *Micro–scripts.* Los micro–scripts, por el contrario, son scripts de grano fino que persiguen disparar los mecanismos propios de las teorías psicológicas que fueron discutidas en la sección 2.3 tales como la creación de conflictos o la elicitación de la información.

- **Grado de coerción.** Los scripts de colaboración también varían según el grado de libertad de actuación que conceden a los estudiantes implicados. Generalmente los scripts de alta granularidad suelen ser más coercitivos. Sin embargo, para un mismo nivel de granularidad, algunos scripts pueden forzar a los estudiantes a realizar algunas actividades mientras que otros simplemente inducen y orientan a su desarrollo. En este sentido, los autores distinguen cinco categorías de scripts:
  - *Scripts de inducción.* Los interfaces de comunicación de las herramientas de interacción colaborativa propias de los VLE suelen inducir, de manera automática, determinados patrones de interacción que transfieren la forma en que el diseñador espera que se produzca la interacción. Este bajo nivel coercitivo es elegante pero no influye significativamente en el proceso colaborativo.
  - *Scripts de instrucción.* Cuando el profesor da instrucciones acerca de cómo deben comportarse los estudiantes, está comunicándoles un script de instrucción. El nivel de coerción es más alto que en el caso anterior ya que las expectativas del profesor se hacen explícitas aunque éstas puedan ser malinterpretadas, incorrectamente aplicadas, olvidadas o completamente ignoradas.
  - *Scripts de entrenamiento.* A veces es conveniente entrenar a los estudiantes en el uso de un script antes de su aplicación en las condiciones reales. El nivel coercitivo aquí es mayor que en los dos casos anteriores ya que el profesor puede controlar la aplicación que hacen los estudiantes de las reglas del script.
  - *Scripts de prompt.* En los scripts de prompt, o scripts de pregunta clave, el sistema va proporcionando pistas a los estudiantes para orientarles en la realización de cada actividad colaborativa [Weinberger et al., 2002]. Estos scripts suelen además condicionar la responsabilidad colaborativa que adquiere cada participante con lo que el nivel de coerción crece.
  - *Scripts follow–me.* En este tipo de scripts los estudiantes interactúan con un entorno VLE que les fuerza a seguir y realizar la colección de fases y actividades prescritas en el script de colaboración.

- **Nivel de apropiación.** Algunos scripts de colaboración tienen por objeto describir un protocolo de actuación para desarrollar una experiencia de aprendizaje como si fueran las reglas de un juego. En otros casos por el contrario, el script conforma un procedimiento estándar que debe ser interiorizado por los estudiantes y forma. En este sentido [Kollar et al., 2005] distinguen entre dos tipos de scripts:

  - *Scripts externos.* Los scripts externos son scripts de colaboración que no son, al menos inicialmente, representados internamente en el sistema cognitivo de los estudiantes, aunque su propósito pedagógico sea que sean asimilados por un mecanismo de interiorización.

  - *Scripts internos.* Los scripts internos por el contrario, son guías de actuación personal de los propios estudiantes basadas en su propia experiencia y conocimientos que son utilizados para entender una situación particular presentada durante el desarrollo de la experiencia.

- **Grado de generalidad.** El grado de generalidad hace referencia a la versatilidad del scripts para aplicarse a diferentes áreas de conocimiento y dominios de aplicación. Los scripts centrados en un dominio, como los de matemáticas, no son de utilidad fuera del mismo mientras que procedimientos estándar del estilo Jigsaw son aplicables a cualquier situación pedagógica independientemente del la disciplina estudiada.

En el punto anterior se ha puesto de ejemplo de script de colaboración el procedimiento Jigsaw, que en realidad es un método pedagógico que fue descrito en la sección 2.4. Esto no es tan extraño si tenemos en cuenta que los scripts pueden concebirse como plantillas parametrizables constituidas esencialmente por una secuencia de fases caracterizadas por diferentes parámetros: la actividad a realizar, la composición del grupo, la asignación de responsabilidades colaborativas, el modo de interacción y la duración temporal [Dillenbourg, 2002].

[Van de Velde et al., 2004] ha clasificado, en este sentido, los scripts de colaboración en ocho familias distintas en función del tipo de interacción que pretenden fomentar y de los fundamentos en los que se apoyan:

- **Jigsaw.** Los scripts de Jigsaw proponen estrategias de división de la información entre los diferentes miembros de un grupo. Su fundamento reside en la idea de que si ningún estudiante dispone de todos los elementos necesarios para realizar una actividad, es de esperar que se disparen mecanismos de interacción colaborativa para alcanzar, de manera conjunta, los objetivos de la misma. El diseñador debe procurar, al definir scripts de este tipo, que la información adquirida por cada una de las partes resulte incomprensible sin la referencia a las demás. Este objetivo puede ser alcanzado dividiendo el grupo en pares de trabajo [Hoppe & Ploetzner, 1999], proporcionando a cada miembro información complementaria e incluso a través de la definición de roles de especialización [Dillenbourg, 2002].

- **Conflicto.** Este tipo de scripts pretenden fomentar estados de argumentación formando pares de estudiantes con puntos de vista diferentes de cara a la actividad planteada [Jermann & Dillenbourg, 2002]. Para asegurar la colaboración argumentativa la actividad debe presentar una evidencia conflictiva o debe solicitar a cada miembro jugar un rol antagonista dentro del escenario propuesto. Estos scripts se apoyan bajo la hipótesis de que la resolución del conflicto obligará a los estudiantes a reorganizar sus propias estructuras cognitivas.

- **Enseñanza recíproca.** Los scripts de enseñanza reciproca [McKeachie et al., 1986] se articulan en base a dos roles opuestos cuya asignación a los dos miembros del grupo va alternativamente cambiando en varias fases de ejecución. Dentro del escenario, el primer rol realiza una actividad, mientras que el segundo enseña al primero y regula su trabajo.

- **Regulación.** El aprendizaje por auto–regulación es definido como un proceso de construcción activo a través del cual los estudiantes establecen sus propios objetivos e intentan planificar, monitorizar, regular y controlar su aprendizaje, motivación y comportamiento. Aunque los procesos de auto–regulación han venido históricamente vinculados a los paradigmas de aprendizaje individual, existe, en la actualidad, un creciente interés en considerar las dimensiones sociales de la auto–regulación. Aparecen así nuevos términos como regulación social, regulación compartida o co–regulación.

- **Investigación.** Referenciado comúnmente como aprendizaje por investigación, pensamiento crítico o reflexivo, método inductivo, método científico o aprendizaje conceptual, esta aproximación da lugar a scripts que dirigen la colaboración planteando una pregunta abierta que los estudiantes deben responder.

- **Competición.** Los scripts basados en competición [Slavin, 1990] [Slavin, 1980] organizan grupos de trabajo dentro del colectivo de estudiantes y les ponen a competir entre ellos. Dentro del grupo, cada miembro es responsable de su propio aprendizaje y del de sus compañeros. Esta familia de scripts utiliza los conceptos de objetivos y éxito de equipo definidos en términos del aprendizaje individual y grupal.

- **Negociación.** Los scripts de negociación establecen una situación donde se establece entre los estudiantes participantes un espacio de malentendidos y desacuerdos. Dentro de estas situaciones, los estudiantes deben explicar y justificar sus posturas y negociar con el resto de los participantes una conceptualización compartida.

- **SWISH (Split Where Interaction Should happen).** La idea fundamental en este tipo de scripts es que el aprendizaje resulta de las interacciones sociales que ocurren al intentar construir una conceptualización compartida de un problema a pesar del hecho de que ésta sea distribuida. Dicha distribución determina la naturaleza de las interacciones, y éstas son la base para llevar a cabo una división del trabajo. Por tanto, esta división puede ser diseñada para disparar aquellas interacciones que se desee fomentar.

Una vez establecida esta distinción conceptual entre los dos tipos de aproximaciones vigentes para llevar a cabo el diseño de experiencias de aprendizaje colaborativo, estamos en disposición de pasar a presentar las aportaciones realizadas por la comunidad científica para dar soporte a este tipo de tareas de diseño. A lo largo de los siguientes apartado discutiremos las especificaciones, estándares, extensiones, soluciones tecnológicas y sistemas más relevantes en este sentido.

IMS Learning Design

La especificación estándar [IMS–LD] permite definir escenarios de aprendizaje para desarrollar experiencias bajo cualquier enfoque pedagógico. Su modelo semántico parte del concepto de *método* para describir un escenario. Éste está asociado a unos *objetivos pedagógicos* y a un conjunto de *prerrequisitos* e incorpora una colección de *actuaciones* que son desarrolladas en paralelo. A su vez, las actuaciones están descritas mediante una secuencia ordenada de *actos*. Estos elementos permiten definir un flujo de instrucción donde los participantes intervienen representando alguno de los dos roles básicos definidos: *Learner* y *Staff*, que pueden extenderse mediante operaciones de especialización y composición. En concreto los estudiantes, llevan a cabo una colección de actividades organizadas de forma jerárquica dentro de un entorno que proporciona recursos en forma de objetos de aprendizaje y servicios de interacción. El desarrollo de actividades provoca, por un lado, la generación de productos que se usarán potencialmente dentro del entorno y por otro la propagación de notificaciones para intervenir en el acto en curso. La especificación IMS – LD define tres niveles de compromiso al que pueden adscribirse los diferentes sistemas que la implementen. El nivel A, incorpora los elementos centrales del meta–lenguaje, el nivel B permite, adicionalmente, realizar descripciones de escenarios haciendo uso de condiciones y propiedades genéricas y el nivel C articula la posibilidad de realizar notificaciones asíncronas y de añadir dinámicamente nuevas actividades. Actualmente todos los sistemas implementados se encuentran en los dos primeros niveles.

[Miao et al., 2005] destacan, no obstante, cinco problemas que surgen a la hora de tratar de definir scripts de colaboración para que sean interpretados por los entornos de aprendizaje. El primero de ellos aparece al tratar de describir una estructura social de grupos sobre la que desarrollar experiencias colaborativas. En efecto, IMS–LD permite definir múltiples roles que pueden ser desempeñados por una o varias personas. No obstante, este mecanismo resulta insuficiente ya que no permite definir situaciones en las que varios grupos trabajen en paralelo, su estructura interna se componga de otros grupos o puedan darse cambios dinámicos en la composición de los mismos durante el desarrollo de la experiencia. El segundo gran problema hace referencia a la imposibilidad de definir artefactos que sean compartidos por un colectivo de usuarios. En IMS los artefactos pueden ser modelados mediante una propiedad asociada a la persona o rol que los crea. Sin embargo, en los escenarios colaborativos, éstos deben asociarse a todos los miembros del grupo. Además, los artefactos son elementos de intermediación que surgen a lo largo de un flujo de trabajo instruccional y que facilitan la interacción indirecta entre los miembros del grupo. Esta idea es difícilmente trasladable a una especificación IMS–LD. En tercer lugar, los escenarios colaborativos precisan prestar atención a los aspectos dinámicos de la experiencia que deban alterar puntualmente el entorno. IMS–LD proporciona dos categorías de operaciones en esta línea: operaciones de acceso y operaciones de modificación. Esto resulta claramente insuficiente para hacer frente a la gran cantidad de tipos de alteraciones que pueden producirse durante el desarrollo de una experiencia colaborativa. El cuarto inconveniente se refiere a la imposibilidad de definir un modelo estructural

de trabajo que no sea meramente lineal. Este es un aspecto que queda solventado en la extensión de [Caeiro et al., 2003] que describimos más adelante. Finalmente, los autores señalan la necesidad de cambiar el enfoque de IMS–LD hacia un modelo centrado en la actividad colaborativa que permita describir quién, cómo y cuándo se produce la interacción.

IMS Learning Design, Extensión de [Caeiro et al., 2003]

La propuesta IMS–LD constituye un mecanismo de especificación formal de procesos de aprendizaje que pretende ser pedagógicamente neutral por cuanto da soporte a cualquier paradigma de aprendizaje. Esto es así en tanto que reúne en su modelo de información todos los elementos comunes que aparecen en cada uno de los paradigmas existentes. No obstante, [Caeiro et al., 2003] señalan algunas deficiencias del modelo y proponen una forma de extenderlo para suplirlas. En primer lugar, la especificación IMS–LD da una mala cobertura a la descripción de procesos colaborativos. Para ello, los autores utilizan la entidad *Role-Part*, que desarrolla la participación en tareas individuales, para relacionarla con nuevas entidades y otras ya existentes en el modelo. En concreto, dicha entidad estará asociada al *acto* para indicar cuándo se participa en un proceso colaborativo, a la *actividad* para indicar cuál es el propósito de la intervención, al *entorno,* para indicar en cuál de todos los contextos sociales existentes se está participando, al *Role* para indicar el perfil de usuario que se encarna, al *resultado*, para asociar la actuación individual con el producto generado en el desarrollo colaborativo y finalmente a la nueva entidad *rol operacional,* para establecer el rol que el participante juega dentro de la actividad conjunta. Por otro lado, los autores también critican que la especificación del flujo de trabajo instruccional en IMS es poco flexible, en tanto que todas las actividades se desarrollan en paralelo mientras que los actos se realizan de forma necesariamente secuencial. La enmienda a este respecto consiste en adjuntar a la descripción de los actos nueva información para especificar condiciones de transición que permitan realizar un secuenciamiento no lineal, paralelo, condicional o iterativo.

IMS Simple Sequencing

La especificación de secuenciamiento simple [IMS–SS], permite definir un flujo instruccional compuesto por una colección de actividades organizadas en una estructura jerárquica similar a la que se define en [IMS–LD]. En concreto, la propuesta está formada por una compleja red de modelos de información, relacionados entre sí, algunos de los cuales tienen un carácter optativo para describir un proceso de aprendizaje. Por ejemplo, el modelo de definición del secuenciamiento permite describir el flujo de secuenciamiento deseado. El modelo de tracking se utiliza para registrar los resultados de los estudiantes en su interacción con las actividades. El modelo de estado de actividad permite gestionar información acerca del estado de desarrollo de las mismas. El modelo de navegación, que se apoya en el modelo de terminación y en el modelo de entrega, permite atender las solicitudes de navegación de los estudiantes para terminar actividades y entregar aquéllas que hayan sido definidas como subsiguientes. El modelo de replegado permite indicar cuándo una actividad

puede replegarse y considerarse realizada. El modelo de selección y aleatoriedad permite definir recorridos alternativos del árbol de actividades, etc. Esta especificación constituye un marco de trabajo muy flexible para describir muchos de los flujos de instrucción más habituales que se dan cuando se describen procesos de instrucción. En efecto, la combinación de esta especificación con la del diseño instruccional de IMS–LD permite enriquecer considerablemente las capacidades de secuenciamiento de los escenarios. Este es un factor importante cuando se persigue realizar una especificación de alto nivel coercitivo de las intervenciones de los participantes en escenarios colaborativos.

Lenguaje de scripting de CSCL

Reconociendo las limitaciones de la especificación de [IMS–LD], [Miao et al., 2005] han ideado un lenguaje para especificar formalmente scripts de aprendizaje colaborativo. El modelo de información en que se basa este lenguaje indica que un script CSCL está formado por un flujo de trabajo instruccional compuesto por transiciones y actividades. El desarrollo de las mismas queda circunscrito a un entorno poblado con herramientas que usan y producen artefactos y se conectan a las actividades en tanto que conforman recursos de entrada y salida de las mismas. En el plano social, todos los participantes y grupos implicados en la experiencia tienen un rol específico, que puede ser definido por composición, de manera similar a como ocurre con la composición interna de los grupos, que es potencialmente jerárquica. Aunque este modelo resulta apropiado para dar soporte a la especificación de algunos de escenarios de aprendizaje colaborativo, impone, a nuestro juicio, un nivel coercitivo, en ocasiones demasiado elevado. En efecto, como se apunta en [Dillenbourg, 2002] muchas situaciones colaborativas se modelan con scripts de inducción, de instrucción o de entrenamiento, y en ninguno de esos casos un modelo basado en un flujo de trabajo instruccional fijo y estático como el propuesto por estos autores es válido. Además los autores también obligan a que la participación en las actividades venga intermediada por un rol olvidando que esto no siempre es necesario en todas las situaciones colaborativas. Pero sin lugar a dudas, el mayor problema de esta y otras propuestas similares es la rigidez del modelo. Como se discutirá a lo largo de esta tesis, una de las características de los escenarios de aprendizaje colaborativo es su naturaleza eminentemente dinámica y cambiante. Volviendo a los comentarios anteriores, tanto el flujo de trabajo instruccional como la definición de roles y asignación de responsabilidades colaborativas es algo que, en las más de las veces, se ve condicionado por las decisiones de los estudiantes durante el desarrollo de la experiencia y en las menos es prescrito por el escenario. Precisamente en esta línea nuestra propuesta de tesis apunta a utilizar un modelo más flexible y dinámico basado en reglas que permiten atender los acontecimientos ocurridos a lo largo de la experiencia y responder reactivamente a ellos.

Diseño instruccional basado en patrones

Como dijimos al inicio de esta subsección el diseño de escenarios basado en scripts puede resultar más sencillo que la aproximación convencional. Sin embargo, algunos

autores han advertido aún la complejidad de llevar a cabo el proceso de planificación necesario para definir los scripts de colaboración [Kollar et al., 2003] [Dillenbourg, 2002]. Para aliviar esta complejidad, en [Hernández–Leo et al., 2008] [Villasclaras–Fernández et al., 2009] se propone utilizar un mecanismo de descripción basado en patrones de diseño de scripts. En este contexto, un patrón de diseño de scripts debe entenderse como un arquetipo de script de colaboración expresado de forma abstracta (paramétrica) que aparece recurrentemente en el diseño de experiencias colaborativas. Los autores distinguen en este sentido una colección de patrones de ejemplo [Hernandez–Leo et al., 2006] que permiten a los diseñadores instruccionales contemplar el proceso de creación de nuevos scripts como una labor de instanciación y composición de una familia de patrones disponibles en una librería. El proceso, distingue una fase preliminar de selección de patrones, seguida de la configuración de los objetivos de aprendizaje y los prerrequisitos, la definición del flujo de actividades, la configuración de las actividades y los roles y, finalmente, la determinación de los productos obtenidos.

COW

COW [Vantroys & Peter, 2003] [Peter & Vantroys, 2005] es una herramienta que permite interpretar una especificación en IMS – LD y transformarla en una especificación de un proceso de negocio de acuerdo al estándar de workflow XPDL definido por el consorcio [WfMC]. De esta forma, la actividad de IMS se identifica con la actividad de XPDL, la estructura de actividades con el conjunto de actividades o con la definición de un nuevo proceso, los servicios del primero con las aplicaciones del segundo, y el método y la actuación con la definición del proceso y la lógica de transiciones. El objetivo de establecer esta asociación es conseguir una descripción formal de un escenario como un flujo de trabajo de un proceso de negocio para que éste pueda ser interpretado por cualquier motor de workflow que se ajuste al estándar citado. Esto permite desarrollar con éxito escenarios de aprendizaje que satisfacen cuatro requisitos fundamentales. En primer lugar, el mecanismo de traducción mantiene la neutralidad pedagógica del estándar IMS – LD ya que es posible articular diferentes estilos de aprendizaje. En segundo lugar, se pretende dar soporte al desarrollo de actividades colaborativas adaptando el modelo de proceso de negocio a un workflow colaborativo. En tercer lugar, se permite la redefinición dinámica de las rutas instruccionales de modo que éstas puedan ser alteradas bajo demanda sin necesidad de parar el proceso instruccional y finalmente, se posibilita la reutilización de modelos de actividades y cursos preexistentes. El sistema COW, en tanto que es una implantación de la especificación IMS – LD, adolece, no obstante, de todos los elementos necesarios para desplegar satisfactoriamente experiencias de aprendizaje colaborativo tal y como se discutió anteriormente.

Active Document

El Documento Activo [Verdejo et al., 2002], gestado en el grupo LTCS de la Universidad Nacional de Educación a Distancia, es una arquitectura que genera entornos virtuales para permitir a los estudiantes realizar experiencias de aprendizaje colaborativas. El

modelo semántico que permite especificar un escenario de aprendizaje se encuentra organizado en cuatro documentos de especificación: el documento de comunidad, que define la colección de usuarios que desarrollará la experiencia colaborativa, los posibles roles que éstos asumirán y su organización social en grupos. El documento de recursos, que determina la colección de herramientas que se encuentran disponibles en el sistema, el documento activo, que contiene información acerca de las tareas y subtareas que deberán realizarse, los roles involucrados en cada una de ellas, las referencias a las herramientas necesarias para desarrollarlas y el documento de resultados (outcome) que describe los productos generados. Cuando una especificación de un documento activo se compila se genera un entorno que proporciona un espacio de trabajo donde los estudiantes realizan actividades de forma conjunta e interactúan entre sí mediante el uso de las herramientas de soporte.

El modelo del Documento Activo proporciona un mecanismo completo para definir experiencias de aprendizaje colaborativo. Sin embargo, solamente distingue un conjunto fijo de roles (profesor, estudiante y evaluador) que son interpretados por las herramientas integradas. Por ejemplo, el rol profesor puede llevar a cabo funciones de corrección de los productos generados haciendo uso de las mismas. La definición de actividades se articula a través del concepto de tarea colaborativa cuya especificación incluye referencias a herramientas del documento de recursos y a contenidos embebidos en el propio entorno. En lo que respecta al diseño de la colaboración, la propia definición de actividades supone la especificación de un flujo secuencial de tareas que puede controlarse mediante la inclusión de hitos temporales y de precondiciones para desarrollar las mismas. La división del trabajo en subtareas, es sólo posible cuando ésta queda prescrita por el diseñador instruccional y no es posible definir roles específicos para el desarrollo de las mismas. Además su filosofía compilada impide cambios dinámicos en la especificación posteriores a l diseño del escenario

2.5.2.2. Fase de desarrollo

Durante la fase de desarrollo, la tecnología debe dar soporte a la interacción colaborativa que se produce entre los estudiantes mediante la provisión de una serie de medios o herramientas de interacción. Estas herramientas persiguen articular el proceso colaborativo y facilitar el establecimiento de vínculos de unión entre los participantes que fomenten la confianza social y la cohesión de grupo.

A lo largo de estas líneas haremos una revisión de los medios de interacción que aparecen formando parte de las principales entornos virtuales de aprendizaje [Blackboard] [Moodle] [Alf] y las herramientas de Growpware actuales[LotusNotes]. Nos centraremos especialmente en establecer una clasificación de los mismos más que en dar una definición de ellos ya que se trata de herramientas y protocolos estándar y sobradamente conocidos. El marco de clasificación gira en torno a tres dimensiones:

- **Acoplamiento temporal.** El acoplamiento temporal discrimina entre dos posibles valores: medios síncronos y medios asíncronos. En los primeros el protocolo exige que los participantes se encuentren en el mismo instante de tiempo usando el

sistema. En los medios asíncronos, cada participante puede conectarse para usar la herramienta en momentos diferentes.

- **Modo de participación.** El modo de participación establece el rol de participación que desempeña cada usuario en la herramienta. Los dos roles son el de productor y el de consumidor. En este sentido, existen tres posibles valores para esta dimensión: participación 1 a 1 que identifica una situación de interacción bilateral, participación 1 a n, donde existe un productor y muchos consumidores y participación n a n, donde todos los usuarios hacen de productores y consumidores potenciales.
- **Función principal de uso.** Esta dimensión intenta clasificar a las herramientas por su función de uso principal, si bien otros usos diferentes son perfectamente asumibles en la mayoría de los casos. En este sentido distinguiremos entre herramientas dirigidas a dar soporte a la interacción, a la negociación y a la coordinación.

La tabla 2.4, muestra la relación de los medios de interacción colaborativa más comúnmente utilizados en la actualidad clasificado de acuerdo a las tres dimensiones anteriores. Su propósito no es prescriptivo – otras clasificaciones similares podrían haberse establecido – ni tampoco completo. Sólo se trata de ilustrar de qué tipo de sistemas y herramientas estamos hablando cuando nos referimos a medios de interacción colaborativa.

			Acoplamiento temporal	
			Sistemas síncronos	Sistemas asíncronos
Función principal	Interacción	1:1	Sistemas de mensajería instantánea	Correo electrónico
		1:N	Sistema de conferencia web	Foros de discusión, grupos de noticias, blogs
		N:N	Salones de charla	Listas de correo, wikis
	Negociación	1:1	Sistemas de mensajería	Sistemas de regateo
		1:N	Subastas electrónicas coordinadas	Subastas electrónicas
		N:N	Sistemas de votación, sistemas de simulación, editores síncronos	Sistemas de ranking y evaluación, sistemas de simulación, editores asíncronos
	Coordinación	1:1	Teléfono, voz sobre IP	Correo electrónico con documentos adjuntos, repositorios de recursos,
		1:N	Pizarras	Podcast, presentaciones, videos, tablones de anuncios, repositorios de recursos
		N:N	Video conferencias, audio conferencias	Calendarios, sistemas gestores de eventos, repositorios de recursos

Tabla 2.4. Clasificación de los medios de interacción más populares.

2.5.3. Soporte a la integración

Las herramientas de soporte a la interacción colaborativa permiten a los estudiantes realizar las actividades establecidas en el flujo de trabajo instruccional del escenario pedagógico. Como vimos en la subsección anterior, muchas de ellas son comúnmente utilizadas y se han terminado convirtiendo en estándares de facto por lo que la mayoría de los entornos de aprendizaje las incorpora de forma empotrada [Blackboard] [Moodle] [Alf]. Como discutimos en la sección 1.2 de esta tesis, esto es un error ya que las herramientas deberían ser independientes de los entornos e integrarse con éstos con facilidad puesto que las prestaciones de interacción en cada escenario son potencialmente diferentes. A lo largo de esta subsección nos centraremos en describir la familia de soluciones que existen en la actualidad para resolver los problemas de integración, y en particular circunscritos al ámbito de los entornos VLE.

IMS Learning Design, Extensión de [Hernandez–Leo et al., 2005]

Estos autores proponen una nueva extensión a la especificación del estándar [IMS–LD] para satisfacer las necesidades que demanda del desarrollo de experiencias de aprendizaje colaborativo. En concreto, los autores advierten que no existe ningún mecanismo para describir los tipos de interacciones que se van a permitir entre los participantes de una actividad colaborativa (discusión, argumentación, intercambio de ideas, etc.). En efecto, la especificación sólo permite indicar que si varias personas tienen que trabajar juntas en el desarrollo de una actividad lo harán a través de los servicios de su entorno que dan soportan a las diferentes capacidades colaborativas. Es decir, en IMS–LD una actividad será colaborativa en la medida de que disponga de servicios de soporte a la colaboración. La extensión que proponen los autores consiste en la definición de un nuevo tipo de servicios llamado *servicio de grupo*. Esta entidad incluye todos los elementos necesarios para caracterizar un servicio colaborativo: el nivel de consciencia, que proporciona información acerca del contexto socio–colaborativo donde se utiliza el servicio, las políticas de regulación del turno de palabra, que permiten regular las intervenciones de los usuarios dentro del servicio, la colección de roles que participan en el mismo, el ámbito de acceso al servicio dentro del entorno y el tipo de interacción que es soportada (directa, indirecta, mediada por objetos compartidos u orientada a la participación). La propuesta permite distinguir explícitamente las herramientas del entorno que tienen un uso individualizado de las que sirven para ofrecen un servicio de interacción colaborativa y posibilitan contextualizar adecuadamente su uso. Esto supone una clara mejora en el diseño de las actividades, por cuanto las herramientas – los servicios – forman parte integral de la definición de las mismas.

IMS Tools Interoperability

La especificación de IMS para la interoperabilidad entre herramientas [IMS – TI], propone un mecanismo para integrar sistemas desarrollados por terceras partes dentro de entornos virtuales de aprendizaje. La arquitectura general para alcanzar tales objetivos se expresa en términos de tres elementos fundamentales. En primer lugar, las

herramientas externas que son sistemas potencialmente heterogéneos que prestan una colección de servicios de soporte al entorno. En segundo lugar, los proxies. Para integrar cada herramienta dentro del entorno es necesario desarrollar un artefacto software que haga de puente o fachada entre el sistema y la herramienta. Estos proxies implementan un ciclo de vida especificado en la propuesta para permitir al entorno controlar las herramientas. Finalmente, el entorno de ejecución de interoperabilidad, que se añade al VLE para poder comunicarse con las herramientas a través de las operaciones que se definen en su ciclo de vida. La comunicación entre este último elemento y los proxies de las herramientas se lleva a cabo haciendo uso de la tecnología de servicios Web y de acuerdo a la especificación [IMS – WS]. Esto garantiza un nivel de acoplamiento débil entre los proxies y el entorno, permite incluir mecanismos de seguridad y control de protocolo y estandariza el formato de intercambio de información.

La propuesta de interoperabilidad de IMS resulta interesante por cuanto constituye una especificación de referencia para llevar a cabo la integración de diferentes sistemas dentro de una plataforma de e–learning garantizando ciertos niveles de interoperabilidad con un acoplamiento bastante débil. La solución planteada consiste en una arquitectura de componentes que son gobernados de forma centralizada a través de un ciclo de vida. En este sentido, es similar a otras propuestas que analizaremos a continuación. Aunque claramente operativa, este tipo de soluciones presentan un inconveniente importante. La integración de una herramienta se obtiene siempre mediante el desarrollo de un artefacto adaptador – el proxy – que es donde reside la lógica de integración y por tanto, puede decirse que este es el punto real donde reside el acoplamiento. En efecto, el comportamiento de las herramientas, está sometido a un contrato explícito de ciclo de vida que determina las operaciones que pueden realizar las herramientas.

IMS General Web Services

La especificación de servicios Web [IMS – WS], tiene por objeto promover la interoperabilidad entre sistemas de diferentes plataformas de e–learning. Ésta se compone de una serie de perfiles que tratan diversos aspectos normativos para articular una arquitectura de integración basada en servicios Web tales como el direccionamiento punto a punto, la seguridad, el intercambio de contenidos en formato no XML, etc. Pese a ello, el objetivo que se persigue no es crear una arquitectura *plug and play* de servicios que garantice una interoperabilidad completa. Se trata, más bien, de ofrecer interoperabilidad a nivel de aplicación y, en particular, a través de la descripción de los comportamientos expuestos por los subsistemas interconectados mediante servicios Web.

Desde un punto de vista tecnológico la propuesta de servicios Web de IMS ha sido desarrollada para asegurar que todos los servicios definidos bajo este marco usan una misma infraestructura de comunicación con protocolos y formatos de intercambio de mensajes definidos. La especificación presenta un mecanismo mediante el cual la publicación de un servicio Web se obtiene, de forma automática, a partir del mo-

delo de información semántico de la aplicación. De esta manera se consigue que los desarrolladores centren su atención en la lógica de negocio del mismo y no en el soporte tecnológico necesario para publicarlo. El procedimiento automático consiste en definir el modelo de negocio en UML y expresarlo en el formato de intercambio estándar [XMI] que lo describe como un documento XML. A partir de él se obtiene la descripción [WSDL] que describe el servicio mediante la aplicación de una hoja transformación [XSLT] definida por el consorcio para convertir el primer documento en el segundo.

Esta propuesta no proporciona, como ya dijimos antes, un modelo de interoperabilidad completo. Sin embargo constituye un elemento prescriptivo interesante para capacitar a todos los sistemas de e–learning de un mismo modelo de referencia en el uso y publicación de servicios Web. Esta norma, conjugada con otras, puede no obstante constituir un mecanismo potente e interesante para conseguir una fuerte integración con un elevado nivel de interoperabilidad. En concreto nos estamos refiriendo al lenguaje de descripción coreográfico de servicios Web [WS–CDL] y al lenguaje de descripción de procesos de negocio para servicios Web [BPEL4WS]. El primero define un lenguaje de comunicación XML para articular comunicaciones entre cualquier par de participantes en un diálogo soportado con tecnología de servicios Web. El segundo, proporciona una notación y una semántica para especificar procesos de negocio definidos en términos de interacciones de servicios Web.

Estándar OGSA y servicios en Grid

Para llevar a cabo experiencias de aprendizaje colaborativo en un entorno constituido por un conjunto de herramientas externas algunos autores [Bote-Lorenzo et al., 2004] [Bote-Lorenzo et al., 2008] han propuesto utilizar arquitecturas basadas en servicios en Grid de acuerdo al estándar [OGSA]. Esta propuesta parte de la especificación IMS – LD para describir diseños instruccionales y, en concreto, de su extensión para definir servicios de grupo [Hernandez et al., 2005]. La aproximación propone construir un entorno de aprendizaje combinando una colección de servicios en Grid desarrollados por terceras partes que dan soporte a ciertas interacciones colaborativas. Los servicios en grid son recursos hardware o software que exponen su funcionalidad como servicios Web de acuerdo a la interfaz prescrita en la especificación OGSA. Conjugando estas ideas con las del diseño instruccional, cuando un profesor desea poner en marcha una experiencia colaborativa debe hacer una especificación instruccional en un documento IMS – LD que referencia, como servicios de grupo, todos los servicios proporcionados en grid. Un motor, capaz de interpretar dicho documento, buscará en el repositorio de la arquitectura dichos servicios, los levantará y ofrecerá a cada estudiante un entorno colaborativo adaptado al rol que se esté desempeñando. Esta solución plantea, no obstante, algunos problemas serios. En primer lugar los servicios en grid deben ser desarrollados de acuerdo a un estándar específico para conseguir la integración. Esto es un punto en contra de la flexibilidad de desarrollo ya que da lugar a soluciones fuertemente acopladas y dependientes de una infraestructura tecnológica específica. En segundo lugar, la propuesta arquitectónica de grid constituye un middleware para dar soporte a la integración. Sin embargo, las tareas

de coordinación subyacentes que es necesario realizar puntualmente a lo largo del flujo de interacción colaborativa para ofrecer continuidad en el uso trasversal de las herramientas integradas se delegan en las mismas en lugar de automatizarse dentro del middleware. Por último, el middleware propuesto no parece ofrecer ningún mecanismo centralizado para soportar las sesiones de trabajo de los estudiantes ni información del contexto colaborativo.

Lead Flow for Learning Design

Precisamente para dar respuesta a la penúltima debilidad señalada en el midleware anterior – ofrecer continuidad en el uso de las herramientas – los autores han desarrollado el sistema LeadFlow4LD. En efecto, en [Gómez–Sánchez et al., 2009] se reconoce que en la definición de experiencias instruccionales no solamente es necesario especificar, a nivel declarativo, el flujo de trabajo instruccional del escenario y las herramientas necesarias, sino también, a nivel de instancia, proporcionar información acerca de cómo fluyen los datos generados desde unas herramientas a otras a lo largo de la experiencia y qué usuarios pueden utilizarlas. En este sentido, se defiende que un diseño completo incluye 1) la definición del flujo instruccional para articular el secuenciamiento de las actividades, 2) la definición del flujo de datos para determinar cómo se producen y consumen los datos por las herramientas, 3) la definición de la coordinación para mantener sincronizado el progreso del flujo instruccional con el flujo de datos, 4) la definición de la instanciación para identificar los grupos, usuarios y roles que intervienen en el proceso y 5) la definición del modelo de compartición de datos para indicar cómo se comparten potencialmente los datos entre los miembros del grupo, a partir de un sistema de permisos.

El estándar OSGi

El marco de trabajo [OSGi], constituye una especificación de un modelo de componentes completo. Pese a que esta propuesta no está ubicada dentro de la línea de sistemas de e–learning hemos creído conveniente mencionarla en esta sección por cuanto constituye una importante referencia en materia de arquitecturas de integración basada en componentes . De acuerdo a la norma OSGi, las aplicaciones o los componentes que se integran en una plataforma pueden ser instalados, arrancados, parados, actualizados y desinstalados de forma automática sin necesidad de rearrancar la plataforma que los soporta. La gestión del ciclo de vida se lleva a cabo mediante una API que permite detectar la existencia de nuevos servicios o la caducidad de los mismos lo que posibilita a la plataforma adaptarse constantemente a los cambios para mantenerse actualizada. Las plataformas de integración que siguen la especificación OSGi disponen de un repositorio de recursos donde almacenan los componentes (código y datos de despliegue) que han sido integrados en ella. La filosofía de construcción es muy modular. La plataforma es un pequeño núcleo de código sobre la que se van construyendo aplicaciones por la incorporación de nuevos servicios proporcionados por componentes externos. Para facilitar la interoperabilidad entre los componentes OSGi, éstos implementan dos elementos complementarios: los puntos de extensión y las extensiones. Un punto de extensión define un tipo de extensión

que puede ser aprovechada por otro componente para incorporar allí sus servicios. Las extensiones, por su parte, son implementaciones específicas que ofrecen servicios compatibles con ciertos puntos de extensión definidos por otros componentes. Ambos elementos – extensiones y puntos de extensión – se declaran explícitamente mediante un manifiesto XML.

La propuesta OSGi es una solución de integración muy atractiva dentro de la comunidad de desarrolladores ya que proporciona unos elevados niveles de interoperabilidad e integración cuando se pretende desarrollar una arquitectura orientada a componentes que además, en las últimas versiones, ofrece unas capacidades de actualización y adaptación automática cada vez más sencillas. El hecho de ser una especificación con pocas fisuras tecnológicas en la que han colaborado grandes entidades de la comunidad de IT la ha convertido en un estándar de facto que ha dado lugar a diferentes implementaciones sobre la que se han diseñado aplicaciones modulares para distintas áreas y dispositivos (telefonía móvil, PDA, etc.). Sin embargo su mayor ventaja puede que sea, tal vez, su mayor inconveniente. Su buen funcionamiento se debe casi con completa seguridad a que la especificación impone fuertes restricciones sobre el modelo de desarrollo de componentes integrables lo que convierte a la propuesta en una solución fuertemente dependiente de su modelo arquitectónico.

PoEML

La propuesta PoEML [Caeiro et al. 2006] [Caeiro et al., 2008], advierte que la descripción de los servicios que forman parte de un diseño instruccional deberían ser definidos de forma abstracta indicando el comportamiento que se espera de ellos sobre el entorno de aprendizaje en el que se integran. Así, en tiempo de ejecución se podrían utilizar diferentes herramientas que proporcionen una implementación de los servicios especificados. Esta propuesta ofrece un marco para la descripción de servicios que adicionalmente contempla ciertos aspectos de coordinación para posibilitar el modelado de la administración y control de los servicios proporcionados por las herramientas. La descripción de la propuesta de PoEML se articula en torno a tres perspectivas de especificación diferentes: la perspectiva operacional, la perspectiva de interacción y la perspectiva de consciencia. Desde la perspectiva operacional, un servicio se especifica formalmente a partir de una descripción general, el conjunto de operaciones que proporciona, el conjunto de eventos que dispara en cada momento durante su uso y el conjunto de permisos que cada tipo de usuario necesita disponer para acceder a los servicios proporcionados. La perspectiva de interacción describe qué operaciones tienen que ser invocadas para ofrecer un servicio, cómo se organizan estas operaciones y quién las ofrece. Cuando un servicio es arrancado por un usuario, el procesamiento de la interacción trata con tres cuestiones: el elemento particular del entorno sobre el cual se realiza la operación, la gestión de los parámetros de entrada y salida y la composición que debe ser realizada sobre las operaciones primitivas de las herramientas para proporcionar un servicio completo (ejecución secuencial, paralela, iterativa o condicional). Para describir declarativamente esta composición se utiliza un lenguaje de reglas propio basado en la notación XML. Por último, desde la perspectiva de consciencia se describe un modelo que permite

proporcionar información puntual relevante sobre lo que está ocurriendo durante el desarrollo de la actividad colaborativa. Esta información es propagada por las herramientas en forma de eventos. Cuando se lanza un evento, comienza el procesamiento del mismo. Los eventos originados directamente por una herramienta son eventos primitivos. Sin embargo, éstos pueden componerse mediante la aplicación de diferentes operadores (filtrado, agregación o correlación) para generar eventos compuestos. Los eventos son recogidos por otras herramientas, los usuarios finales o las reglas definidas en el lenguaje XML que se disparan ante la llegada de un evento y la satisfacción de ciertas condiciones de disparo. En este último caso, las reglas permiten adaptar el comportamiento de las herramientas en función de la información proporcionada por el evento.

La especificación que proponen los autores de PoEML resulta interesante. No obstante, esta especificación se encuentra fuertemente sometida a los estándares de diseño instruccional que discutimos con anterioridad. Además, podemos destacar una serie de inconvenientes que la hacen menos atractiva de lo que resulta a primera vista. En primer lugar, la propuesta PoEML parece estar descrita como un marco teórico–tecnológico que plantea más un paradigma para describir servicios y coordinar su uso dentro de plataformas de e–learning que una propuesta en firme implementada que pueda ser evaluada. En segundo lugar, y tal vez debido a que no existe una implementación de referencia, pese a que se presenta un lenguaje de coordinación, no existe un modelo conceptual que describa las entidades específicas que pueden aparecer para expresar dicha lógica. Los autores plantean que las reglas del lenguaje permiten coordinar el uso de las herramientas integradas en un entorno de aprendizaje. No obstante, a menudo, las tareas de coordinación no afectan tanto a las operaciones proporcionadas por las herramientas sino a alteraciones sobre el contexto social y colaborativo de la comunidad virtual de aprendizaje. Además, las posibilidades expresivas del lenguaje de reglas de PoEML resulta muy reducido en cuanto a que el control de flujo está limitado a unos pocos operadores y en tanto que mucha lógica adaptativa resulta poco operativa expresarla en forma de reglas siendo mejor expresarla de forma algorítmica.

2.5.4. Soporte a la adaptación

Los sistemas adaptativos inteligentes de e–learning que se encuentran disponibles en la actualidad provienen de la confluencia de dos áreas de investigación diferentes, por un lado los sistemas tutores inteligentes y por otro los sistemas de hipermedia adaptativos. [Brusilovsky, 1999] reconoce, en este sentido, que es posible distinguir varios tipos de sistemas de adaptación:

- **Sistemas de secuenciamiento curricular.** Los sistemas de secuenciamiento curricular constituyen una solución para el paradigma pedagógico clásico de transmisión de conocimiento y conforman un tipo de sistemas tutores inteligentes (ITS) cuyo propósito es el de proporcionar al estudiante una ruta formativa, dentro del espacio de materiales pedagógicos, adaptada a sus necesidades y estilos de aprendizaje.

- **Sistemas de soporte a la resolución de problemas.** Los sistemas de soporte a la resolución de problemas aplican también técnicas de tutorización inteligente para asistir al aprendizaje basado en la resolución de problemas propio del enfoque constructivista. En este sentido, existen tres aproximaciones principales: sistemas de análisis inteligente de soluciones (que determinan la corrección en la respuesta del alumno y establecen las acciones adaptativas a aplicar), sistemas de asistencia interactiva a la resolución de problema (que ayudan al estudiante durante el proceso de aprendizaje basado en problemas) y sistemas de resolución de problemas basado en casos (que utilizan ejercicios anteriormente resueltos por el propio estudiante para orientarle).

- **Sistemas de hipermedia adaptativos.** Los sistemas de hipermedia adaptativos proporcionan capacidades inteligentes para adaptar la experiencia del usuario dentro de un entorno Web de aprendizaje a sus necesidades y habilidades. La adaptación en este sentido suele basarse en adecuar la navegación del usuario dentro del hiperespacio de recursos pedagógicos y en la composición ad–hoc de contenidos siempre en función de las características del mismo. En los últimos años están proliferando sistemas de esta naturaleza que se aplican dentro del área del CSCL como [Alf] o [Constantino-González & Suthers, 2002]. Nosotros en este trabajo nos centraremos en este último tipo de sistemas.

La arquitectura de todo sistema adaptativo se ajusta a un modelo conceptual de funcionamiento basado en tres fases. A continuación describimos brevemente cada una de ellas:

- **Adquisición de datos.** Todo sistema de adaptación basa su estrategia de adecuación en la información que éste dispone del usuario. En efecto, estos sistemas suelen recoger una colección de datos que pueden clasificarse, de manera general, en tres categorías:

  – *Datos de usuario.* Esta categoría representa todos los datos relativos a la información personal del usuario, tal como su identificación, localización geográfica, información demográfica, destrezas y habilidades, intereses y preferencias etc.

  – *Datos del entorno.* Los datos del entorno se refieren a información sobre el entorno virtual de aprendizaje sobre el que trabaja el usuario (comunidad a la que pertenece, actividades y proyectos en los que está implicado, etc.). Esta información suele tener un carácter dinámico ya que cambia a medida que el usuario progresa en su proceso instruccional.

  – *Datos de uso.* Este tipo de información es recolectado a partir de los datos de traza provenientes de la interacción que los usuarios del entorno hacen sobre cada uno de las herramientas puestas a su disposición, típicamente foros de discusión, repositorios de recursos, herramientas de mensajería instantánea, etc. A partir de esta información, altamente dinámica, el sistema es capaz de inferir adaptaciones para aplicarlas como se discute en los puntos posteriores.

- **Representación de datos.** El conjunto de información capturada en la fase anterior es procesado para constituir un *modelo de información de usuario* que capture todas sus características esenciales. En los casos más sencillos éste contiene datos elementales de carácter personal mientras que en otros casos es enriquecido progresivamente para reflejar los conocimientos del usuario. En sistemas de adaptación aplicados al CSCL también es frecuente encontrar *modelos de grupo* que caracterizan las propiedades de los colectivos de aprendizaje que existen dentro de la comunidad. Estos modelos suelen ser construidos compositivamente a partir de los modelos de usuario de los miembros del grupo. [Gaudioso, 2002] distingue tres mecanismos de representación computacional de los modelos:
  - *Modelos explícitos.* Las representaciones explicitas se realizan aplicando algún formalismo declarativo para representar el conocimiento que contienen los modelos de usuario, típicamente reglas. La ventaja de esta aproximación es que los administradores y diseñador instruccionales pueden alterar estos modelos para ajustarlos puntualmente a los requerimientos de cada escenario ya que resultan inteligibles por un humano.
  - *Modelos implícitos.* Las representaciones implícitas hacen uso de formalizaciones no simbólicas del conocimiento de los modelos – típicamente redes neuronales – para obtener una colección de atributos y propiedades que podrán ser ajustar de forma automática por aplicación de algún método de aprendizaje asociado al formalismo de representación. Por contrapartida, el conocimiento capturado no puede ser claramente expuesto para un humano.
  - *Modelos híbridos.* También existen soluciones híbridas que hacen uso de ambas aproximaciones para aprovechar las ventajas representativas de ambos modelos. De un lado la claridad de los modelos explícitos y de otro la capacidades de inferencia de los modelos implícitos.
- **Ejecución de la adaptación.** A partir de la formalización de los modelos de usuario, los sistemas adaptativos pueden determinar la tarea de adaptación que deben aplicar para adecuarse a las necesidad. En este sentido, podemos distinguir tres tipos de tareas de adaptación:
  - *Adaptación de la navegación.* La adaptación de la navegación es el proceso mediante el cuál los hipervínculos que le son presentados al usuario durante su interacción con el sistema son alterados (ocultados, reordenados, etc.) para orientarle dentro del espacio de aprendizaje.
  - *Adaptación de la presentación.* La adaptación de la presentación consiste en la construcción compositiva personalizada que se hace del material que le es proporcionado al usuario durante su interacción con el sistema.
  - *Adaptación del contexto socio–colaborativo.* En sistemas colaborativos, el contexto socio–colaborativo donde se desenvuelve el usuario (grupos, actividades, proyectos, etc.) también puede ser adaptado.

Debido a la relativa reciente aparición de los entornos de aprendizaje colaborativo, son pocos los sistemas adaptativos que existen en la actualidad vinculados a este tipo de experiencias. Junto con [Alf] y [Constantino-González & Suthers, 2002], Pelican es uno de estos sistemas. En efecto, como se discutirá en profundidad a lo largo del capítulo 3 de este trabajo el modelo de usuario de la plataforma esta constituido por datos de usuario que son introducidos por los administradores al registrar usuarios y datos del entorno capturados en los contextos de usuario (ver sección 3.5). El modelo de inferencia está basado en reglas de manera que cuando se produce un acontecimiento relevante en el entorno, la plataforma aplica la estrategia de adaptación pertinente. Esta puede consisir en actualizar el modelo de usuario (sus datos de entorno) o aplicar tareas de adaptación de la navegación (por ocultación de enlaces sobre los recursos disponibles) y de adaptación del contexto socio–colaborativo donde se desarrolla la experiencia del usuario (creación de grupos, activación de actividades, etc). La diferencia esencial con otros modelos adaptativos como el propuesto en [Alf] es que las estrategias de recolección de los datos de uso en estos sistemas se encuentran cableados ya que las herramientas de interacción se encuentran empotradas en el entorno. En Pelican, la captura de datos debe expresarse a través de reglas (ver sección 3.5 y subsección 4.5.4) puesto que los medios de interacción disponibles son externos a la plataforma.

2.6. Conclusiones

A lo largo de este capítulo hemos presentado una caracterización del las experiencias de aprendizaje colaborativo. El primero de los objetivos, a este respecto, era precisar que entenderemos en este trabajo por aprendizaje colaborativo, y discutir después este paradigma pedagógico desde su perspectiva psicológica y pedagógica. En la dimensión psicológica la preocupación ha sido intentar entender cómo tiene lugar el aprendizaje entre los alumnos cuando se ubica a éstos en una situación de trabajo colaborativo. En concreto, tratamos en esta línea de responder a dos preguntas principales: por qué se produce el aprendizaje cuando los estudiantes son organizados en grupos para trabajar juntos y en qué medida ese aprendizaje produce distintos y mejores resultados que los obtenidos en otras aproximaciones pedagógicas. En este sentido, hemos revisado las corrientes principales que constituyen el fundamente teórico en que se apoya este paradigma. Desde el enfoque pedagógico, por el contrario, el problema ha sido intentar encontrar métodos y estrategias que puedan ser de utilidad para inducir el aprendizaje en escenarios colaborativos. La pregunta a este respecto es cómo debemos diseñar estos escenarios para aumentar la probabilidad de que se produzcan ciertos tipos de interacción social hipotéticamente beneficiosos en este tipo de experiencias. Para encontrar una respuesta se han revisado los principales métodos colaborativos propuestos desde la comunidad de pedagogía. Sin embargo, nuestros objetivos son encontrar un formalismo que permita caracterizar de forma completa los escenarios colaborativos y comprobar la viabilidad de hacer esa especificación computable. Para hacer frente a esta necesidad hemos presentado en este capítulo todas las aportación encontradas en la literatura relativas a las experiencias de aprendizaje colaborativo soportado por computador.

3 Solución propuesta

En el capítulo anterior presentamos el aprendizaje colaborativo soportado por computador desde sus perspectivas psicológica, pedagógica y tecnológica. Nos preocupamos en este sentido de discutir por qué funciona este paradigma, cómo se articula en la práctica y qué herramientas y soluciones computacionales existen para darle soporte. En este capítulo describimos la plataforma Pelican, un sistema que tiene por objeto dar cobertura tecnológica a este tipo de experiencias tanto en las fases iniciales de diseño como en el desarrollo posterior de las mismas.

La plataforma Pelican proporciona todos los elementos necesarios para modelar escenarios colaborativos y provee de una infraestructura tecnológica para articular los mismos en diferentes contextos sociales. Su arquitectura está constituida por cuatro subsistemas. El subsistema social se encarga de dar soporte al entramado social donde se desarrollan las experiencias de aprendizaje. El subsistema de colaboración proporciona los mecanismos necesarios para definir este tipo de experiencias y poder desplegarlas posteriormente en diferentes contextos sociales. El subsistema de integración permite que las herramientas externas se integren en el entorno a distintos niveles de interoperabilidad. Y finalmente, el subsistema de intervención posibilita adaptar los escenarios bajo demanda de los requerimientos de cambio de los flujos de trabajo instruccional. Esta última facilidad, permite capturar todos los aspectos dinámicos que aparecen inherentemente vinculados a la mayoría de las experiencias de aprendizaje colaborativo y constituye la aportación fundamental de este trabajo de tesis.

3.1. Introducción

En el capítulo anterior describimos el aprendizaje colaborativo soportado por computador centrándonos en su dimensión pedagógica, psicológica y tecnológica. Los resultados allí arrojados, se alinearon con el primero de los objetivos identificados en el capítulo inicial de esta tesis: realizar un estudio de todas las aportaciones relevantes que se han hecho dentro del área del CSCL. En este capítulo se abordan los otros dos objetivos planteados: proporcionar un modelo para la especificación formal de los escenarios de aprendizaje colaborativo y presentar la plataforma Pelican, un sistema capaz de dar soporte computacional a dicha especificación formal.

De acuerdo a lo anterior, la plataforma Pelican puede ser considerada desde tres perspectivas diferentes. A continuación describimos en detalle cada una de ellas:

- **Pelican como herramienta de modelado para la especificación de escenarios.** Pelican es, en primer lugar, una herramienta de autor que sirve para especificar formalmente escenarios de aprendizaje colaborativo de forma computable. En este sentido, la plataforma proporciona un lenguaje de modelado educacional (EML) similar a otras propuestas [Rodriguez & Verdejo, 2004] [IMS–LD] [IMS–SS]. Sin embargo, también existen elementos que le diferencian de ellas. En primer lugar, el lenguaje de Pelican está dirigido por modelos y no por la sintaxis como todas las especificaciones citadas. Esto quiere decir que el diseñador instruccional es asistido en su trabajo por las interfaces de usuario de la plataforma proporcionándole los elementos necesarios para realizar la especificación de los escenarios de forma interactiva. En segundo lugar, el lenguaje de Pelican proporciona un mecanismo de especificación sintáctico y basado en reglas que tiene por objeto capturar todos los aspectos dinámicos que aparecen vinculados de forma inherente a las experiencias colaborativas. Con este mecanismo, es posible definir respuestas reactivas ante los acontecimientos ocurridos en el flujo de instrucción y modificar así dinámicamente tanto el comportamiento de la plataforma como el de las herramientas externas integradas para adaptarlas continuamente a los requerimientos potenciales de cambio de los escenarios de aprendizaje.

- **Pelican como entorno de aprendizaje orientado a la colaboración.** Adicionalmente, Pelican constituye un entorno virtual de aprendizaje orientado al desarrollo de experiencias colaborativas. Para dar soporte a las mismas, la plataforma proporciona una colección de mecanismos tecnológicos que serán descritos a lo largo de este capítulo en detalle y que se alinean con aquéllos para la especificación formal de los escenarios. Es decir, si los artefactos del punto anterior permiten articular la fase de diseño, los de este punto, dan soporte al desarrollo de experiencias de aprendizaje colaborativo. Como recordaremos más adelante, esto permite llevar a cabo la implantación de este tipo de experiencias en Pelican en dos fases complementarias. Primero, en la fase de diseño, el diseñador instruccional debe realizar la especificación del escenario y posteriormente, en la fase de desarrollo, desplegarla sobre el entramado social y colaborativo soportado por este tipo de artefactos. El valor diferencial a este respecto en relación a otros entornos

virtuales existentes tales como [Blackboard] [Moodle] o [Alf] radica, según nuestros análisis, en su flexibilidad. La plataforma Pelican ofrece un desacoplamiento total entre la especificación formal de escenarios (y sus productos resultantes) y el despliegue de estos mismos en diferentes contextos. Esta característica permite, por un lado fomentar la reutilización de las especificaciones de los escenarios de aprendizaje y por otro controlar separadamente por parte de profesores y monitores el desarrollo de las experiencias en distintos colectivos y a distintos niveles sociales (clase, grupo, individuo, etc.)

- **Pelican como plataforma de integración.** En tercer lugar, Pelican puede ser considerada como una plataforma de integración que permite que herramientas externas, potencialmente desarrolladas por terceras partes, se integren de manera sencilla y eficaz dentro de la misma. La justificación para este principio fundamental de diseño radica en la hipótesis de que el desarrollo de experiencias de aprendizaje requiere de una colección de herramientas específicas de soporte a la colaboración que comúnmente dependen de los objetivos pedagógicos perseguidos, el contexto socio–cognitivo y el dominio de aplicación. En efecto, los requerimientos de interacción social, de soporte a la decisión, de modelado y simulación etc., presentan peculiaridades específicas para cada escenario de uso. Por eso no es conveniente ofrecer una colección canónica, cerrada y generalmente insuficiente de servicios de interacción, como ocurre tradicionalmente en los entornos virtuales [Blackboard] [Moodle] o [Alf], sino que debemos proporcionar la capacidad de extender las prestaciones de la plataforma con aplicaciones exteriores. La integración sencilla significa que ésta no debe estar fundamentada en la aplicación de protocolos de integración aunque éstos sean estándares de facto. Los esfuerzos de desarrollo para la integración en este sentido deben minimizarse al máximo. La integración eficaz significa que debe proporcionarse un nivel de interoperabilidad entre las herramientas lo suficientemente elevado como para desarrollar la experiencia, si bien dicho nivel no es un valor absoluto sino que depende de cada escenario. A este respecto, el papel de Pelican es el de un elemento orquestador que proporciona una sensación de continuidad a los usuario en el uso transversal de las herramientas a lo largo del flujo de instrucción. Como veremos, esto se traduce en poder especificar declarativamente protocolos que fomenten la interoperabilidad conectando los productos de salida generados por unas herramientas como recursos de entrada de aquéllas otras que sean utilizas en las actividades subsiguientes dentro del flujo de instrucción.

A lo largo de este capítulo, describiremos en detalle todos los elementos tecnológicos que articulan las tres perspectivas funcionales que acabamos de describir. En concreto, en la siguiente sección comenzaremos haciendo una revisión general de la arquitectura de la plataforma y presentaremos cada uno de los cuatro subsistemas que la constituyen. En los subsiguientes apartados abordaremos su estudio en profundidad, discutiendo, para cada uno de ellos, los objetivos específicos perseguidos, el modelo software utilizado para dar soporte a los mismos y las interfaces Web que son proporcionadas para interactuar con la plataforma.

3.2. Visión arquitectónica de Pelican

Como describimos en el capítulo anterior, la puesta en práctica de experiencias colaborativas soportadas por computador requiere por un lado prestar especial atención al diseño de los escenarios de aprendizaje y por otro disponer de la infraestructura tecnológica necesaria para llevarlos a cabo. En esta labor aparecen involucrados hasta un total de cinco actores bien diferenciados: administradores, diseñadores, profesores, monitores y estudiantes. La plataforma de aprendizaje Pelican, que presentamos en este capítulo ofrece una colección de servicios que dan soporte a las responsabilidades que cada uno de ellos tienen asignadas dentro de la experiencia colaborativa de aprendizaje:

- **Pelican para diseñadores.** A la hora de especificar un escenario, los diseñadores instruccionales definen en la plataforma los aspectos sociales y colaborativos del mismo determinando los tipos de miembros y colectivos requeridos y las actividades que lo constituyen así como la colección de recursos, roles y herramientas que serán necesarias para realizarlas. Adicionalmente, los diseñadores también pueden definir estrategias reactivas para indicar a la plataforma cuál debe ser su comportamiento ante determinadas condiciones ambientales o cómo debe adaptarse a los requerimientos de cambio del flujo de trabajo instruccional.

- **Pelican para profesores.** Los profesores por su parte, utilizan la plataforma para desarrollar la experiencia de aprendizaje colaborativo a lo largo del tiempo con diferentes colectivos de estudiantes y desplegar ésta potencialmente de manera transversal a distintos niveles sociales dentro del escenario, típicamente clase, grupo e individuo. La arquitectura de Pelican permite llevar a cabo la gestión de la experiencia de forma separada e independiente para cada colectivo social lo que permite su evaluación haciendo uso del ciclo de vida de las actividades.

- **Pelican para monitores.** El desarrollo de las experiencias de aprendizaje dentro de cada grupo debe ser supervisado regularmente y, si procede, deben aplicarse los mecanismos de control previstos. Los monitores son los encargados de llevar a cabo estas tareas y para ello la plataforma gestiona el ciclo de vida de las actividades. Pero como se verá en la sección 3.5 y más específicamente en ejemplo de la subsección 4.4.3, Pelican también permite, a través de sus facilidades de intervención, automatizar el cálculo de los indicadores de análisis y la aplicación de mecanismos de control cuando se alcanzan ciertos valores umbrales sobre los mismos.

- **Pelican para estudiantes.** Los estudiantes perciben la plataforma Pelican como un entorno virtual de aprendizaje constituido por una colección de espacios de trabajo aparentemente independientes entre sí pero subyacentemente asociados a los diferentes colectivos sociales donde se desarrollan las experiencias de aprendizaje colaborativo. En estos espacios de trabajo es, precisamente, donde tiene lugar toda la interacción colaborativa dentro de la plataforma. Por tanto es allí donde se despliegan los escenarios y se incorporan las herramientas de soporte a la colaboración integradas.

- **Pelican para administradores.** Finalmente, los administradores son los encargados de dar asistencia a todas las tareas que se realizan a lo largo de las fases de diseño y desarrollo de la experiencia y que involucran a otros actores. Así, por ejemplo, éstos, comúnmente, realizan labores de gestión social ingresando nuevos usuarios y creando nuevos grupos, asisten a los profesores en el desplegado y replegado de los escenarios en espacios de trabajo, o definen los procedimientos de intervención para capturar aspectos dinámicos tales como la creación de nuevas actividades, grupos, etc.

Desde un punto de vista arquitectónico, la plataforma Pelican está constituida por cuatro subsistemas interrelacionados que serán objeto de estudio detallado en los apartados subsiguientes. A continuación describimos brevemente las funcionalidades de cada uno de ellos:

- **Subsistema social.** El subsistema social es el encargado de proporcionar toda la infraestructura tecnológica necesaria para dar soporte a la definición del entramado social donde se desarrollan las experiencias de aprendizaje colaborativo. A partir de esta definición el sistema ofrece facilidades para poblar dicha estructura con distintos tipos de colectivos y estudiantes reales.

- **Subsistema de colaboración.** El subsistema de colaboración permite llevar a cabo la definición de los aspectos colaborativos de los escenarios de aprendizaje. Este subsistema articula dicha definición en términos de proyectos y actividades que primero son definidos de manera general por los diseñadores instruccionales y posteriormente pueden ser implantados, dentro de los espacios de trabajo, en diferentes colectivos del entramado social subyacente soportado por el subsistema social.

- **Subsistema de intervención.** El subsistema de intervención proporciona un lenguaje de scripting para definir procedimientos que realizan transformaciones puntuales y necesarias sobre la plataforma y las herramientas externas. Estas transformaciones están dirigidas a dar respuesta reactiva a las necesidades adaptativas que demandan los acontecimientos que se producen en el flujo de trabajo instruccional de la mayoría de escenarios colaborativos. Para evitar que la aplicación de tales transformaciones se realice de forma manual por parte de los administradores, el subsistema de intervención proporciona también un mecanismo tecnológico basado en reglas y dirigido por eventos para especificar declarativamente la ejecución temporal de las trasformaciones.

- **Subsistema de integración.** El subsistema de integración proporciona diferentes mecanismos para permitir a los desarrolladores de herramientas externas incorporar los servicios proporcionados por éstas a diferentes niveles de interoperabilidad e integración con el entorno de la plataforma. Cada mecanismo no es incompatible con los demás. Es decir, las herramientas pueden utilizar varios de ellos simultáneamente ya que cada uno ofrece, a este respecto, distintas posibilidades y supone un grado diferente de agnosticismo con respecto a los modelos internos de la plataforma.

Ambas perspectivas anteriores – actores y subsistemas – pueden ser reconciliadas en la imagen arquitectónica que aparece representada en la figura 3.1. En ella se discute la relación entre los subsistemas y el uso que los actores hacen de los mismos. Debe entenderse, dentro de esta figura, que cuando uno de estos elementos se apoya en otro, se está representando una relación de uso.

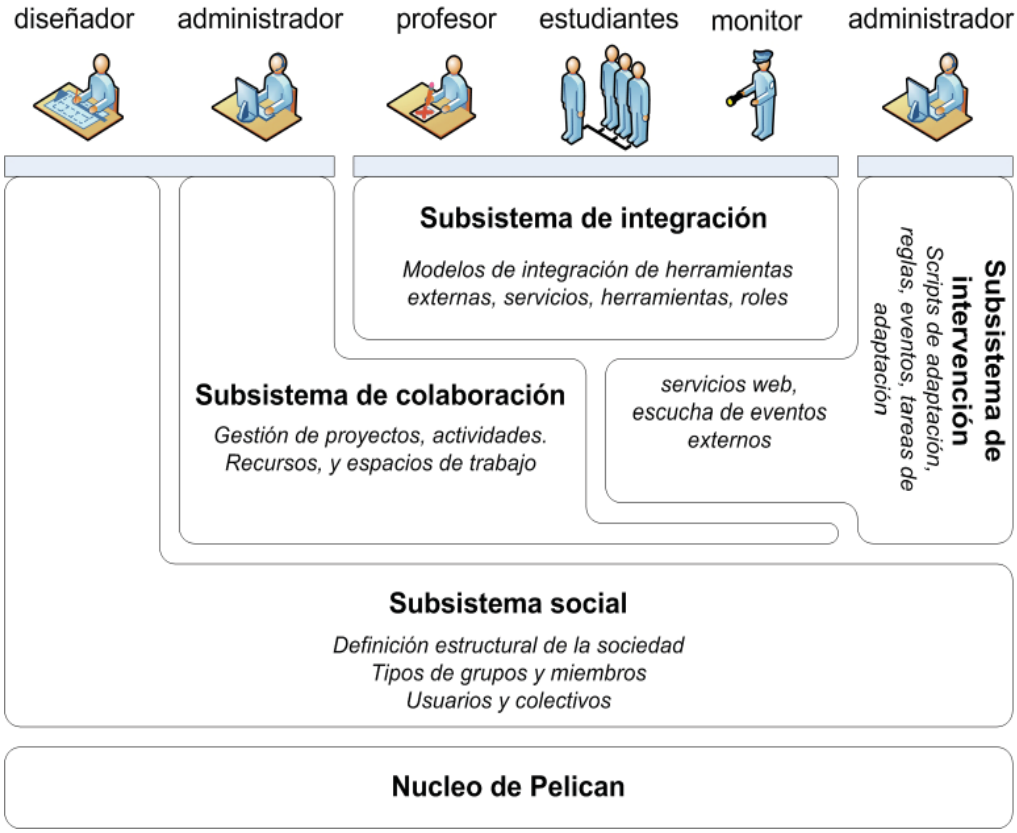

Figura 3.1. Arquitectura general de la plataforma Pelican.

Como puede apreciarse en la figura, toda la arquitectura descansa sobre un núcleo de librerías que permiten soportar aspectos de bajo nivel relacionados con el control del flujo de navegación de aplicaciones Web, el mantenimiento de la sesión de usuario en el navegador del cliente, etc. Por encima, aparece el subsistema social que ofrece capacidades para la definición estructural y administración de los aspectos sociales del aprendizaje. El subsistema de colaboración se encarga de la gestión de los proyectos y las actividades colaborativas y se apoya en las prestaciones del subsistema social, ya que las experiencias se desarrollan sobre los colectivos de estudiantes mantenidos por éste. El subsistema de intervención gestiona las transformaciones adaptativas que deben ser realizadas sobre los dos subsistemas anteriores y también sobre las herramientas externas integradas en el entorno, por eso descansa sobre ellos. Finalmente, el subsistema de integración se apoya en el subsistema de colaboración y en el de intervención para hacer posible la incorporación de nuevas herramientas externas al entorno de Pelican.

Por lo que respecta a los actores, todos ellos hacen uso de los subsistemas social y de colaboración, y en especial los diseñadores y administradores, que son los encargados de definir el entramado social y poblarlo con usuarios reales posteriormente. Los estudiantes, monitores y profesores, tienen su participación en la plataforma principalmente a través del uso de las herramientas integradas en el entorno. Finalmente el sistema de intervención es utilizado fundamentalmente por administradores y diseñadores. Los primeros para especificar los scripts de adaptación y la planificación temporal de la ejecución de los mismos y los segundos para transmitir a los primeros la lógica adaptativa que debe ser especificada.

3.3. Soporte a la estructuración social

Uno de los aspectos más consolidados dentro de la literatura del aprendizaje colaborativo indica que toda experiencia de aprendizaje de esta naturaleza se desarrolla siempre sobre un determinado entramado social. Generalmente se suele hacer referencia a este entramado mediante el término de *comunidad de aprendizaje*. Al hablar de aprendizaje colaborativo soportado por computador (CSCL) dicha comunidad se reúne en torno a la tecnología de soporte y se constituye así una *comunidad virtual de aprendizaje*.

En algunas ocasiones, las comunidades virtuales de aprendizaje son obtenidas a través de un proceso constructivo que intenta representar una comunidad existente en el mundo real. Tal es el caso de la educación formal. Allí, los entornos virtuales de aprendizaje como [Moodle] [Alf] o [Blackboard] dan soporte a una comunidad virtual que refleja la realidad social de la institución donde se desarrollan las experiencias colaborativas. En otras ocasiones, por el contrario, la comunidad virtual existe primero al proporcionar a los usuarios un medio de interacción y como consecuencia surge una comunidad real asociada a la misma. (Los sistemas software de soporte que típicamente se utilizan en este caso son también llamados comunidades virtuales de aprendizaje y, como tales, fueron descritos en el capítulo anterior. Esta lamentable polisemia hace confusa la discusión para distinguir la comunidad como realidad social y como herramienta software. A lo largo de este texto al hablar de comunidad virtual de aprendizaje nos referiremos a la primera de las acepciones).

Sea como fuere, a la hora de llevar a la práctica experiencias de aprendizaje colaborativo es preciso realizar dos tipos de tareas que aparecen alineadas con las fases de diseño y desarrollo. A continuación comentamos brevemente cada una de ellas:

- **Modelado social de la comunidad.** En la fase de diseño de la experiencia, los diseñadores instruccionales llevan a cabo una colección de tareas que tienen por objeto dar forma a la comunidad virtual de aprendizaje. Esta estructuración requiere considerar los tipos de colectivos sociales, su tamaño y composición interna, su nivel de heterogeneidad, su tiempo de vida o los tipos de actores involucrados en el proceso. Así por ejemplo, en educación formal, es común identificar como colectivos el curso, la clase o el equipo colaborativo y como actores implicados el profesor o el estudiante.

- **Construcción de la comunidad virtual.** Una vez que la estructura social de la comunidad virtual ha sido definida es posible construir ésta a través de tareas típicas de administración tales como el registro de nuevos usuarios en el sistema o la creación de cursos, grupos y equipos de trabajo específicos.

La descripción anterior transmite una idea de secuencialidad en las labores de construcción de una comunidad virtual. Sin embargo, en la práctica esto no es así. La naturaleza cambiante de las comunidades virtuales, tal vez en respuesta a una constante necesidad de adaptación a los escenarios colaborativos a los que se pretende dar soporte, hace que el uso de ambas fases de especificación social se extienda a lo largo de todo el tiempo de vida de la misma.

El propósito del subsistema social de la plataforma cuya descripción abordamos en esta sección es, precisamente, atender a todos los requisitos relacionados con los procesos de estructuración social, y en particular a los relativos a la construcción de comunidades virtuales de aprendizaje, dando soporte a las tareas que se inscriben dentro de las fases anteriores.

En efecto, con respecto al modelado social, la hipótesis de partida de este subsistema es que toda comunidad virtual presenta un entramado social inherente que puede ser caracterizado a partir de una colección de estructuras organizativas de agregación dinámicamente cambiantes. Dentro de ellas, puede haber diferentes tipos de miembros que tienen conferidas ciertas responsabilidades sociales. Para dar soporte a esta idea, la plataforma proporciona un sencillo lenguaje de modelado que utiliza el concepto de **actor** para representar a los tipos de miembros y el de **plantilla de grupo** para representar a dichas estructuras organizativas.

Por lo que respecta a las tareas propias de la fase de desarrollo, esto es, la construcción de comunidades virtuales de aprendizaje, Pelican ofrece facilidades que van dirigidas a la administración y mantenimiento de las mismas, tales como el registro de nuevos **usuarios** en la plataforma o la creación y borrado de nuevos **grupos**. La misión de la plataforma en este sentido es velar por el cumplimiento de las condiciones estructurales impuestas en la fase de diseño precedente. De esta manera, por ejemplo, cada vez que se crea un grupo debe indicarse cuál es la plantilla de grupo utilizada para su construcción, lo cual condiciona su estructura interna así como la cantidad de grupos de ese tipo que pueden ser creados en el sistema.

El proceso constructivo completo para obtener una comunidad virtual de aprendizaje puede ser considerado como un trabajo que se desarrolla a tres niveles de abstracción diferentes, tal y como aparece representado en la figura 3.2:

- **Nivel de lenguaje.** El nivel de lenguaje es el proporcionado por el subsistema social para permitir a los diseñadores instruccionales definir la estructura de una comunidad. Como se comentó previamente, este lenguaje de modelado está articulado en torno a dos conceptos fundamentales, la plantilla de grupo y el actor. Es decir, cualquier estructuración queda descrita a partir de una relación de plantillas de grupo constituidas por otras plantillas de grupos hijas y actores.

- **Nivel de modelo.** Los diseñadores instruccionales utilizan el lenguaje proporcionado por el subsistema social para definir un **modelo de sociedad** específico que describe los elementos y restricciones estructurales propios de un determinado tipo de comunidad. En concreto, el trabajo a este nivel consiste en identificar la colección de actores participantes en la comunidad y las plantillas de grupo específicas requeridas, indicando, para cada una de ellas, su composición interna en términos de otros actores y plantillas de grupo. Además, los modelos de sociedad incluyen restricciones de cardinalidad mínima y máxima permitida para ambos tipos de entidades lo cual permite condicionar la estructura de la comunidad subyacente. En el nivel de modelo de la figura 3.2, pueden verse definidos dos modelos de sociedad diferentes para un colegio y una universidad. El primero prescribe que, desde una perspectiva estructural, un colegio está formado por un director y un conjunto de no más de 50 clases, cada clase contiene exactamente un profesor, entre 1 y 20 estudiantes y entre 1 y 10 grupos y cada grupo está formado por un profesor y entre 2 y 5 estudiantes. Por su parte, el modelo de sociedad universitario indica que una universidad tiene un único decano, un claustro de profesores formado por no más de 20 miembros y el propio decano así como una colección de departamentos hasta un total de 5. Cada departamento está formados por equipos docentes, con hasta 5 profesores, y un coordinador y grupos de investigación que incluyen hasta 20 investigadores y 1 jefe de grupo. Como puede apreciarse, el proceso de especificación del modelo de sociedad consiste en la utilización de instancias de los elementos del nivel de lenguaje (actor y plantilla de grupo) para definir tipos específicos de actores y grupos (estudiante, profesor, investigador, clase, claustro...). Las líneas discontinuas de la figura entre el nivel de lenguaje y de modelo indican esta relación de instanciación.

- **Nivel de comunidad.** Una vez obtenido un modelo de sociedad, los administradores pueden utilizar este artefacto para crear una **comunidad virtual de aprendizaje** asociada al mismo. Las tareas de creación, en este sentido, consisten, como ya se apunto anteriormente, en registrar a los usuarios reales dentro de la plataforma, crear grupos a partir de alguna plantilla de grupo y enrolar a los primeros en los segundos indicando el actor que encarnarán estos en cada grupo. La responsabilidad del modelo de sociedad es, a este nivel, dirigir el proceso de construcción de la comunidad imponiendo ciertas restricciones estructurales. En el ejemplo de la figura 3.2, se han creado dos comunidades virtuales para sendos colegios a partir del modelo de colegio (Diego Velázquez y Antonio Machado) y una para una universidad a partir del modelo de universidad (UNED). Nuevamente las líneas discontinuas indican aquí relaciones de instanciación. Como puede apreciarse, los elementos del modelo de sociedad, que fueron descritos como instancias del lenguaje, son contemplados desde el nivel de comunidad como los elementos propios de un lenguaje específicamente creado para construir la comunidad virtual. Es decir, cada modelo de sociedad constituye un lenguaje para describir comunidades virtuales a partir de él. Así por ejemplo, para crear la comunidad del un colegio se utilizan estudiantes y profesores como tipos de usuarios válidos y clases y grupos como tipos de colectivos.

Es necesario advertir que esta labor constructiva no siempre se realiza completamente desde cero para cada nueva experiencia de aprendizaje sino que, típicamente, pasa por sucesivos estadios de desarrollo. En concreto, en el caso de experiencias de educación formal, es posible distinguir tres fases asociadas a los tres tipos de grupos identificados por [Johnson & Johnson, 1994]:

- **Definición de la estructura base.** La definición de la estructura base se realiza, típicamente, en la fase inicial de la implantación de métodos pedagógicos colaborativos en una institución educativa. Aquí se trata de definir la estructura de los elementos inmutables de la misma tales como los actores participantes (director, profesor, estudiante, etc.) y las agrupaciones constituyentes (departamentos, claustro, etc.).
- **Definición de la estructura formal.** La definición de la estructura formal de los grupos se realiza típicamente al principio de cada curso académico para definir los colectivos de usuarios que se mantendrán durante el mismo. Típicamente, nos estamos refiriendo a las clases de cada promoción escolar formadas generalmente por un profesor y un colectivo de estudiantes.
- **Definición de la estructura informal.** La definición de la estructura informal se refiere a cambios frecuentes que se realizan para dar soporte a las experiencias colaborativas. Por ejemplo, en el método de Jigsaw es necesario crear grupos expertos con una duración temporal breve.

Esta división pone de manifiesto el carácter inherentemente dinámico de la construcción de comunidades y de los modelos de sociedad. En efecto, estos últimos son productos que se construyen paulatinamente y que van evolucionando a lo largo del tiempo extendiéndose para dar cobertura a otros tipos de agrupaciones. Las prestaciones de este subsistema ponen de manifiesto algunas ventajas relevantes que suponen un valor diferencial con respecto a otras propuestas existentes, desde modelos teóricos como [IMS–EM] hasta plataformas reales como [Rodriguez & Verdejo, 2004] [Verdejo et al., 2002] [Blackboard] [Moodle] o [Alf]:

- **Independencia entre las tareas de diseño y desarrollo.** Con la separación en dos fases se consigue independencia en la realización de las tareas de modelado social, a nivel de diseño, y las de construcción y mantenimiento de las comunidades virtuales específicas que se ajustan a dicho modelo, a nivel de desarrollo. Esto hace posible que diferentes equipos, diseñadores instruccionales por un lado y administradores por otro, dediquen sus esfuerzos en paralelo a tareas diferentes.
- **Obtención de 2 tipos de productos complementarios.** Las tareas de estas dos fases generan sendos tipos de productos distintos pero interdependientes: la especificación de un modelo de sociedad y la de las comunidades virtuales específicas que se adscriben a él. Los modelos constituyen un artefacto que permite definir y entender cuál es la estructura organizativa de una comunidad virtual y cada comunidad virtual mantiene información acerca de cómo cada colectivo de usuarios se organiza en torno a las restricciones estructurales impuestas por el modelo.

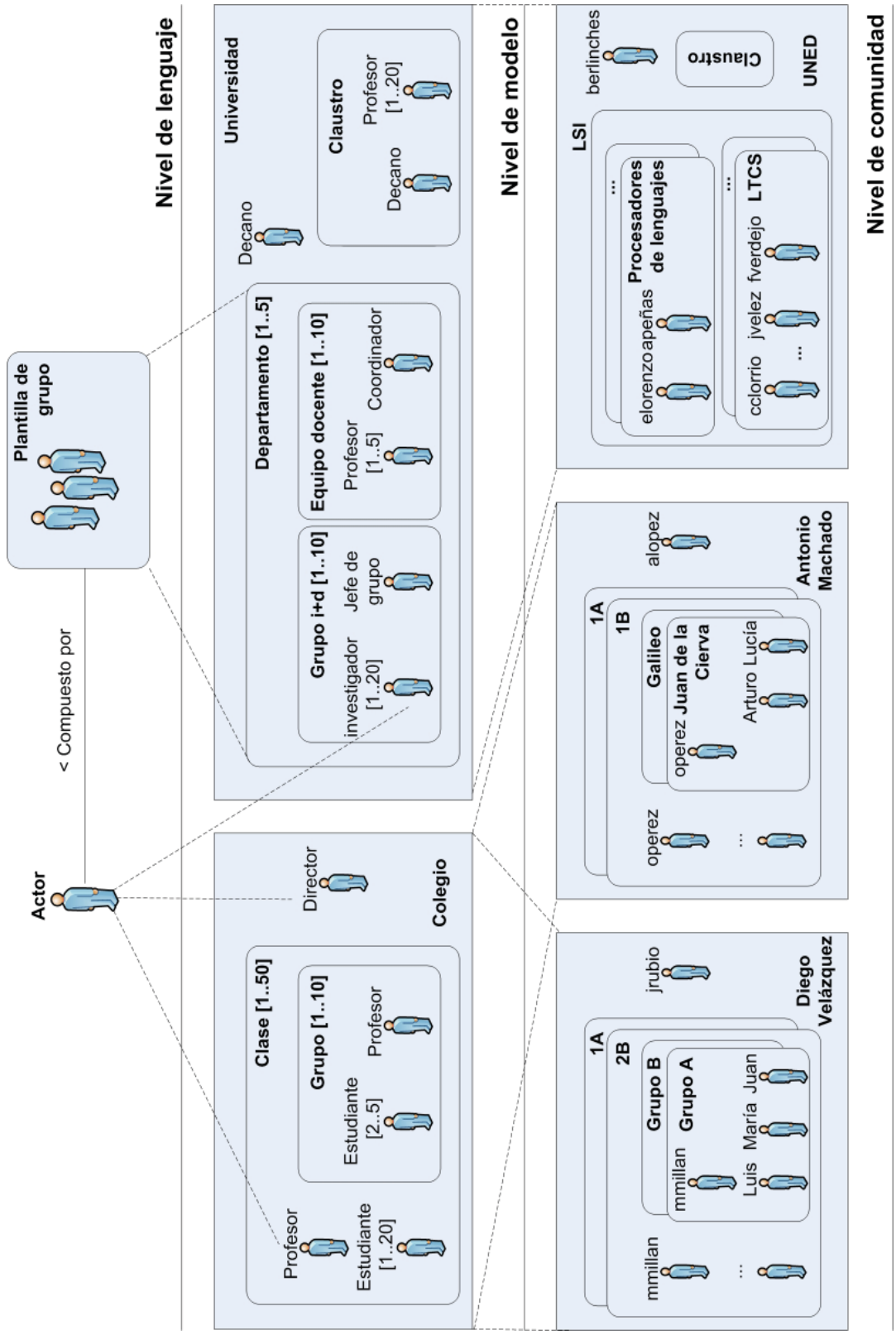

Figura 3.2. Niveles de especificación social en Pelican.

- **Reutilización de los modelos sociales.** Como consecuencia del carácter independiente de sendos productos, aparece una nueva ventaja. Los modelos de sociedad se convierten en especificaciones potencialmente reutilizables que capturan requisitos sociales necesarios para la realización de un tipo de escenario de aprendizaje colaborativo. Por ejemplo, es bien conocido que el método pedagógico Jigsaw requiere la existencia, aunque no simultanea, de grupos de trabajo y grupos de expertos formados los segundos a partir de una reorganización de los primeros. Los aspectos estáticos de la especificación social pueden incluir el tamaño y composición interna de estos dos tipos de colectivos.

- **Centralización de la administración social.** La existencia de un modelo social que describa, en términos abstractos, la estructura organizativa de una o varias comunidades virtuales permite ahorrar trabajo en las tareas de mantenimiento de las mismas. En efecto, un cambio en el modelo de sociedad se propagará de manera inmediata a todas las comunidades virtuales que se ajustan a él. Esto es una capacidad importante de la plataforma que permite gestionar el carácter evolutivo y dinámicamente cambiante de las estructuras sociales asociadas a las experiencias de trabajo colaborativo.

- **Independencia del modelo de sociedad.** La ventaja anterior se apoya en la existencia de una dependencia estructural entre el modelo de sociedad que describe una comunidad y la comunidad en sí misma. Sin embargo esta dependencia no es simétrica, lo cual permite mantener el modelo de sociedad independiente de cualquier contexto socio–pedagógico donde se utilice. De acuerdo a esto, es posible realizar cambios dentro de una comunidad virtual alterando su composición especifica (usuarios reales que la constituyen, colectivos específicos que forman estos usuarios, etc.) sin necesidad de modificar el modelo de sociedad y por tanto manteniendo intacta la organización estructural de otras comunidad virtuales que sigan el mismo modelo.

3.3.1. El subsistema social

Una vez presentadas las funcionalidades del subsistema social, podemos entrar a describir cómo éstas son proporcionadas desde un punto de vista más técnico. En concreto, el subsistema social de Pelican está formado por un sencillo modelo que conjuga varios artefactos para dar soporte tanto al diseño de modelos de sociedad como a la construcción y mantenimiento de comunidades virtuales asociadas a los mismos [Vélez et al. 2005], [Vélez et al., 2006].

Con relación al diseño de modelos, pueden destacarse varias entidades que vienen a representar, a nivel computable, los elementos conceptuales del lenguaje de modelado. A continuación describimos en detalle cada uno de ellos:

- **Actor.** Los actores permiten representar los diferentes arquetipos de usuarios que aparecen en una comunidad virtual. Este artefacto sirve para ofrecer a los usuarios una ubicación precisa dentro de la misma y conferirles una serie de responsabilidades y competencias sociales. En el caso del modelo de sociedad para

colegios que discutimos anteriormente en la figura 3.2, los actores son el director, el profesor y el estudiante. Dentro de este modelo, cada uno de ellos tiene una responsabilidad diferente y está confinado a un contexto social distinto. Por ejemplo, los estudiantes se implican en actividades de aprendizaje colaborativas e interactúan con otros estudiantes de igual a igual mientras que el profesor es el encargado de gestionar el tiempo y los recursos para el desarrollo de las mismas. Hay que recordar que en nuestro modelo teórico identificamos como participantes del proceso instruccional a cinco actores (administradores, diseñadores, profesores, monitores y estudiantes). Estos han sido, son y serán utilizados a lo largo de este texto para articular nuestro discurso teórico. No obstante, la idea que pretendemos transmitir aquí es que Pelican ofrece la libertad a los diseñadores instruccionales de determinar los actores que estimen más oportunos para definir sus escenarios colaborativos, mediante el uso de este artefacto.

- **Permission.** Como acabamos de describir, el propósito de la entidad actor es conferir a los usuarios una serie de responsabilidades y competencias sociales dentro de la sociedad que se está modelando. Para dar soporte a estas responsabilidades, cada actor tiene asociados una colección de permisos que lo caracterizan semánticamente. Cada permiso, representado por la entidad Permission, encapsula un término clave que indica la concesión de cierto nivel de acceso sobre alguno de los elementos gestionados por Pelican. En concreto, el vocabulario de permisos básicos que puede asignarse a un actor contempla todos los posibles niveles de acceso a las interfaces y funcionalidades de la plataforma. La tabla 3.1 resume este vocabulario. En este sentido, volviendo al ejemplo de la figura 3.2, puede ser interesante prescribir que los estudiantes sólo tengan acceso al área Web donde se desarrollan sus actividades colaborativas, mientras que los profesores, adicionalmente, puedan acceder a la parte de gestión de la sociedad para dar de alta nuevos usuarios y grupos. Nuevamente queremos enfatizar aquí el hecho de que es responsabilidad de los diseñadores instruccionales determinar la caracterización, en términos de permisos, que se hará para cada actor. Aunque la colección de permisos es un vocabulario inicialmente cerrado, éste puede extenderse con nuevos tipos de permisos para gestionar aspectos propios de los escenarios que se estén desarrollando haciendo uso de las facilidades del subsistema de intervención como se verá en la sección 3.5.

- **Group Template.** Las plantillas de grupo de un modelo de sociedad sirven para identificar los tipos de grupos que aparecen potencialmente en una comunidad virtual y prescriben ciertas características estructurales sobre los mismos. En concreto, este artefacto indica qué actores – esto es, tipos de usuarios – formarán parte de este tipo de grupos y qué otros tipos de grupos aparecerán como posibles subgrupos del mismo. A esta información también le acompaña la cardinalidad mínima y máxima permitida para ambos tipos de elementos. Las plantillas de grupo se utilizan en tiempo de desarrollo para construir grupos de usuarios reales e imponer las restricciones estructurales allí definidas. En este sentido, una plantilla puede entenderse como un molde que permite al administrador crear

grupos de usuarios con unas características estructurales similares. Así un modelo de sociedad en Pelican se estructura de acuerdo a una organización jerárquica de plantillas de grupo donde existe un única plantilla raíz para representar a la comunidad (el colegio en el ejemplo precedente) y cada plantilla incluye actores y, opcionalmente otras plantillas hijas (clases con profesores, estudiantes y grupos).

	Permiso	Descripción
Subsistema social (3.3)	actorDesign.Add	Permite crear nuevos actores
	actorDesign.Delete	Permite borrar actores
	actorDesign.Update	Permite actualizar la información sobre actores
	actorDesign.View	Permite acceder a la interfaz Web para el diseño de actores
	groupTemplates.Add	Permite crear nuevas plantillas de grupo
	groupTemplates.Delete	Permite borrar plantillas de grupo
	groupTemplates.Update	Permite actualizar la información sobre las plantillas de grupo
	groupTemplates.View	Permite acceder a la interfaz Web para el diseño de plantillas de grupo
	groupManagement.Add	Permite crear grupos
	groupManagement.Delete	Permite borrar grupos
	groupManagement.Update	Permite actualizar la información sobre los grupos
	groupManagement.View	Permite acceder a la interfaz web para la administración de grupos
	userManagement.Add	Permite registrar usuarios
	userManagement.Delete	Permite borrar usuarios
	userManagement.Update	Permite actualizar el perfil de los usuarios
	userManagement.View	Permite acceder a la interfaz Web para la administración de usuarios
Subsistema de colaboración (3.4)	projectDesign.Add	Permite crear plantillas de proyecto
	projectDesign.Delete	Permite borrar plantillas de proyecto
	projectDesign.Update	Permite actualizar la información sobre las plantillas de proyecto
	projectDesign.View	Permite acceder a la interfaz Web para el diseño de plantillas de proyecto
	activityDesign.Add	Permite crear plantillas de actividad
	activityDesign.Delete	Permite borrar plantillas de actividad
	activityDesign.Update	Permite actualizar la información sobre las plantillas de actividad
	activityDesign.View	Permite acceder a la interfaz Web para el diseño de plantillas de actividad
	projectManagement.Add	Permite desplegar proyectos sobre los espacios de trabajo
	projectManagement.Delete	Permite replegar proyectos de los espacios de trabajo
	projectManagement.Update	Permite configurar el estado y nivel de visibilidad y acceso de los proyectos
	projectManagement.View	Permite acceder a la interfaz Web para la administración de proyectos
	activityManagement.Add	Permite desplegar actividades sobre los espacios de trabajo
	activityManagement.Delete	Permite replegar actividades de los espacios de trabajo
	activityManagement.Update	Permite alterar el ciclo de vida y el nivel de visibilidad y acceso de las actividades
	activityManagement.View	Permite acceder a la interfaz Web para la administración de actividades
	activity.ChangeRoles	Permite cambiar los roles de una actividad
	activity.ChangeState	Permite cambiar el estado de una actividad
	activity.changeState.DevelopmentToPending	Permite realizar la transición de actividad entre Initial y Pending
	activity.changeState.InitialToDevelopment	Permite realizar la transición de actividad entre Initial y Development
	activity.changeState.PendingToFailed	Permite realizar la transición de actividad entre Pending y Failed
	activity.changeState.PendingToPassed	Permite realizar la transición de actividad entre Pending y Passed
	activity.changeState.PendingToReview	Permite realizar la transición de actividad entre Pending y Review
	activity.changeState.ReviewToPassed	Permite realizar la transición de actividad entre Review y Passed
	activity.changeState.ReviewToPending	Permite realizar la transición de actividad entre Review y Pending
Subsistema intervención (3.5)	policyScriptManagement.Add	Permite crear scripts de adaptación
	policyScriptManagement.Delete	Permite borrar scripts de adaptación
	policyScriptManagement.Update	Permite actualizar la información y el código de los scripts de adaptación
	policyScriptManagement.View	Permite acceder a la interfaz Web para la administración de scripts de adaptación
	policyManagement.Add	Permite crear nuevas reglas políticas
	policyManagement.Delete	Permite borrar reglas políticas
	policyManagement.Update	Permite actualizar la información sobre las reglas políticas
	policyManagement.View	Permite acceder a la interfaz Web para la administración de reglas políticas

Tabla 3.1. Vocabulario de permisos asociados a la entidad Actor.

Estos tres elementos permiten dar soporte al lenguaje de modelado social de la plataforma que sirve para construir modelos de sociedad. Sin embargo, para hacer frente a la construcción y mantenimiento de comunidades virtuales es necesario incorporar otras entidades en este subsistema. Dado que la construcción de comunidades está dirigida en Pelican por los elementos de modelado anteriores, no será de extrañar que exista una estrecha relación entre los primeros y los segundos. A continuación describimos en detalla cada uno de ellos:

- **User.** Cada usuario registrado en la plataforma Pelican queda representado por una instancia de usuario. Este artefacto sirve para identificar a los usuarios que hay dentro del sistema y a lo largo de sus sesiones de trabajo. Además, el sistema almacena un perfil de usuario que contiene varios tipos de informaciones:
  - *Información de autentificación.* La información de autentificación está formada por un login y una clave que son proporcionados al usuario por el administrador que lo registra en la plataforma. Estos datos son de obligada cumplimentación ya que se utilizan para autorizar la entrada en el sistema.
  - *Información personal.* La información personal del usuario es un conjunto de atributos que representan ciertas características de relevancia para las tareas de gestión y, aunque no son obligatorios para registrar usuarios, proporcionarlos resulta conveniente. En concreto, el modelo contempla el nombre completo de usuario, su dirección postal completa, su número de teléfono, su correo electrónico, su fecha de nacimiento y su género (ver figura 3.13 ❸). Como veremos más adelante, disponer de estos datos puede ser muy conveniente ya que el subsistema de intervención puede utilizarlos para automatizar tareas de gestión relacionadas con el desarrollo del flujo instruccional. Por ejemplo, es posible mandar un correo electrónico a algún alumno si no está haciendo un seguimiento de las actividades en curso. Similarmente, el subsistema de integración puede utilizar ocasionalmente estos datos para imponer condiciones de acceso a las herramientas integradas en el entorno. Por ejemplo, permitir que sólo las mujeres de un grupo accedan a una herramienta o que se perciba su uso de forma diferente en función de la edad.
  - *Información sobre el actor por defecto.* Como discutiremos a continuación cada usuario puede encarnar diversos actores en el mismo o diferentes grupos aunque no de forma simultanea. El actor por defecto, indica al sistema qué actor encarnará este usuario nada más acceder a la plataforma. Esta es una propiedad de especificación obligatoria del perfil del usuario que, como las anteriores, son indicadas por el administrador cuando registra al mismo.
  - *Información de contexto.* La información de contexto consiste en una colección de atributos que mantienen el estado del contexto socio–colaborativo del usuario dentro de la plataforma (grupo en curso, actividad colaborativa en curso, compañeros conectados, etc.). Dado que esta información es explotada por el subsistema de intervención postergaremos su descripción detallada para la sección 3.5, donde abordaremos dicho subsistema.

- **Group.** Para crear la comunidad virtual, los administradores de la plataforma deben, además de registrar a todos los usuarios en el sistema, crear un grupo para cada uno de los colectivos que éstos van a formar. Esta es una tarea de mantenimiento que se desarrolla a lo largo de todo el tiempo de vida de la comunidad ya que muchos grupos de usuarios tienen un carácter eminentemente dinámico e informal y se van creando y destruyendo bajo demanda de las necesidades puntuales. Como se comentó anteriormente, para crear grupos en Pelican debe utilizarse una plantilla de grupo previamente defina lo cual obliga a respetar la estructura jerárquica del modelo de sociedad al que se ajusta la comunidad. En el ejemplo de la figura 3.2, se han creado 2 comunidades que siguen el modelo de colegio. Como se ve, crear un colegio requiere utilizar la plantilla *colegio*. Una vez creada ésta, se pueden crear grupos dentro de la misma con la plantilla *clase*, pero sin superar el máximo impuesto de 50. De forma similar, dentro de cada clase podemos crear nuevos grupos utilizando en esta ocasión la plantilla *grupo* hasta un máximo de 10, pero por ejemplo no es posible crear grupos hijos de una clase utilizando la plantilla *clase* o *colegio,* ya que supone una violación del modelo de sociedad predefinido. La plantilla se encarga de velar por el cumplimiento de este tipo de restricciones estructurales.

- **Actor Binding.** Una vez creados los grupos en la comunidad virtual es posible enrolar a los usuarios en los mismos. Dado que el subsistema social de Pelican está basado en actores, esta tarea requiere indicar qué actor encarnará cada usuario dentro del grupo. Para dar soporte a esta información, que vincula a cada actor de cada grupo con los usuarios que lo encarnan, se utiliza la entidad Actor Binding. Esta entidad flexibiliza la estructura social de una comunidad virtual ya que permite que un usuario pueda encarnar a diferentes actores en diferentes grupos y dentro del mismo grupo. Por ejemplo, un usuario podría actuar de estudiante en un grupo y de monitor en otro. Este modelado es interesante si se pretende articular experiencias tales como la descrita en el método de Jigsaw, según el cual, el estudiante pertenece a distintos grupos y juega un papel diferente en cada uno de ellos (experto o estudiante), si bien, como veremos más adelante, otros artefactos del subsistema colaborativo permiten modelizaciones alternativas para este escenario. En el ejemplo de la comunidad Diego Velázquez de la figura 3.2, el grupo A está formado por Luis, María y Juan que participan encarnando al actor *estudiante* y mmillan que aparece encarnando al actor *profesor*. En Pelican la tarea de enrolado de usuarios en grupos está pensada para que sea desarrollada por los administradores de la plataforma ya que en los escenarios de educación formal este es el requerimiento más frecuente. Sin embargo, nada impide que, con las capacidades proporcionadas por el subsistema de intervención, se implanten protocolos de acceso para permitir, por ejemplo, a un usuario solicitar la admisión en un grupo y delegar la decisión de aceptación en una actividad de votación que involucra a los actuales miembros del mismo para que después Pelican procese el resultado de la votación y enrole si procede al usuario solicitante de forma automática. Hay que advertir además que la asignación de actores a usuarios permite referenciar de manera unívoca a todos los usuarios que encarnan un determinado actor. Esta

es una característica que se puede aprovechar desde los subsistemas de intervención e integración para definir comportamientos adaptativos sobre determinados tipos de usuarios o expresar condiciones de uso de las herramientas que dependan del tipo de usuario que interactúe con ella (estudiante, profesor, etc.).

El diagrama conceptual en UML del subsistema social de Pelican que aparece representado en la figura 3.3, ilustra cómo se conjugan las entidades del nivel de modelo con aquellas que permiten dar soporte a la comunidad. En línea con lo que fue descrito anteriormente puede verse que cada plantilla de grupo (Group Template) está compuesta por una colección de plantillas de grupos hijas a través de una relación de parentalidad. Además, esta plantilla incluye, como parte de su definición, una colección de actores. La cardinalidad 0 a n en ambos extremos de la relación indica, en efecto, que una plantilla puede incluir varios actores y también que cada actor puede aparecer formando parte de otras plantillas de grupo diferentes. La entidad actor (Actor) contiene una relación de permisos (Permission) que pertenecen con exclusividad a éste y que como dijimos sirven para caracterizarlo semánticamente dentro de la plataforma.

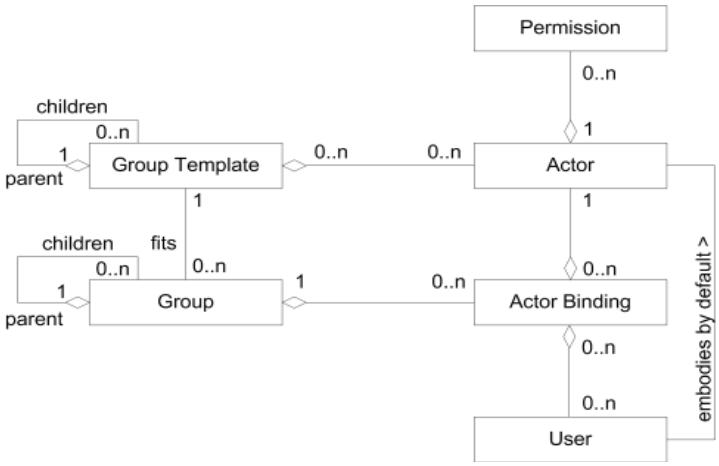

Figura 3.3. Diagrama conceptual en UML del subsistema social.

Con relación a los artefactos del nivel de comunidad puede verse que cada grupo (Group) se ajusta a una plantilla de grupo (Group Template) que lo describe estructuralmente. En efecto, la relación de ajuste (fits) indica que cada grupo queda vinculado exactamente a una plantilla de grupo y que cada plantilla de grupo puede ser utilizada para definir la estructura interna de varios grupos. La existencia de esta relación fuerza a que la organización jerárquica de plantillas de grupo propia del modelo de sociedad sea conferida, de manera natural, a la estructura comunitaria. En efecto, si las plantillas de grupo se utilizan para construir grupos y cada plantilla potencialmente incluye a otras plantillas hijas, entonces los grupos contendrán como parte de ellos a otros grupos hijos que a su vez quedarán descritos por las plantillas hijas del nivel de modelo. Esto justifica que la relación de parentalidad también aparezca presente sobre la entidad Group. Además un grupo mantiene una asociación compositiva con un conjunto de entidades Actor Binding. Esta entidad es en realidad la representación

de una relación ternaria que vincula un grupo, un actor y un conjunto de usuarios que lo encarnan. En efecto, como puede verse en la figura, cada actor binding tiene asociado exactamente un actor y cada actor puede aparecer en varios actor binding. Cada actor binding pertenece con exclusividad a un grupo pero un grupo puede tener varios actor binding y, finalmente, cada actor binding incluye potencialmente a varios usuarios, aunque los usuarios pueden aparecer formando parte de distintos actor binding, asociados al mismo o diferentes grupos. Por último, la entidad usuario (User), que como dijimos se utiliza para gestionar el perfil de los usuarios dados de alta en la plataforma, incluye una relación que sirve para mantener el actor que un usuario adquiere por defecto cada vez que se autentifica en Pelican. Por supuesto, en cuanto éste entra en un grupo, el actor que pasa a encarnar cambia de acuerdo a lo establecido dentro de las entidades actor binding asociadas a dicho grupo.

3.3.2. La interfaz del subsistema social

Para dar soporte al diseño, gestión y desarrollo de experiencias de aprendizaje colaborativo, Pelican proporciona una serie de sencillas interfaces Web que permiten a diferentes tipos de usuarios (administradores, diseñadores, profesores, estudiantes, etc.) interactuar con la misma. Su descripción se distribuye a lo largo de este capítulo dentro de cada una de las cuatro secciones donde se abordan los subsistemas de la plataforma. En concreto, en esta subsección presentaremos la interfaz del subsistema social. No obstante, antes de entrar de lleno en ello, conviene comentar algunas cualidades comunes a todas estas interfaces:

- **Separación de las interfaces en diferentes áreas.** Las interfaces de Pelican pueden pertenecer a una de las tres área definidas en la plataforma. Cada una de éstas da acceso a una colección de funcionalidades relacionadas entre sí y vinculadas con algún tipo de tarea. En concreto estas áreas son:
  - *Área de diseño.* El área de diseño reúne todas las interfaces proporcionadas por los subsistemas de la plataforma relacionadas con las tareas de diseño de los escenarios.
  - *Área de administración.* En el área de administración aparecen todas las interfaces de usuario destinadas a realizar tareas de gestión y mantenimiento de las experiencias de aprendizaje colaborativo que están siendo desarrolladas en cada momento dentro del entorno.
  - *Área de trabajo.* El área de trabajo está formada por los espacios de trabajo, proporcionados por el subsistema de colaboración y constituye el lugar dentro del entorno donde toda la actividad colaborativa tiene lugar.
- **Estructura general de las pantallas.** Independientemente del área donde estén inscritas las interfaces de la plataforma, todas ellas disponen de una misma estructura. En concreto pueden identificarse cuatro partes dentro de la misma (ver, por ejemplo, figura 3.4). Arriba aparece una cabecera que incluye a su derecha información sobre el usuario al que pertenece la sesión de trabajo, la fecha y hora

en que ésta comenzó y un botón para abandonar la misma, así como dos iconos para cambiar el idioma de las interfaces de ingles a español o viceversa. En la parte de la izquierda aparece un menú vertical que recoge todas las opciones que ese usuario tiene accesibles. Básicamente es una relación de enlaces de acceso a las diferentes interfaces de la plataforma organizados en las tres áreas anteriormente descritas. En la parte inferior aparece un pie de página con enlaces al grupo donde se ha desarrollado este producto y, finalmente, la parte central es un espacio reservado para albergar el contenido de cada interfaz específica y que será descrito en detalle para cada una de ellas en sucesivas subsecciones.

Interfaz para el diseño de actores

Como expusimos con anterioridad uno de los primeros pasos para definir un modelo de sociedad es identificar todos los tipos de usuarios de la comunidad y especificarlos como actores dentro de la plataforma. Para llevar a cabo esta labor se utiliza la interfaz presentada en la figura 3.4. Como puede apreciarse, a la izquierda (❶) aparece la relación de todos los actores definidos. Seleccionando cada uno de los elementos de esa lista se pasa a editar, sobre el interfaz de la derecha, la información sobre dicho actor. Para alterar esta lista, creando nuevos actores o borrando los ya existentes, utilizamos los iconos de eliminación y creación (❷). Cuando pulsamos sobre el botón de eliminación desaparece el actor en curso, iluminado en negrilla sobre la lista. Cuando pulsamos sobre el botón de creación aparece un nuevo actor en la lista con un nombre temporal que más adelante podremos cambiar.

El formulario de edición de actores permite, en primer lugar asignar un nombre apropiado a cada actor e incorporar una descripción literal sobre las responsabilidades y competencias generales del mismo (❸). La especificación de estas responsabilidades, a nivel computable, se refina indicando sobre el formulario el conjunto de permisos que se desea conceder al actor (❹). Como puede apreciarse en la figura, existe una caja de selección para cada uno de los permisos que fueron descritos en la tabla 3.1. Como se describió en dicha tabla, la alteración de estos permisos de actor condiciona considerablemente la percepción que tendrán los usuarios sobre la plataforma. Por ejemplo el menú principal de la izquierda que da acceso a otras áreas de la plataforma se ve modificado cuando no están seleccionados los permisos pertinentes. En el ejemplo de la figura, se ha presentado una configuración típica de permisos para el estudiante. Éste solo tendrá acceso a las interfaces relacionadas con su área de trabajo y no a ninguna relativa a las tareas de diseño o administración ya que resulta potencialmente peligroso que un estudiante pueda adquirir estos permisos.

Hay que advertir que, por construcción, cada vez que la plataforma se instala, se crea por defecto un actor administrador (manager) que contiene todos los permisos de acceso. Esto es necesario para permitir al usuario administrador comenzar a configurar el entorno. Además, debajo del formulario de edición, esta interfaz presenta una relación con todas las plantillas de actividad donde este actor aparece (❹). Pinchando sobre ellas accederemos al interfaz para el diseño de plantillas de grupo que describimos a continuación.

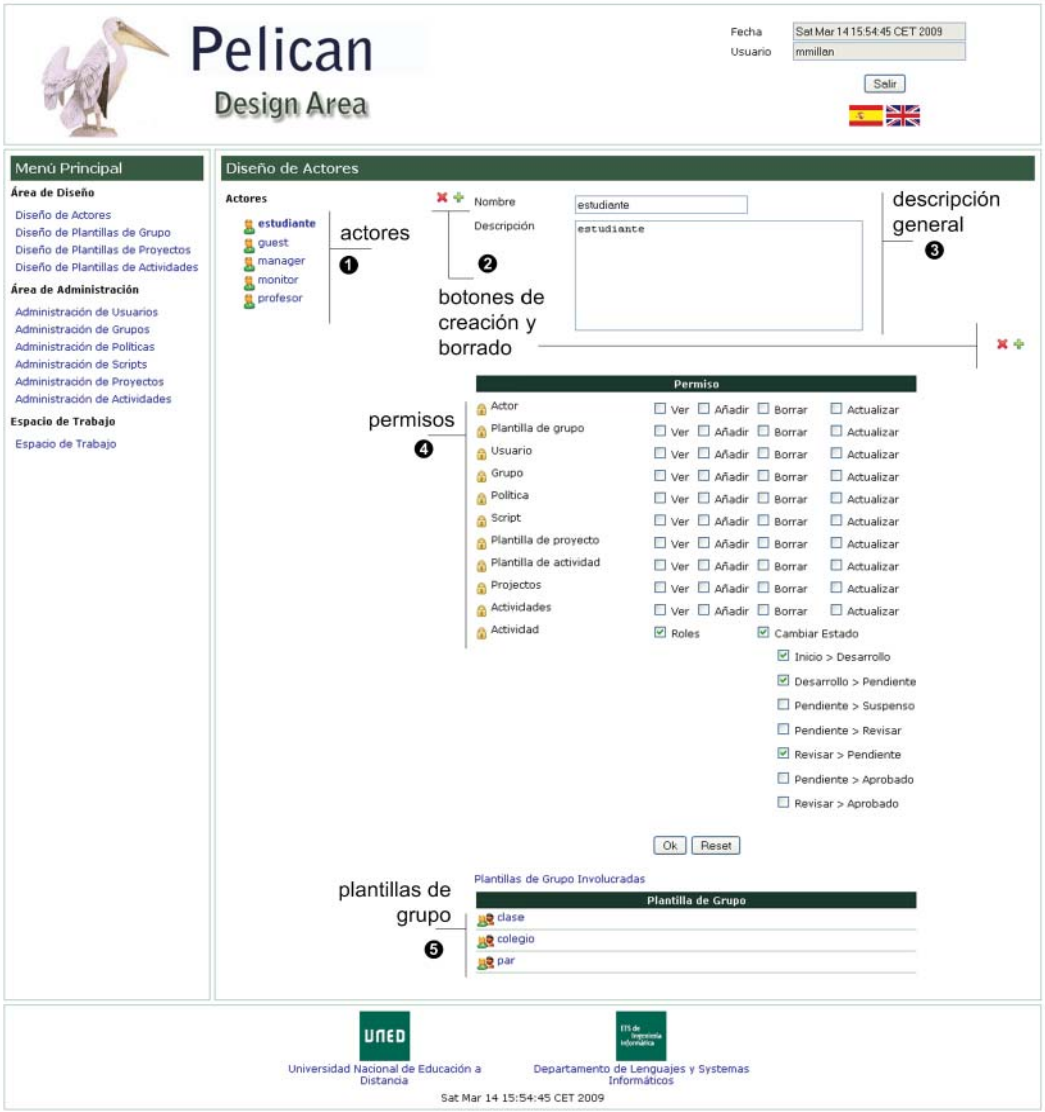

Figura 3.4. Interfaz para el diseño de actores.

Interfaz para el diseño de plantillas de grupo

Una vez definidos los actores que participan en una comunidad virtual, es posible identificar los diferentes tipos de colectivos de usuarios que van a formar parte de la misma y definir para cada uno de ellos una plantilla de grupo. Para ello, la plataforma proporciona la interfaz de diseño que aparece en la figura 3.5. Como puede verse, a la izquierda aparece un árbol que representa la organización jerárquica de todas las plantillas definidas (❶). En el caso de la figura, se ha definido como plantilla raíz un colegio que está formado por colectivos de profesores, monitores y clases. Y cada clase, a su vez incluye un tipo de agrupamiento llamado par que representa un grupo de dos estudiantes. El diseñador puede moverse por esta estructura arborescente para editar, en el formulario de la derecha las propiedades de la plantilla de grupo deseada. La plantilla seleccionada aparece resaltada en negrilla.

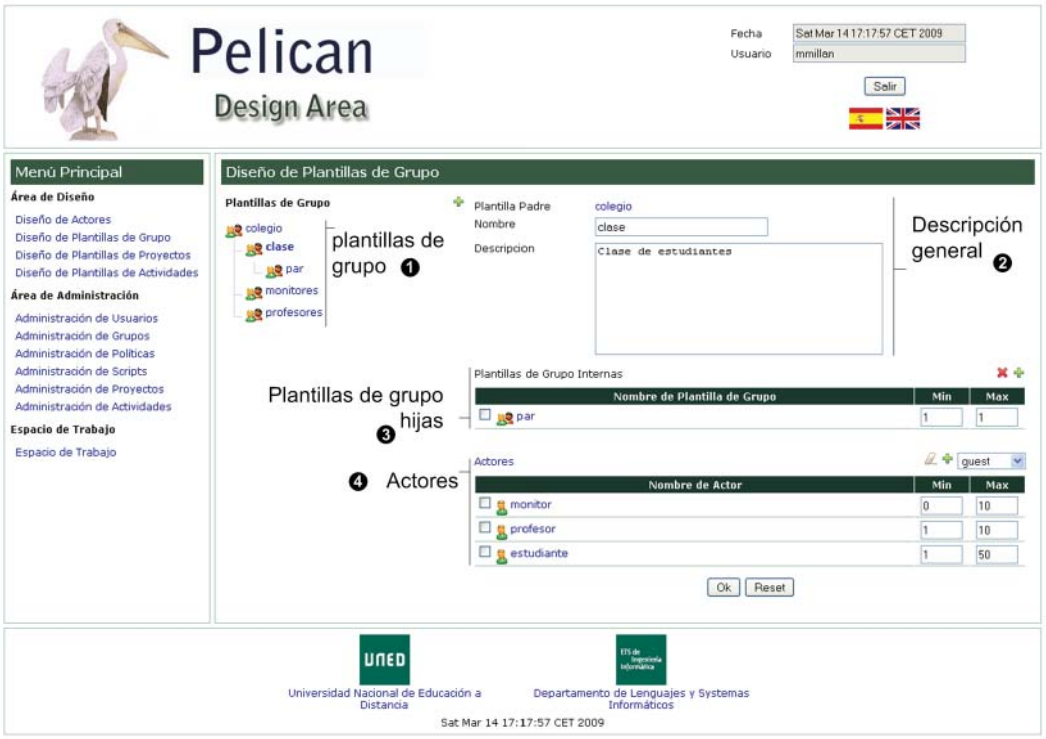

Figura 3.5. Interfaz para el diseño de plantillas de grupo.

El formulario de edición permite, en primer lugar, asignar un nombre a la plantilla, que servirá para identificarlo dentro del sistema así como incluir una descripción y acceder a su plantilla padre (❷).Debajo se proporciona una tabla con todas las plantillas hijas definidas dentro de ella (❸). Cada entrada de la tabla incluye el nombre de la plantilla y dos campos para indicar la cardinalidad mínima y máxima permitidas. Pinchando sobre los nombres se salta a editar la plantilla de grupo hija seleccionada. Encima de la tabla, a la derecha, aparecen dos botones para permitir crear y borrar plantillas de grupo. Para borrar una plantilla de grupo es necesario seleccionar el cuadro a la izquierda de la entrada en la tabla y pulsar sobre el botón en forma de aspa. Para crear una nueva plantilla, se pincha el botón de adición y aparecerá una nueva entrada en la tabla. Los datos relativos a la cardinalidad pueden modificarse sobre la misma tabla y después pinchar sobre el nombre de la plantilla nueva para editarla y pasar a especificar su estructura interna. Debajo de las tablas hijas aparece otra tabla que muestra la relación de actores incorporados a la plantilla junto con su información de cardinalidad (❹). Estos datos pueden modificarse sobre la tabla. Si pinchamos sobre el nombre de un actor saltaremos al interfaz de diseño de actores. Los botones de encima de la tabla, a la derecha, permiten quitar y añadir actores a la plantilla. Para quitar un actor lo seleccionamos con el cuadro de selección que aparece a la izquierda de su nombre y pulsamos sobre el botón de borrado. Para añadir un nuevo actor debemos seleccionar uno de ellos de la lista desplegable y presionar el botón de añadir. Esto provocará la aparición de una nueva entrada con valores de cardinalidad por defecto que pueden modificarse sobre la tabla.

Interfaz para la administración de usuarios

Una de las labores de los administradores para construir la comunidad virtual de aprendizaje consiste en dar de alta en el sistema a los usuarios de la misma. Esta tarea requiere rellenar, para cada nuevo usuario el perfil indicando los datos de autentificación, información personal y actor por defecto. Para realizar esto, la plataforma proporciona la interfaz de la figura 3.6.

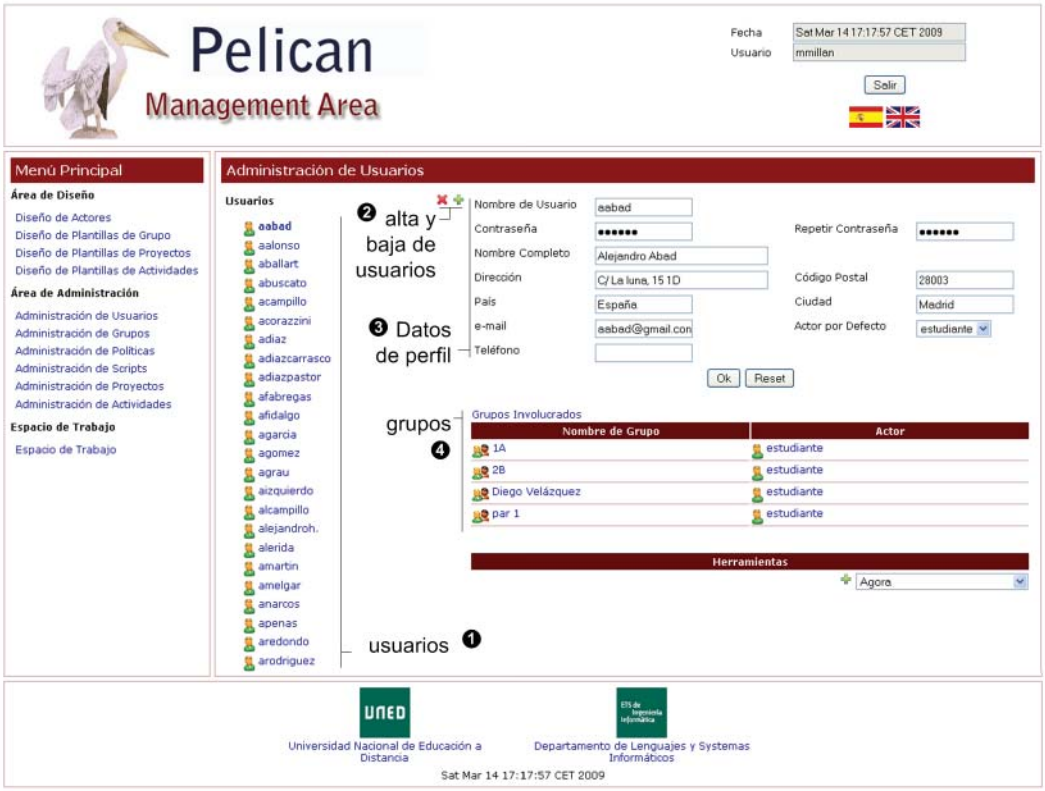

Figura 3.6. Interfaz para la administración de usuarios.

A la izquierda aparece la colección de usuarios dados de alta en la plataforma (❶). El administrador puede recorrer esta lista para seleccionar el usuario que desee editar en el formulario de la derecha. Este usuario aparecerá resaltado en negrilla. Para dar de alta nuevos usuarios y borrar los ya existentes se utilizan los botones de creación y borrado (❷). Al pulsar sobre el botón de borrado, el usuario que está seleccionado en la lista se elimina del sistema. Al pulsar sobre el botón de creación aparecerá un nuevo usuario seleccionado en la lista con un nombre temporal que puede ser editado con el formulario de la derecha (❸).

Este formulario permite incluir la información del perfil indicando el login y la contraseña, los datos personales y el tipo de actor por defecto, seleccionable a través de un desplegable donde aparecen todos los actores especificados en el modelo de sociedad. Debajo, aparece una relación de los grupos a los que el usuario pertenece (❹). Cada entrada de la tabla indica el nombre del grupo y el actor que encarna dentro

de dicho grupo. Al pinchar sobre el actor accedemos al interfaz de diseño de actores para editar aquél seleccionado. Al pinchar sobre el nombre del grupo saltamos a la interfaz de administración de grupos que describiremos a continuación.

Es necesario advertir que cuando la plataforma se instala, aparece un usuario administrador por defecto (manager) que tiene como actor por defecto aquél de su mismo nombre y que como dijimos antes tiene concedidos todos los permisos sobre la plataforma. Esto permite comenzar las tareas de administración iniciales entrando con usuario manager y clave manager.

Interfaz para la administración de grupos

De forma potencialmente simultanea al registro de usuarios, los administradores de la plataforma pueden crear nuevos grupos vacios a partir de alguna plantilla de grupo previamente definida y, posteriormente, poblarlos con usuarios registrados. La interfaz Web que aparece en la figura 3.7 permite a los administradores llevar a cabo estas dos tareas.

En la parte de la izquierda de la interfaz para la administración de grupos aparece un árbol que representa la estructura jerárquica de los grupos que conforman la comunidad virtual y cuya organización responde a aquélla prescrita en el modelo de sociedad (❶). En efecto, en el caso de la figura por ejemplo, se ha creado el colegio Diego Velázquez que está formado por un grupo de profesores (descrito por la plantilla profesores), un grupo de monitores (con plantilla monitores) y la colección de grupos 1A, 1B, 1C y 2B (todos ellos asociados a la plantilla clase). Cuando se selecciona uno de estos grupos sobre el árbol, este pasa a iluminarse en negrilla y se edita sobre el formulario adjunto.

El formulario de edición de grupos comienza por una sección donde es posible cambiar el nombre del grupo y describir el propósito general mismo de forma literal (❷). Debajo aparece una relación de todos los grupos que éste incluye como grupos hijos (❸). Como puede apreciarse, éstos se organizan según la plantilla de grupo que haya sido utilizada para crearlos. En el ejemplo de la figura adjunta, sólo existe una plantilla de grupo hija definida llamada *par* por lo que en este caso todos los grupos aparecen bajo ella. Si se pincha sobre el nombre de la plantilla se puede acceder al interfaz de diseño de plantillas de grupo para editar la plantilla seleccionada. Si pinchamos sobre el nombre de uno de los grupos de esta tabla pasamos a editarlo sobre este mismo interfaz.

Para crear y borrar grupos hijo se utilizan los botones de creación y borrado que aparecen a la derecha sobre cada entrada de la tabla que indica una plantilla de grupo. Para borrar un grupo debemos marcarlo en la tabla y pulsar el botón de borrado. Esta acción desencadena un borrado en cascada de todos los grupos hijo que a su vez el grupo borrado pudiera tener. Para crear un nuevo grupo pulsamos en el botón de creación asociado a la plantilla de grupo que deseamos usar. Esto generará una nueva entrada en la tabla con un nombre temporal que podremos cambiar pinchando sobre su nombre.

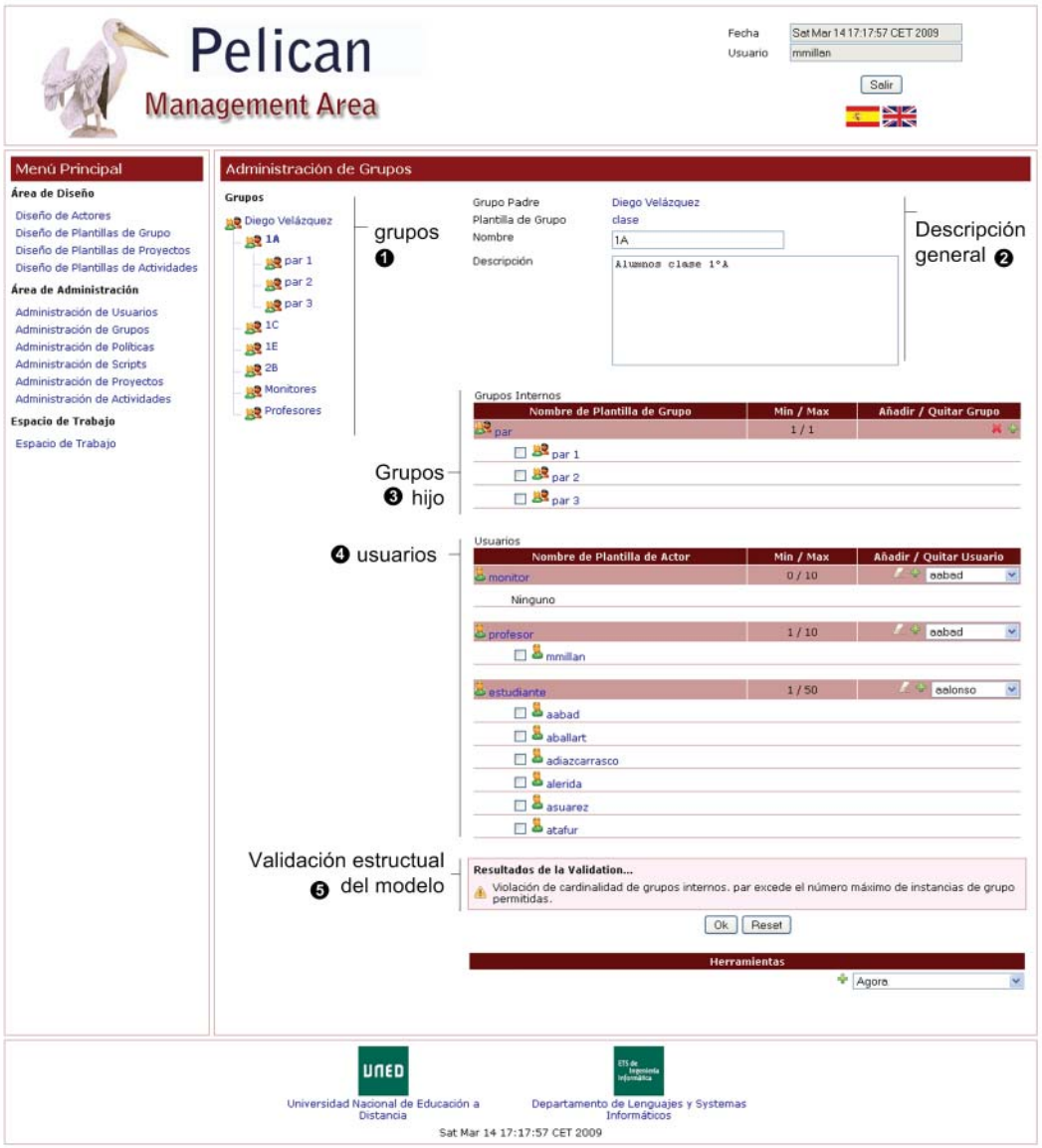

Figura 3.7. Interfaz para la administración de grupos.

Un mecanismo similar al anterior se utiliza para incluir o eliminar los usuarios del grupo que está siendo editado. En la tabla de actores (❹), aparece una relación de todos los usuarios que forman parte del grupo organizados según el o los actores se encarnan. En concreto, la tabla está divida en actores y debajo de cada actor se muestran los usuarios que aparecen vinculados al grupo encarnando dicho actor. Para borrar un usuario es necesario seleccionarlo en la tabla y pulsar sobre el botón de borrado correspondiente al actor que esta representando. Adviértase que un usuario puede aparecer encarnando varios actores por lo que la acciones de borrado tiene que aplicarse sobre el actor del que se desea eliminar al usuario. De forma similar, para incluir un nuevo usuario debe seleccionarse éste de la caja desplegable del actor deseado y pulsar sobre el botón de adición. En el ejemplo de la figura puede apreciarse cómo

aparece una entrada para cada actor definido en la plantilla de clase (profesor, monitor y estudiante). Ocasionalmente, al final del todo aparece una caja de información que alerta sobre las violaciones que se están cometiendo de acuerdo a las prescripciones del modelo de sociedad (❺). En el caso de la figura, la violación indica que el número de grupos hijo de tipo *par*, que es tres (par 1, par 2, par 3), excede la cardinalidad máxima permitida, que ha sido establecida a 1. En este caso, el error parece estar en la descripción del modelo ya que no parece sensato imponer que una clase pueda tener un sólo grupo par. Para solucionarlo es necesario volver a la interfaz de definición de la plantilla de grupo clase y corregir esta restricción de cardinalidad. En la mayoría de las ocasiones por el contrario, los errores detectados de esta forma corresponden a verdaderas violaciones de un modelo bien definido.

3.4. Soporte a la colaboración

En la sección anterior fueron descritas las capacidades del subsistema social de Pelican y se discutió cómo éstas pueden ser convenientemente utilizadas para crear un modelo de sociedad que prescriba las características estructurales que el entramado social subyacente donde se desarrolla la experiencia formativa debe presentar. Este estructuración es una parte fundamental de la definición de un escenario de aprendizaje. Sin embargo, la descripción del trabajo colaborativo que los estudiantes deben realizar es otro de los componentes esenciales de la misma.

Las facilidades proporcionadas por el subsistema de colaboración de Pelican que presentamos en esta sección van dirigidas a dar cobertura a la especificación de todos los aspectos estructurales y organizativos relacionados con la definición del trabajo colaborativo. Éstas se complementan con aquéllas proporcionadas por el subsistema de integración que, como veremos en la sección 3.6, completará la definición al permitir proporcionar las herramientas y servicios necesarios para desarrollar las experiencias de forma efectiva.

De manera similar a como ocurría en el subsistema social, en el subsistema de colaboración se pueden distinguir dos tipos de tareas diferentes que se realizan a lo largo de las fases de diseño y desarrollo respectivamente:

- **Diseño del trabajo colaborativo.** En la fase de diseño, los diseñadores instruccionales hacen uso de las capacidades de este subsistema para definir la dimensión colaborativa de un escenario de aprendizaje. Es decir, prescriben todo el trabajo colaborativo e individual que los estudiantes deben realizan cuando llevan a cabo la experiencia. Esta especificación se hace en términos abstractos, independientes del contexto social específico al que vaya dirigida la experiencia haciendo uso del lenguaje de modelado que, a este respecto, proporciona la plataforma.

- **Desarrollo del trabajo colaborativo.** Una vez que se ha realizado una especificación del trabajo colaborativo del escenario, ésta puede desplegarse en diferentes contextos sociales – esto es, grupos soportados por el subsistema social – para que sea llevada a cabo. La plataforma proporciona, en este sentido, toda la infraestructura tecnológica necesaria para permitir el desarrollo.

El lenguaje de modelado proporcionado por el subsistema de colaboración permite articular la especificación del trabajo de los estudiantes en términos de plantillas de proyecto y plantillas de actividad. Una **plantilla de proyecto** representa la especificación colaborativa completa de un escenario. Cada plantilla de proyecto está compuesta, básicamente, por un conjunto de **plantillas de actividad** que describen las actividades que deben llevar a cabo los estudiantes. El orden de desarrollo de las mismas no puede especificarse con las capacidades del subsistema de colaboración. Éste es considerado un aspecto dinámico de diseño que puede ser descrito haciendo uso de las prestaciones del subsistema de intervención (ver sección 3.5). Nótese que no todos los escenarios colaborativos prescriben un orden de ejecución de actividades. En algunos casos, el orden de desarrollo e incluso las propias actividades a desarrollar puede ser determinadas por los estudiantes según avanzan en la experiencia.

Por su parte, el desarrollo del trabajo colaborativo queda soportado dentro de la plataforma, principalmente, por los **espacios de trabajo**. Un espacio de trabajo es un entorno virtual donde tiene lugar toda la construcción colaborativa – acción e interacción de los estudiantes – circunscrita al desarrollo de las experiencias de aprendizaje. En concreto, es posible distinguir dos tipos de espacios de trabajo:

- **Espacios de trabajo privados.** Los espacios de trabajo privados pertenecen, de manera individual, a cada uno de los usuarios registrados en la plataforma. De esta manera, cada usuario de Pelican dispone de un espacio de trabajo privado al que sólo él tiene acceso. Dentro de estos espacios suele tener lugar el desarrollo de tareas individuales que hayan surgido fruto de la descomposición de una actividad en tareas más sencillas que implican a usa sola persona, de acuerdo a un esquema cooperativo de división del trabajo.

- **Espacios de trabajo compartidos.** Por su parte, los espacios de trabajo compartidos, están asociados a un grupo de los soportados por el subsistema social. Así, todo grupo de la comunidad virtual de aprendizaje dispone de un espacio de trabajo compartido al que tienen acceso todos sus miembros (y es visible solamente por ellos). En este tipo de espacios tiene lugar el desarrollo de las actividades colaborativas que implican a los miembros del mismo.

De los dos tipos de espacios de trabajo, los segundos resultan los más utilizados ya que, naturalmente, las experiencias colaborativas requieren desarrollarse en un entorno virtual compartido que fomente la interacción. A este respecto, cabe destacar que estos espacios adquieren una estructuración jerárquica conferida por la organización social de la comunidad virtual. En efecto, si cada grupo dispone, por construcción de un espacio de trabajo, y éstos se organizan de forma arborescente en el subsistema social, los espacios de trabajo también se organizarán de tal forma. Esto permite hablar de niveles o capas sociales sobre los que, potencialmente, se desarrollan las experiencias de aprendizaje. En el modelo de colegio que se ilustró en la figura 3.2, puede verse cómo en concreto, para cada plantilla de grupo, existe un nivel social (colegio, clase y grupo) y al mismo nivel social se ubican todos los espacios de trabajo asociados con grupos de una misma plantilla de grupo.

En la figura 3.8 se representa esta organización jerárquica para el caso de la comunidad Diego Velázquez. En el nivel más alto se encuentra el espacio de trabajo del colegio. Por debajo, a nivel de clase, encontramos los espacios de los grupos clase 1A y 2B. Y, finalmente, a nivel de grupo, encontramos los espacios de trabajo para los grupos A y B de la clase 2B y C y D de la clase 1A, aunque estos dos últimos no aparecían representados en la figura 3.2. Adicionalmente puede considerarse que los espacios de trabajo privados se sitúan en el nivel social inferior por debajo de todos los anteriores.

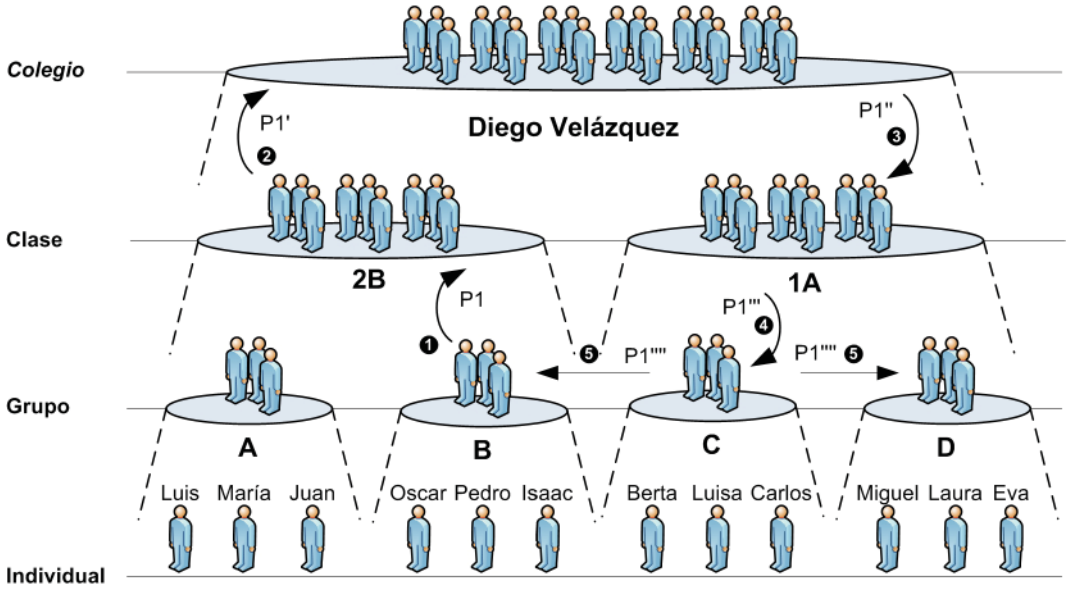

Figura 3.8. Espacios de trabajo organizados a distintos niveles sociales.

Como describimos en el capítulo anterior, algunos autores han apuntado que el desarrollo de las experiencias de aprendizaje colaborativo se extiende, a menudo, a lo largo de diferentes contextos sociales ubicados a distinto nivel social [Dillenbourg, 2002] [Dillenbourg, 2004]. Así, por ejemplo, un proyecto de aprendizaje puede comenzar con una actividad inicial planteada a nivel de clase, continuar con actividades que son realizadas a nivel de grupo, o a nivel individual y terminar con una actividad final nuevamente desarrollada a nivel de clase. Los productos generados por unas actividades serán utilizados, potencialmente, como entrada de otras actividades subsiguientes, aunque éstas se desarrollen a un nivel social diferente.

De esta manera, las interacciones colaborativas que se desarrollan en Pelican para realizar las experiencias de aprendizaje pueden ser de dos tipos:

- **Interacciones colaborativas intra–grupales.** Este tipo de interacciones son aquéllas que se producen entre los miembros de un mismo grupo cuando participan en un espacio de trabajo compartido como consecuencia del desarrollo de una actividad de aprendizaje que les implica únicamente a ellos. Muchos de los métodos pedagógicos descritos en el capítulo 2, como por ejemplo los escenarios propuestos por [Johnson & Johnson, 1994] son de esta naturaleza.

- **Interacciones colaborativas inter–grupales.** Las interacciones inter–grupales son aquellas que implican a los miembros de varios espacios de trabajo diferentes. Este tipo de interacciones están basadas en la compartición de productos entre los usuarios de cada grupo. Es decir, un grupo genera uno o varios productos que entrega a otro u otros grupos para que lo utilicen o modifiquen. Estas transferencias implican potenciales transformaciones del producto a cada paso que pueden desarrollarse en dos direcciones:

  – *Transferencia horizontal.* La transferencia horizontal se produce cuando los productos de un grupo son transferidos, horizontalmente, a otro grupo que se encuentra en su mismo nivel social. Este esquema colaborativo se utiliza en algunos métodos pedagógicos como en la estructura *Send a problem* de [Kagan & Kagan, 1994] donde un problema se va resolviendo en sucesivas iteraciones al ser revisado por todos los grupos de la clase.

  – *Transferencia transversal.* La transferencia transversal implica movimientos verticales, ascendentes y descendentes, [Dillenbourg, 2002] desde un espacio de trabajo a otros situados un nivel por encima o por debajo de éste respectivamente. La ventaja de este tipo de movimientos es que respeta la estructura organizativa de la comunidad, lo cual a veces es interesante para imponer un uso y revisión jerarquizado de los productos. De esta manera, cuando un grupo de un nivel termina un producto, lo entrega a su nivel inmediatamente superior para que lo revise y transforme si lo considera oportuno. Este procederá escalándolo nuevamente hacia arriba si desea moverlo a espacios de trabajo hermanos o situados en un nivel superior o descendentemente si desea propagarlo hacia otros espacios de trabajo hijos. La idea subyacente aquí es que cada nivel dentro de la jerarquía de espacios de trabajo supone un grado de autoridad que se encarga de regular las tareas del nivel inferior. Esto corresponde a una división del trabajo horizontal propia de escenarios colaborativos [Dillenbourg, 1996]. Sin embargo, hay que advertir que esto no es más que una interpretación particular del carácter jerárquico de la estructura de espacios de trabajo en Pelican.

En la figura 3.8, se ilustra un recorrido de 5 pasos de transferencia que transforman un producto de forma sucesiva. En concreto, el producto P1 generado por el grupo B es transferido al espacio de la clase 2B donde se transforma para convertirse en P1'. Desde allí, se entrega P1' al colegio transformándolo en P1'' que lo entregará descendentemente a la clase 1A para convertirlo en P1''' y entregarlo a su destino, el grupo C. Este es un ejemplo de transferencia transversal. El grupo C por su parte decide modificarlo convirtiéndolo en P1'''' y entregarlo a los grupos D y B directamente a través de una transferencia horizontal.

Por supuesto que todos estos tipos de movimientos se realizan en la plataforma haciendo uso de herramientas de soporte a la interacción colaborativa potencialmente desarrolladas por terceras partes e integradas en la misma. Piénsese como ejemplo, en un repositorio de objetos de aprendizaje que residiera en el espacio de trabajo

del grupo de origen y destino de cada transferencia. Los movimientos consistirían, en este ejemplo, en almacenar y recuperar los productos de aquél repositorio al que ambos grupos tuvieran acceso. En este sentido, puede decirse que para articular esta idea de transferencia de forma completa en Pelican es necesario hacer uso de las prestaciones del subsistema de integración. Sin embargo, muchas veces el subsistema de intervención también es utilizado, en situaciones como estas, para articular transferencias de productos automáticamente entre espacios de trabajo.

La puesta en práctica de experiencias de aprendizaje colaborativo se lleva a cabo a partir de una tarea de administración propia de la fase de desarrollo. Una vez que un escenario ha sido diseñado en términos de una plantilla de proyecto y una colección de plantillas de actividad que lo componen, los administradores deben desplegar éste a lo largo del entramado jerárquico de espacios de trabajo. En concreto, éstos deben indicar qué plantillas de actividad serán desplegadas sobre cada uno de los espacios de trabajo implicados en el desarrollo de la experiencia. Al hacer esto, se crean proyectos y actividades contextualizados respecto a un grupo del subsistema social. De esta manera, cada **proyecto** vinculado a un espacio de trabajo surge al desplegar una plantilla de proyecto sobre dicho espacio. De forma similar, cada **actividad** que aparece en un espacio de trabajo se crea cuando se despliega una plantilla de actividad sobre el mismo.

El propósito de estos artefactos – proyectos y actividades – es el de mantener el estado de desarrollo de un determinada experiencia colaborativa, desplegada sobre un determinado espacio de trabajo. Para dar soporte a ello, este subsistema asocia a cada actividad un ciclo de vida que determina el estado en que ésta puede encontrarse desde su comienzo a su finalización. La figura 3.9 muestra un diagrama de dicho ciclo de vida donde puede apreciarse tanto los estados que atraviesa una actividad como las transiciones que se producen entre los mismos a lo largo de su desarrollo. A continuación pasamos a describir la semántica subyacente a cada uno de los estados del ciclo de vida:

- **Inicio.** Por construcción, cada vez que una plantilla de actividad se despliega sobre un espacio de trabajo, la actividad resultante adquiere el estado de *inicio*. Este estado identifica aquella situación en la que la actividad aún no ha comenzado a desarrollarse.

- **En desarrollo.** Cuando los estudiantes – o cualquier actor implicado en el desarrollo de la actividad – comienzan a trabajar, el estado de la misma pasa a ser *en desarrollo*. Una actividad en desarrollo es toda aquélla que está siendo llevada a cabo y que aún no ha sido revisada.

- **Pendiente de evaluación.** Cuando los estudiantes terminan de desarrollar la actividad y deciden que ésta ya está preparada para ser revisada, el estado de la misma cambia a *pendiente de evaluación*. Esto permite identificar aquellas actividades que deben ser revisadas, atendiendo a los productos generados en la misma para determinar si se han cumplido los objetivos pedagógicos perseguidos.

- **Aprobada.** Si el profesor – o cualquier actor con permisos – decide que la actividad que está evaluando ha alcanzado unos mínimos exigibles entonces la pasa al estado de *aprobada.* Éste es un estado final, de manera que cuando la actividad lo alcanza, su desarrollo se considera finalizado y los estudiantes pueden pasar a realizar la actividad subsiguiente dentro del flujo de trabajo instruccional.
- **En revisión.** Si el profesor determina que los resultados de la actividad son insuficientes entonces puede ponerse el estado de la misma a *en revisión*, lo que indicará a los estudiantes que deben revisarla, corregir los errores y volver a dejarla pendiente de evaluación.
- **Suspensa.** Si la actividad contiene errores y el profesor considera que ya se han realizado demasiados ciclos de revisión sin una mejora significativa en los resultados, entonces puede poner la actividad en estado *suspensa* para que los estudiantes pasen a realizar otra actividad. Este es el otro estado final del ciclo de vida ya que, una vez alcanzado, no es posible continuar con el desarrollo de la misma.

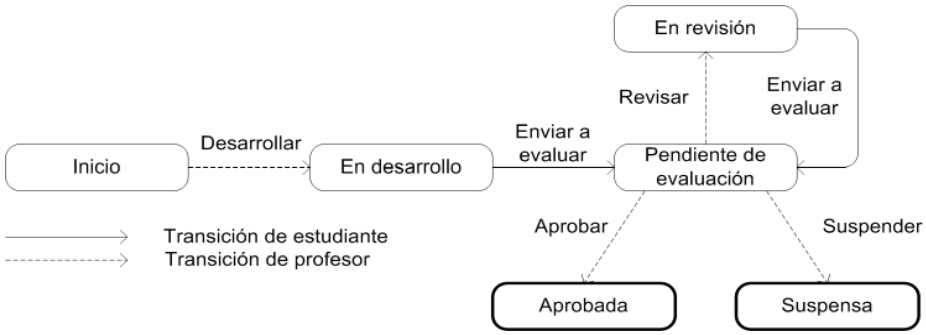

Figura 3.9. Ciclo de vida de la actividad.

El ciclo de vida de las actividades de Pelican permite dar asistencia a las tareas de monitorización y evaluación de la actividad colaborativa. En efecto, los estados de las actividades hacen posible llevar a cabo un seguimiento del desarrollo de las mismas dentro de un grupo tanto por monitores como por profesores y establece un medio de comunicación indirecto entre éstos y los estudiantes al permitir indicar el estado de las mismas tras su evaluación.

En este sentido, el cambio de estado de una actividad no es algo que la plataforma realice automáticamente. Muy al contrario, se trata de acciones de transición que los actores participantes de la experiencia deben realizar manualmente para indicar a los demás participantes, y al propio sistema, el estado actual de la misma. Con respecto a los participantes, esto permite mantener sincronizadas la vista que, desde sus navegadores, tienen todos los miembros del grupo sobre la actividad. En relación al sistema, cada cambio de estado es atendido, potencialmente, por el subsistema de intervención para aplicar políticas adaptativas si procede, tal y como se estudiará en la sección 3.5. La pregunta que cabe hacerse ahora es ¿sobre quién recae la responsabilidad de hacer las transiciones entre estados? Lo más habitual en los escenarios formales es conceder la transición desde el estado *inicio* hasta el estado *en desarrollo*

y desde el estado *pendiente de evaluación* a los estados *aprobada*, *suspensa* o *en revisión* al profesor y permitir realizar a los estudiantes las transiciones desde el estado *en desarrollo* o *en revisión* a *pendiente de evaluación*, tal y como aparece en la figura 3.9. Sin embargo, dado que el objetivo de la plataforma es ofrecer un mecanismo genérico para la especificación de escenarios y que los actores de éstos no están prefijados sino establecidos por diseño, se considera que la asignación de la responsabilidad en las transiciones entre los estados de la actividad es un tipo más de permisos que caracteriza la definición semántica de un actor dentro del modelo de sociedad. En efecto, como puede verse en la tabla 3.1 y en la interfaz de la figura 3.4, la especificación de los permisos relativos a una actividad incluye aquéllos relacionados con las transiciones de estado de la misma.

Con respecto a los proyectos, éstos también dan soporte a la gestión del estado de los mismos pero de una manera pretendidamente más general. En lugar de proporcionar un ciclo de vida, contienen un atributo que permite indicar el estado del proyecto. Lo más frecuente es que sea el profesor el encargado de proporcionar manualmente dicho estado. No obstante, dado que parece razonable pensar que el estado de un proyecto sea una función del estado de todas las actividades que lo constituyen, nada impide que se utilicen los mecanismos de automatización del subsistema de intervención para mantener actualizado este atributo en todo momento.

Según lo expuesto, el trabajo que ha de realizarse para especificar y desarrollar experiencias de aprendizaje en su dimensión colaborativa dentro de la plataforma Pelican, también puede considerarse dividido en niveles de especificación tal como ocurría en el subsistema social. Nuevamente, en este caso, podemos distinguir entre:

- **Nivel de lenguaje.** A este nivel, la plataforma proporciona un lenguaje de modelado para describir la dimensión colaborativa de un escenario de aprendizaje. Este lenguaje está constituido por los artefactos plantilla de proyecto y plantilla de actividad, que son utilizados para definir experiencias instruccionales y permiten organizar el trabajo a desarrollar por los estudiantes en estos mismos términos.

- **Nivel de modelo.** En este nivel, los diseñadores instruccionales utilizan el lenguaje de modelado colaborativo para crear un **modelo de colaboración** abstracto que define experiencias de aprendizaje de manera independiente al contexto social donde éstas vayan a desarrollarse. Cada plantilla de proyecto y actividad específicas de un modelo de colaboración son instancias de los artefactos del nivel de lenguaje y constituyen, a su vez, un estereotipo de proyecto y actividad que podrá ser reutilizado y desplegado en diferentes espacios de trabajo.

- **Nivel de despliegue.** A este último nivel, el modelo de colaboración establecido en el nivel anterior, y más particularmente, sus plantillas de proyecto y actividad, son desplegadas sobre uno o varios espacios de trabajo lo que provocará la creación de un **despliegue de trabajo colaborativo** formado por un proyecto y varias actividades en cada uno de los espacios de trabajo involucrados. A este nivel los estudiantes realizan las actividades y éstas van registrando el proceso de desarrollo mediante su estado interno.

La figura 3.10 ilustra esta estratificación en tres niveles de especificación para el caso de una experiencia de aprendizaje sobre las leyes de electricidad y magnetismo. Como puede apreciarse, a nivel de lenguaje aparecen los artefactos plantilla de proyecto y plantilla de actividad con una relación agregativa de las primeras para con las segundas. El modelo del ejemplo de la figura contiene una plantilla de proyecto denominada electricidad y magnetismo, que incluye a las plantillas de actividad ley de ohm y leyes e Kirchhoff. Finalmente, a nivel de despliegue, el modelo es desplegado sobre los espacios de trabajo de los grupos A y B, que aparecían en la figura 3.8. Allí cada actividad implica a un colectivo de estudiantes diferentes.

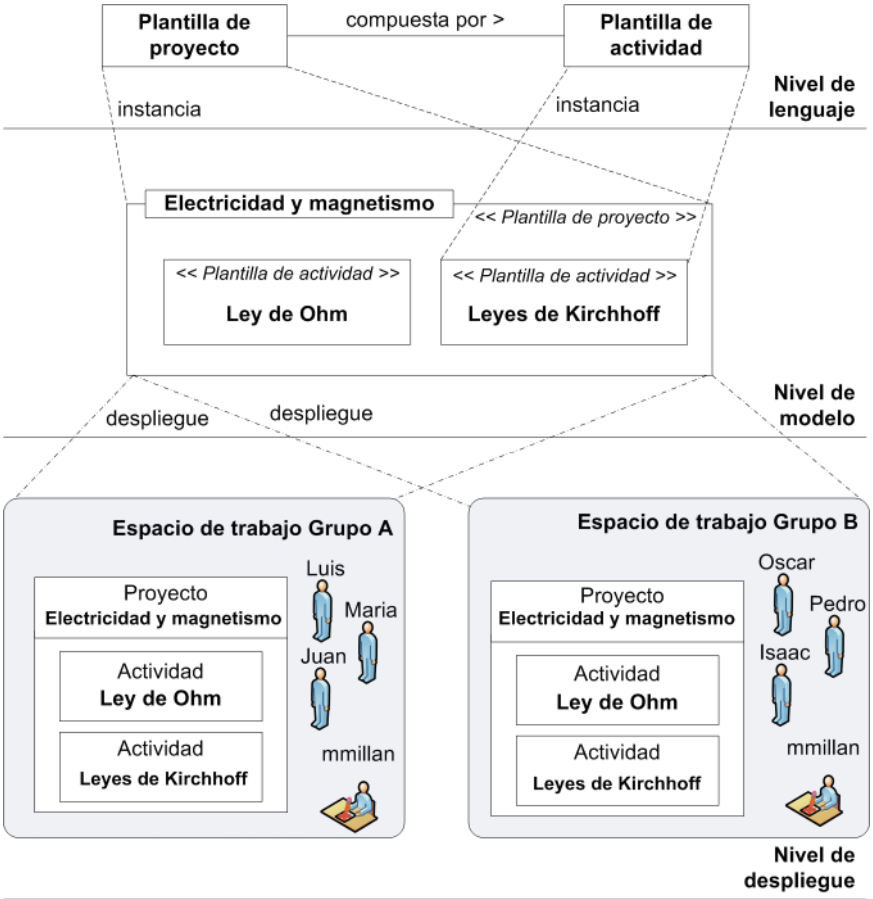

Figura 3.10. Niveles de especificación de la colaboración en Pelican.

Las tareas de diseño y desarrollo de la dimensión colaborativa de un escenario, suelen llevarse a cabo en la práctica de manera iterativa e incremental. Al principio se realiza un diseño preliminar de la experiencia colaborativa y se despliega ésta sobre los espacios de trabajo cuyos grupos vayan a realizarla. Tras recibir la retroalimentación acerca de cómo responden los estudiantes a la estructuración inicial se procede a un refinamiento del diseño que, generalmente, consiste en una redefinición del proyecto, basada fundamentalmente en la inserción, eliminación, fusión o redefinición de ciertas actividades de aprendizaje. Después, se espera a recibir una nueva retroalimentación y se comienza una nueva fase de diseño para la misma y así sucesi-

vamente. El modelo de colaboración de Pelican permite que los cambios producidos en el diseño se apliquen de manera automática a todos los proyectos ya desplegados sin comprometer la integridad de la experiencia.

Las prestaciones ofrecidas por el subsistema de colaboración de Pelican presentan varias ventajas destacables, algunas de las cuales resultan similares a aquéllas que fueron descritas en el subsistema social. A continuación describimos las mas relevantes:

- **Independencia entre las tareas de diseño y desarrollo.** Al igual que en el subsistema social, aquí el trabajo de los diseñadores para crear el modelo de colaboración y el de los administradores para desplegarlo resulta no solapante con lo que el trabajo en ambas fases puede hacerse de forma independiente.
- **Obtención de 2 tipos de productos complementarios.** En el subsistema de colaboración los dos productos resultantes son por un lado el modelo de colaboración, formado por una plantilla de proyecto y una colección de plantillas de actividad y por otro el trabajo desplegado en los espacios de trabajo, formado por un proyecto y un conjunto de actividades. Los segundos están asociados a los primeros en tanto que éstos son una representación, sobre un contexto social especifico, de los primeros. Sin embargo los primeros son productos independientes.
- **Reutilización de los modelos de colaboración.** El carácter independiente de los modelos de colaboración permite que éstos puedan reutilizarse en diferentes contextos sociales y situaciones pedagógicas. De hecho, lo más habitual es que los diseñadores instruccionales creen especificaciones de escenarios de aprendizaje y almacenen los modelos de colaboración resultantes en Pelican para obtener así un compendio de escenarios de aprendizaje colaborativo que podrán ser desplegados y adaptados en el futuro.
- **Despliegue de un modelo sobre múltiples espacios de trabajo.** Pelican permite que los modelos de colaboración sean desplegados sobre múltiples espacios de trabajo simultáneamente. En concreto podemos destacar, en este sentido, dos usos interesante de esta capacidad:
  - *Desplegar un modelo transversalmente a varios niveles sociales.* Como se describió en detalle con anterioridad, en Pelican es posible indicar dentro de qué espacio de trabajo deseamos que se despliegue cada plantilla de actividad. Dado que cada espacio de trabajo se encuentra ubicado a un determinado nivel social, los diseñadores pueden aprovechar esto para definir actividades y prescribir en qué nivel deben desarrollarse éstas (individual, grupo, clase, etc). Asimismo deberá indicarse cómo debe producirse la comunicación inter–grupal lo cual es posible haciendo uso de las prestaciones del sistema de intervención, como se verá en la sección 3.5.
  - *Desplegar un modelo simultáneamente en varios espacios de trabajo.* Frecuentemente los modelos de colaboración realizados por los diseñadores son desplegados de forma simultanea en varios espacios de trabajo, típicamente

asociados a algunos o todos los grupos colaborativos creados en un aula para desarrollar la experiencia. La independencia de proyectos y actividades para cada espacio de trabajo permite realizan un seguimiento del proceso instructivo de cada grupo independientemente de los demás. En efecto, dado que los proyectos y actividades dan soporte al mantenimiento del estado de desarrollo de la experiencia dentro de un grupo, es posible personalizar las estrategias de evaluación y monitorización que se aplicarán a éste.

- **Centralización de la administración colaborativa.** Al disponer, para cada escenario, de un único modelo que centraliza toda la especificación del trabajo colaborativo que deben realizar los estudiantes, se aligeran considerablemente las tareas de administración relativas al mantenimiento y actualización de las experiencias dentro de cada espacio de trabajo. En efecto, los cambios que realicemos sobre las plantillas de proyecto y actividad del modelo de colaboración se transferirán de manera inmediata a cada uno de los espacios de trabajo actualizando sus proyectos y actividades correspondientes.

3.4.1. El subsistema de colaboración

Para dar soporte a las capacidades y características que fueron descritas anteriormente, el subsistema de colaboración de Pelican está provisto de una colección de artefactos interrelacionados entre sí. A lo largo de esta subsección describiremos en detalle cada uno de ellos [Vélez & Verdejo, 2007] [Velez et al., 2006] [Vélez et al, 2005].

En concreto, éstos pueden clasificarse en dos categorías: aquéllos que permiten dar soporte al lenguaje de modelado colaborativo de la plataforma, y que se usan en la fase de diseño, y aquéllos que proporcionan la infraestructura necesaria para desarrollar las experiencias de aprendizaje en un determinado contexto social. A continuación dedicamos nuestra atención a los primeros:

- **Project Template.** Las plantillas de proyecto tienen un propósito organizativo, dentro del lenguaje de modelado de la plataforma, permitiendo especificar la colección de actividades formativas que conforman una experiencia. En este sentido puede decirse, que en su dimensión colaborativa, la especificación de un escenario de aprendizaje queda representada por una instancia de este tipo de artefacto. Internamente, las plantillas de proyecto están constituidas por un nombre y una descripción. El nombre permite asignar un título al escenario mientras que la descripción suele utilizarse para detallar literalmente los objetivos del proyecto y presentar un resumen del trabajo que debe ser realizado para alcanzarlos. El texto de estos campos va dirigido a los estudiantes, ya que éste aparecerá sobre el espacio de trabajo una vez realizado el despliegue. Las plantillas de proyecto incluyen además una colección de plantillas de actividad que, como veremos, detallan, a nivel computacional, el trabajo de la experiencia. Esta colección es una relación no ordenada de elementos y no se incluyen mecanismos para describir la lógica de secuenciamiento de los mismos. Si se desea anexar a la especificación un flujo de trabajo instruccional específico que active y desactive las actividades en fun-

ción de condiciones tales como el estado de compleción de las mismas o de sus fechas de comienzo y fin pueden utilizarse para ello las facilidades del subsistema de intervención. Nótese que éste es un aspecto dinámico que no aparece inherentemente vinculado a la especificación de cualquier escenario. En efecto, hay experiencias abiertas donde los alumnos tienen la libertad de escoger el orden de ejecución de las actividades y el carácter secuencial o paralelo de su desarrollo.

- **Activity Template.** Las plantillas de actividad describen una unidad de trabajo cohesionada que, como hemos comentado en el punto anterior, forman parte de la especificación de una determinada plantilla de proyecto. Esta definición se realiza, no obstante, de manera independiente de las plantillas de proyecto. Lo más frecuente es definir primero todas las plantillas de actividad de un escenario y luego construir una plantilla de proyecto que las agrupe. Esta independencia, fomenta la reutilización de estos artefactos, ya que nada impide que una plantilla de actividad aparezca formando parte de más de una plantilla de proyecto. Para describir una plantilla de actividad, los diseñadores instruccionales deben asignar un nombre y una descripción a la misma. De manera similar a como ocurría con las plantillas de proyecto, el nombre permite indicar el titulo de la actividad y la descripción contiene instrucciones de desarrollo para los estudiantes. Estos valores también serán presentados en el espacio de trabajo tras el despliegue. Adicionalmente, las plantillas de actividad incluyen una colección de servicios, roles y referencias que constituyen los elementos necesarios para desarrollar la misma. De ellos, los dos primeros, pertenecen al modelo de integración y serán discutidos en profundidad en la sección 3.6. Baste saber por el momento que permiten caracterizar la colección de herramientas que se requerirán en el desarrollo de la actividad y los roles que surgirán en el uso de las mismas. Las referencias por su parte son discutidas a continuación.

- **Reference.** Dentro de una plantilla de actividad, todos los elementos pedagógicos de información que deben ser proporcionados a los estudiantes para el desarrollo de la misma se proporcionan, en Pelican, a través de referencias Web. De esta manera, una referencia representa, dentro del lenguaje de modelado, un enlace a un recurso de la Web, tal como una página estática, una aplicación Web, un video, una imagen, un documento, un sonido, etc. Como tales, las referencias están constituidas internamente por un nombre y una descripción que es utilizado para presentar el enlace sobre el espacio de trabajo y una dirección URI a un recurso en la Web, que sirve para construir el vínculo html en el entorno.

Con estos elementos, los diseñadores instruccionales pueden especificar el modelo de colaboración de un escenario de aprendizaje. Como veremos más adelante, en la sección 3.6, esta especificación se completa con los artefactos del subsistema de integración, que permiten incluir herramientas externas en la misma. Sin embargo, estos elementos ya permiten definir escenarios de aprendizaje colaborativo y desarrollarlos en diferentes contextos sociales. A continuación describimos los elementos del subsistema de colaboración que dan soporte a este proceso de desarrollo:

- **Workspace.** El espacio de trabajo representa el entorno virtual dentro del cual se inscribe la interacción colaborativa relacionada con el desarrollo de las experiencias de aprendizaje. En el caso de los espacios de trabajo privados, esta entidad está vinculada a un usuario de la plataforma, mientras que en los espacios compartidos el vínculo es establecido con uno de los grupos soportados por el subsistema social. En este último caso, uno de los propósitos de los espacios de trabajo es proporcionar a cada miembro del grupo conciencia acerca del contexto social en que se circunscriben las experiencias de aprendizaje. En efecto, como veremos en la siguiente subsección, los miembros de un grupo pueden acceder al espacio de trabajo para ver quienes son sus compañeros y comprobar su estado de conexión en el momento actual. Dentro de los espacios de trabajo también podemos encontrar una colección de servicios que articulan la posibilidad de proporcionar acceso desde ellos a un conjunto de herramientas externas típicamente destinadas a dar soporte a la interacción colaborativa. No obstante, este es un aspecto de integración que será abordado más adelante en la sección 3.6. El otro elemento fundamental de los espacios de trabajo es la colección de proyectos desplegados dentro del mismo. Así, cuando una plantilla de proyecto es desplegada sobre un espacio de trabajo, se crea un nuevo proyecto y se añade a la colección de proyectos desplegados. En este sentido, los espacios de trabajo compartidos pueden ser contemplados como un portal de acceso a todas aquellas experiencias de aprendizaje que estén siendo desarrolladas por el grupo de trabajo asociado al mismo.

- **Project.** Los proyectos surgen como consecuencia de un proceso de despliegue de una plantilla de proyecto sobre un espacio de trabajo. De esta manera, para cada plantilla de proyecto y cada espacio de trabajo donde ésta se haya desplegado existe una instancia diferente de este elemento. Desde el punto de vista de su estructura interna, un proyecto está formado por tres elementos diferentes. En primer lugar la plantilla de proyecto. Todo proyecto mantiene una relación de uso con la plantilla de proyecto a la que corresponde. En efecto, los proyectos, para mostrarse sobre el espacio de trabajo, requieren información que pertenece a la especificación de la plantilla de proyecto asociada. De esta manera, como se explicó, los aspectos de especificación están centralizados en la plantilla de proyecto mientras que el proyecto mantiene la gestión de su desarrollo para un determinado grupo. Esto implica a su vez que los cambios que se hagan sobre la plantilla se verán reflejados de manera automática en todos los proyectos construidos a partir de ella. En segundo lugar, cada proyecto incluye una colección de todas aquellas actividades cuyas plantillas de actividad están contenidas dentro de la plantilla de proyecto y que hayan sido desplegadas sobre el espacio de trabajo. Y finalmente, los proyectos también mantienen un atributo estado para almacenar el estado de desarrollo del proyecto. No en vano, este es el fin último de este artefacto dentro del subsistema de colaboración, tal y como fue descrito con anterioridad.

- **Activity.** Una vez que una plantilla de proyecto ha sido desplegada sobre un determinado espacio de trabajo, es posible indicar a la plataforma concretamente qué plantillas de actividad desea desplegar sobre dicho espacio, de manera que para

éstas se creará una nueva actividad y se incluirá a la lista de actividades del proyecto. Así, para cada plantilla de actividad desplegada sobre un espacio de trabajo existirá una instancia de actividad asociada a dicho espacio y contenida dentro del proyecto. Las actividades están constituidas por cuatro elementos fundamentales. En primer lugar, la plantilla de actividad a la que corresponde. Similarmente al caso anterior, las plantillas de actividad centralizan cierta información como el nombre, la descripción, la colección de roles, referencias y servicios, que son utilizados por las actividades para dar soporte al desarrollo de las mismas. En segundo lugar, el estado actual de su desarrollo de acuerdo al ciclo de vida que describimos anteriormente. En tercer lugar, aparece la asignación temporal entre los roles definidos en la plantilla y los miembros del grupo. Y finalmente, se encuentra información relativa a la adaptación particular de los servicios proporcionados por las herramientas descritas en la plantilla para el contexto social asociado al espacio de trabajo donde se inscribe la actividad. Estos dos últimos elementos – asignación de roles y adaptación de servicios – están vinculados con las capacidades del subsistema de integración y serán descritos en la sección 3.6. En el último punto de esta relación abordaremos el estado de las actividades.

- **State.** La entidad estado sirve para representar el estado de desarrollo en el que se puede encontrar una determinada actividad en cada instante de tiempo. Cada posible valor de estado, queda representado por un subtipo diferente de este artefacto, que contiene una etiqueta identificativa y codifica la lógica de transición interna que aparece reflejada en la figura 3.9. A continuación describimos brevemente cada uno de los estados:

  – *Init.* La entidad Init representa el estado de inicio del ciclo de vida de una actividad. Por construcción cada actividad tiene una instancia de este elemento en su atributo estado. Desde Init la única transición posible es a In Progress.

  – *In Progress.* Esta entidad se corresponde con el estado *en desarrollo* del ciclo de vida. Desde él es posible transitar únicamente al estado Pending.

  – *Pending.* Pending es la entidad utilizada para representar el estado *pendiente de evaluación*. Desde este estado las transiciones posibles son hacia Passed, Failed o Review.

  – *Passed.* Passed representa el estado *aprobada* dentro del ciclo de vida de la actividad. Dado que se trata de un estado final, desde este estado no es posible hacer ninguna transición a otro estado.

  – *Failed.* Esta entidad representa el estado *suspensa* de una actividad. Como en el caso anterior, desde este estado no están permitidas las transiciones a otros estados ya que se trata de un estado final.

  – *Review.* Finalmente, review representa el estado *en revisión* que insta a los alumnos a que revisen una actividad. Desde este estado es posible transitar nuevamente al estado de Pending.

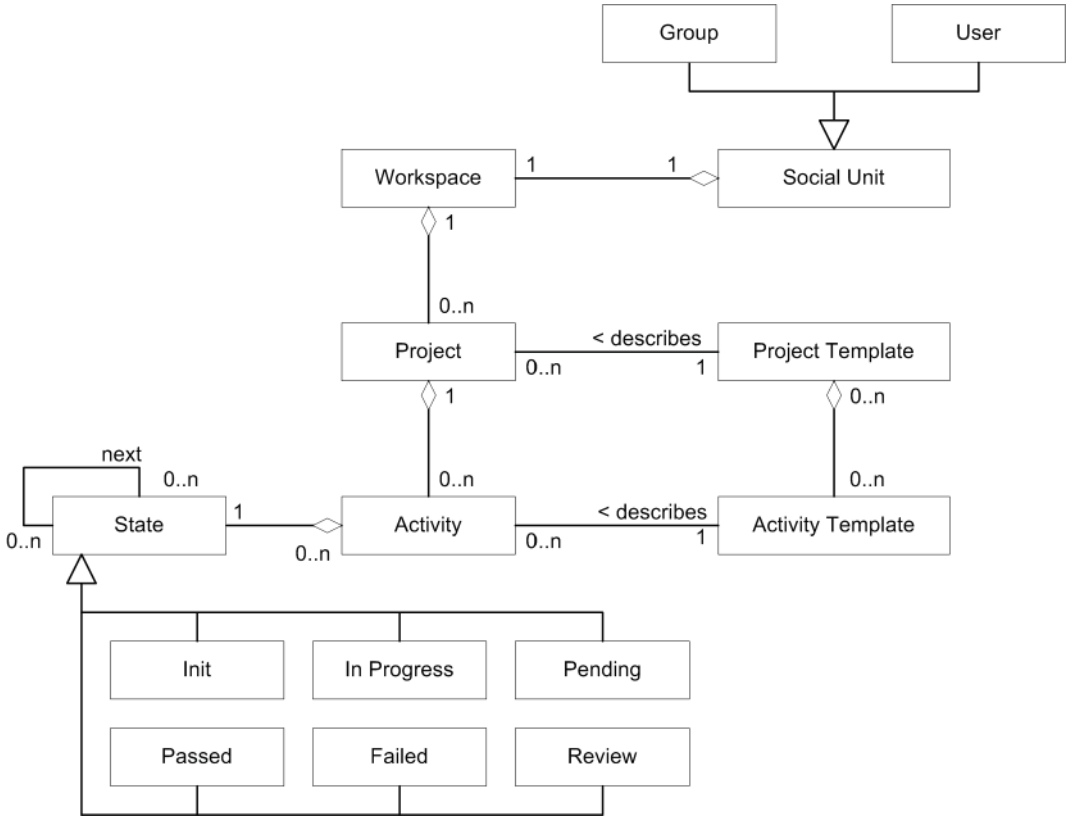

Figura 3.11. Modelo conceptual en UML del subsistema de colaboración.

Para terminal con la presentación del modelo interno utilizado por es subsistema de colaboración discutiremos la relación existente entre todos estos elementos a partir del diagrama conceptual en UML que aparece representado en la figura 3.11. El primer elemento destacable de la figura es que, para dar soporte a los dos tipos de espacios de trabajo, privados y compartidos, se utiliza la entidad Social Unit, que pertenece al sistema social, así como Group y User, y que sirve para referenciar de forma abstracta a estos dos tipos de entidades. Todo espacio de trabajo se encuentra vinculado a una unidad social y, recíprocamente, cada unidad social, ya sea usuario o grupo, dispone de un espacio de trabajo propio. Según lo expuesto, el carácter privado o compartido de un espacio de trabajo dependerá del tipo de unidad social vinculada a él: si se trata de un usuario hablamos de espacio privado, si es un grupo es un espacio compartido. Por otro lado, desde cada espacio de trabajo es posible acceder a una colección de proyectos y cada proyecto pertenece con exclusividad a un espacio determinado.

Como se puede observar en la figura, nada impide en el modelo incluir proyectos en un espacio privado. Sin embargo, desde las interfaces de usuario de este subsistema que abordaremos a continuación, se impone la restricción de que sólo se pueden realizar despliegues sobre espacios compartidos. Esto es debido a la orientación colaborativa de la plataforma. Cada proyecto guarda un enlace con su plantilla de pro-

yecto y ésta puede describir varios proyectos, cada uno de ellos desplegados sobre un espacio diferente. Además los proyectos incluyen una colección de actividades y éstas guardan una relación con el proyecto al que pertenecen. Asimismo se observa que cada actividad queda descrita con una única plantilla de actividad, agregada en la plantilla de proyecto correspondiente, y que ésta puede describir varias actividades pertenecientes a proyectos diferentes, sobre espacios de trabajo distintos. Finalmente, cada actividad mantiene una relación con un estado que representa en abstracto uno de los 6 posibles valores del ciclo de vida. La relación next sirve para dar soporte a la lógica de transición característica de cada estado específico.

3.4.2. La interfaz del subsistema de colaboración

Una vez descritos los elementos que constituyen el subsistema de colaboración, estamos en disposición de discutir cómo éstos se utilizan para dar forma a las interfaces de usuario Web. En este sentido, podemos distinguir, dentro de este subsistema, tres tipos de interfaces que aparecen en cada una de las áreas que se presentaron en la sección 3.3.2: interfaces para el diseño de modelos de colaboración, interfaces para el despliegue de estos modelos sobre los espacios de trabajo e interfaces de los espacios de trabajo en sí mismos. A lo largo de esta subsección presentaremos en detalle todos ellos. No obstante, hay que advertir, que la discusión de algunos de los elementos que aparecen en estas interfaces, en especial aquellos relacionados con el diseño y gestión de roles y servicios, se postergará para más adelante, en la sección 3.6.2, donde abordaremos las interfaces del sistema de integración.

Interfaz para el diseño de plantillas de proyecto

La interfaz para el diseño de plantillas de proyecto permite a los diseñadores instruccionales definir la dimensión colaborativa de un escenario indicando la colección de plantillas de actividad que lo constituyen. Tal y como se comentó con anterioridad, el propósito de las plantillas de proyecto es tan sólo organizar el trabajo de los estudiantes mediante la agregación de un conjunto de plantillas de actividad. Por tanto, para poder definir una plantilla de proyecto es conveniente disponer de las plantillas de actividad previamente definidas, aunque ambas tareas de diseño –definición de la plantilla de proyecto y plantillas de actividad – puede realizarse en paralelo.

Como se aprecia en la figura 3.12, en la parte de la izquierda aparece la relación de las plantillas de proyecto ya definidas en la plataforma (❶). El diseñador instruccional puede seleccionar un elemento de esta lista para editarlo en el formulario de edición de la derecha. Este elemento aparecerá resaltado en negrilla. Para crear nuevos proyectos y borrar los ya existentes se utilizan los botones de creación y borrado (❷). Al borrar una plantilla de proyecto, ésta desaparece de la lista y la plantilla editada pasa a ser otra. Sin embargo la eliminación de una plantilla de proyecto no supone la eliminación de las plantillas de actividad incluidas en él, ya que se consideran productos independientes y potencialmente reutilizables por otras plantillas de proyecto. Si se desea eliminar una plantilla de actividad, debe accederse a la interfaz de diseño de plantillas de actividad para hacerlo.

Figura 3.12. Interfaz para el diseño de plantillas de proyecto.

Por su parte, cuando se pulsa sobre el botón de creación de plantillas de proyecto, aparece una nueva plantilla de proyecto seleccionada sobre la lista con un nombre temporal. Este nombre se puede cambiar editando la descripción general del proyecto haciendo uso del formulario a tal efecto (❸).

El último paso que debe ser llevado a cabo para definir una plantilla de proyecto consiste en incluir dentro de la misma todas las plantillas de actividad que formen parte del escenario. Para ello se utiliza la parte del formulario de edición de plantillas de actividad (❹). En ella aparece una relación de las plantillas incluidas. Pinchando sobre cada una de ellas, puede accederse a su edición en la interfaz de plantillas de actividad. Encima de la lista hay un menú desplegable para seleccionar la plantilla que se desea añadir y un botón para realizar la adición. Para borrar una plantilla de actividad debe marcarse ésta en la lista y pulsar el botón de borrado.

Interfaz para el diseño de plantillas de actividad

La interfaz para el diseño de plantillas de actividad permite a los diseñadores instruccionales definir plantillas para las actividades que conforman una experiencia de aprendizaje. Una vez definida una colección de plantillas de actividad, los diseñadores instruccionales puedes acceder a la interfaz para el diseño de plantillas de proyecto e incluir allí todas las plantillas de actividad que consideren oportunas. En cualquier caso, no hay que olvidar que las plantillas de proyecto mantienen sólo una referencia a una colección de plantillas de actividad previamente definidas, de manera que si, una vez incluidas en una plantilla de proyecto, éstas son modificadas, la plantilla de proyecto se ve automáticamente actualizada.

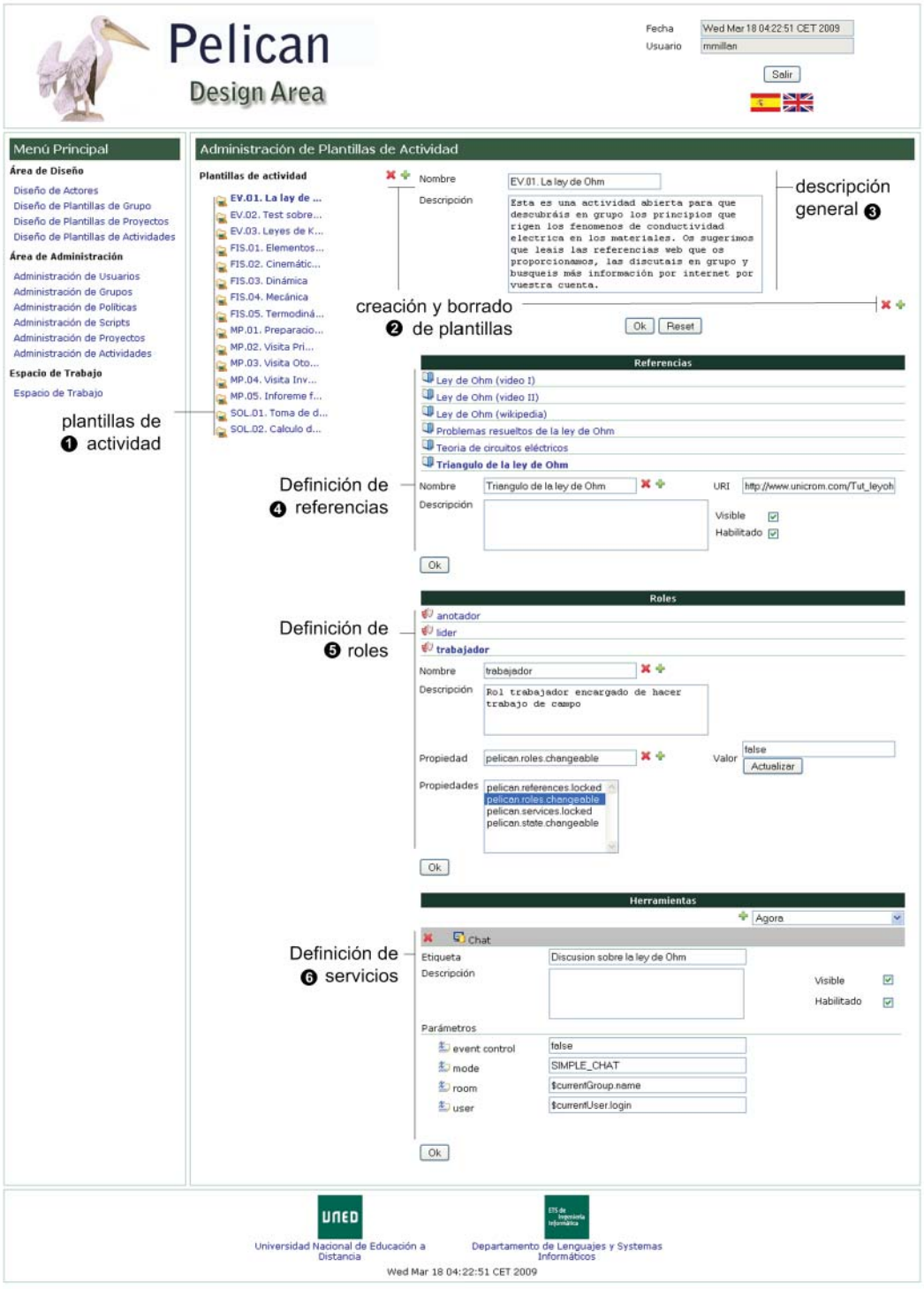

Figura 3.13. Interfaz para el diseño de plantillas de actividad.

Como puede apreciarse en la figura 3.13, en la parte de la izquierda aparece listada la colección de plantillas de actividad que están definidas en la plataforma (❶). Al seleccionar una de ellas, el elemento de la lista se ilumina en negrilla y la plantilla pasa a ser editada por el formulario de edición de la derecha. Para crear y borrar plantillas

se utilizan los botones de creación y borrado a tal efecto (❷). En concreto, si el diseñador pulsa el botón de borrado, la plantilla de actividad seleccionada se borrará de la lista y una nueva plantilla pasará a ser editada. La eliminación de una plantilla de actividad implica, naturalmente, su desaparición de todas aquellas plantillas de proyecto donde aparecía referenciada. Por su parte, la creación de una nueva plantilla requiere que el diseñador presione el botón de creación. Como consecuencia, aparecerá seleccionada en la lista una nueva plantilla cuyo nombre podrá ser modificado haciendo uso de la parte de descripción general del formulario (❸). Aquí es posible incluir un nombre para la plantilla y una descripción sobre la misma.

Tras la descripción general, se encuentra la sección dedicada a la definición de referencias (❹). En primer lugar, aparece una relación de todas las referencias que forman parte de la plantilla. El diseñador puede seleccionar una de ellas para editarla sobre el formulario que aparece debajo. Como se puede apreciar, este formulario está constituido por un campo que permite asignar un nombre a la referencia, otro para incluir una descripción sobre la misma y un tercero donde debe escribirse la dirección Web del recurso. El nombre y la descripción son utilizados, como veremos, para presentar, sobre el espacio de trabajo, la referencia mientras que la dirección Web es necesaria para construir el hipervínculo que permite acceder al recurso en Internet.

Adicionalmente, las referencias disponen de dos selectores que marcan el grado de visibilidad de cada una de ellas sobre el espacio de trabajo. En concreto, sólo si una referencia se marca como visible, el hipervínculo asociado se mostrará en el espacio de trabajo. Similarmente, si una referencia se marca como habilitada, en el espacio de trabajo aparecerá la referencia como un hipervínculo. Si por el contrario esta casilla no está seleccionada (y la de visibilidad sí) entonces aparecerá el nombre de la referencia pero no podrá accederse al recurso ya que ésta no se mostrará como un hipervínculo. Estas propiedades de visibilidad y accesibilidad, propias también de otro tipo de elementos del subsistema de colaboración, permiten adaptar los espacios de trabajo sobre los que se presenta el escenario activando o desactivando las referencias. Es un requerimiento frecuente de la especificación de los escenarios alterar la visibilidad y acceso de las referencias dinámicamente a lo largo del flujo de instrucción. Para dar soporte a esta necesidad puede hacerse uso de las capacidades del subsistema de intervención, que serán descritas en la siguiente sección.

Interfaz para la administración de proyectos

Con la interfaz para la administración de proyectos, los administradores de la plataforma pueden crear un nuevo proyecto a partir de una plantilla previamente definida y asociarlo a un espacio de trabajo determinado. Además, esta interfaz permite indicar, concretamente, qué plantillas de actividad se desea desplegar sobre el mismo. La figura 3.14 ilustra esta interfaz. El primer paso de las tareas de administración consiste, en este sentido, en seleccionar sobre la estructura jerárquica de espacios de trabajo compartidos aquél sobre el cual se desea crear el proyecto (❶). El espacio seleccionado aparece resaltado en negrilla y los datos relativos al mismo son editados sobre el formulario de la derecha.

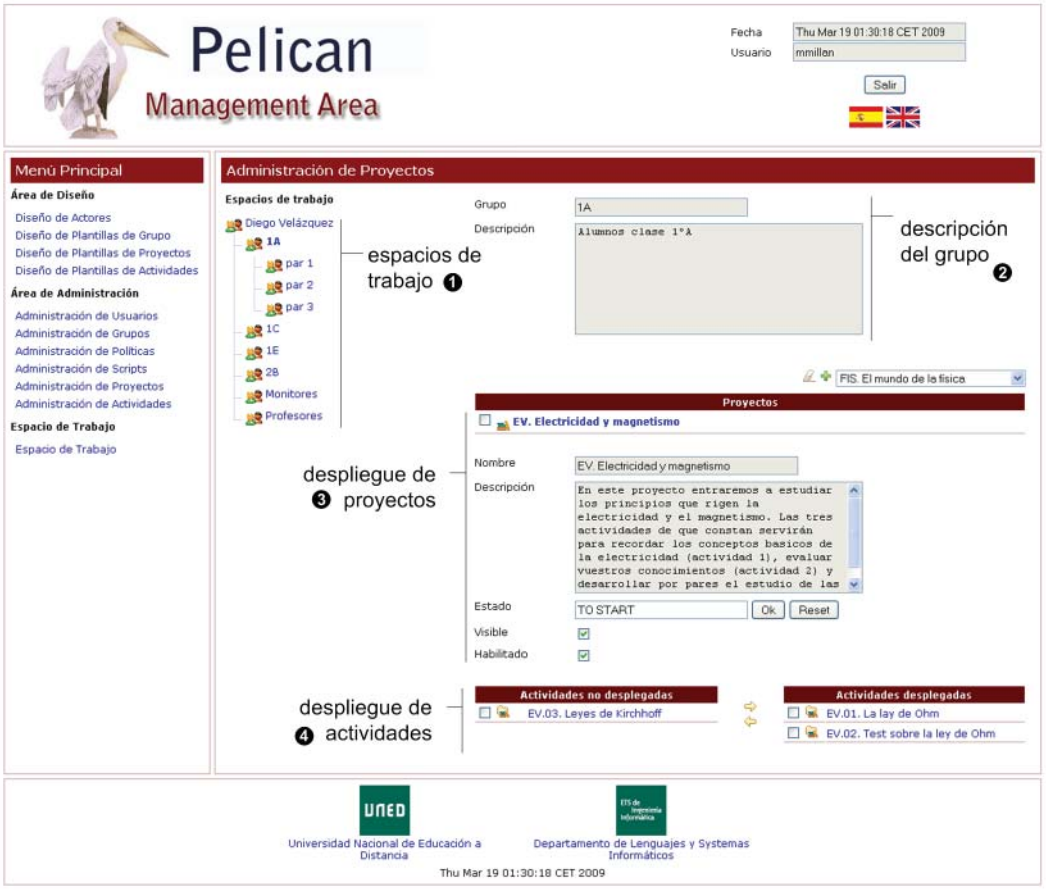

Figura 3.14. Interfaz para la administración de proyectos.

En concreto, la parte superior del formulario proporciona información acerca del nombre y la descripción del grupo al que pertenece el espacio seleccionado (❷). Debajo de esta información, se encuentra una lista con los proyectos que están actualmente desplegados (❸). Para desplegar un nuevo proyecto es necesario seleccionar su plantilla de la caja de selección que aparece encima, a la derecha, de esta lista y pulsar sobre el botón de despliegue. Como consecuencia se creará un nuevo proyecto en el espacio seleccionado y se añadirá éste sobre la lista. Similarmente, para replegar un proyecto es necesario seleccionarlo de la lista y pulsar sobre el botón de replegado. No obstante, esta acción sólo puede ser llevada a cabo si previamente se han eliminado todas las actividades contenidas en el proyecto. De otra manera, el replegado no surtirá efecto.

En cualquier momento los administradores pueden editar la información de un proyecto desplegado seleccionándolo en la lista. Esto provocará que la entrada correspondiente se resalte en negrilla y los datos del proyecto aparecerán sobre el formulario inferior. Aquí se presenta información general sobre la plantilla de proyecto – en concreto nombre y descripción – y se permite configurar tres aspectos fundamentales del mismo: su estado, su visibilidad y su accesibilidad. En efecto, desde este formulario se puede introducir manualmente un valor que describa el estado del proyecto.

Aquí se recomienda usar palabras concisas y específicas que permitan identificar el nivel de desarrollo de mismo, como por ejemplo *comenzado, en curso, interrumpido,* etc. Por su parte, los selectores de visibilidad y acceso permiten indicar a la plataforma si debe mostrar y permitir la entrada al proyecto o no al mostrarse éste sobre el espacio de trabajo. Estos tres aspectos del proyecto están habitualmente sometidos a cambios dinámicos según avanza el flujo de trabajo instruccional con lo que las capacidades del subsistema de intervención pueden utilizarse para efectuar automáticamente dichos cambios.

Una vez que se ha creado un proyecto y se han configurado sus valores iniciales, se dispone de una nueva entrada para el mismo dentro del espacio de trabajo (ver imagen 3.17). Sin embargo, este proyecto no contiene ninguna actividad ya que aún no se ha desplegado ninguna plantilla de actividad. Es decir, antes de poder comenzar con el desarrollo del proyecto sobre el espacio de trabajo, es necesario indicar a la plataforma qué plantillas de actividad, de entre todas aquéllas contenidas en la plantilla de proyecto, deseamos desplegar. Recuérdese que algunos escenarios requieren que el despliegue de las actividades se distribuya sobre diferentes espacios de trabajo, típicamente situados a distinto nivel social con lo que en cada espacio de trabajo solamente sería necesario desplegar un subconjunto del total de actividades definidas. Para realizar este despliegue, esta interfaz dispone de dos listas complementarias (❹): a la izquierda,la de las actividades aún no desplegadas y, a la derecha, la de las actividades ya desplegadas. Para desplegar una actividad es necesario seleccionarla en la primera lista y pulsar la flecha hacia la derecha, que moverá dicho elemento a la lista de la derecha. Recíprocamente para replegar una actividad, se selecciona de la lista de la derecha y se pulsa sobre la flecha hacia la izquierda. Cuando pulsamos sobre el nombre de una actividad no desplegada la plataforma salta al interfaz de diseño de plantillas de actividad para editar la plantilla asociada a la misma. Cuando pinchamos sobre el nombre de una actividad desplegada accedemos a la interfaz para la administración de la misma, que describimos en la siguiente subsección. El diseño de esta interfaz de usuario ha sido desarrollada para favorecer el despliegue transversal de actividades sobre espacios de trabajo a diferentes niveles sociales. Sin embargo, como contrapartida ver dónde hay actividades desplegadas de un proyecto requiere recorrer extensivamente el árbol de espacios de trabajo.

Interfaz para la administración de actividades

Una vez que las actividades se han desplegado sobre los espacios de trabajo de la plataforma, es posible configurarlas y adaptarlas al contexto social donde van a ser desarrolladas. Para ello, Pelican proporciona la interfaz de administración de actividades, que aparece representada en la figura 3.15. Para trabajar con esta interfaz lo primer que hay que hacer es seleccionar un espacio de trabajo en el árbol que aparece a la izquierda (❶). Una vez seleccionado, aparece sobre el formulario de la derecha (❷) todas las actividades desplegadas sobre el mismo. En concreto podemos encontrar, en primer lugar, el nombre del grupo al que pertenece el espacio de trabajo compartido y después sendas cajas de selección para indicar la actividad que queremos configurar y el proyecto al que ésta pertenece.

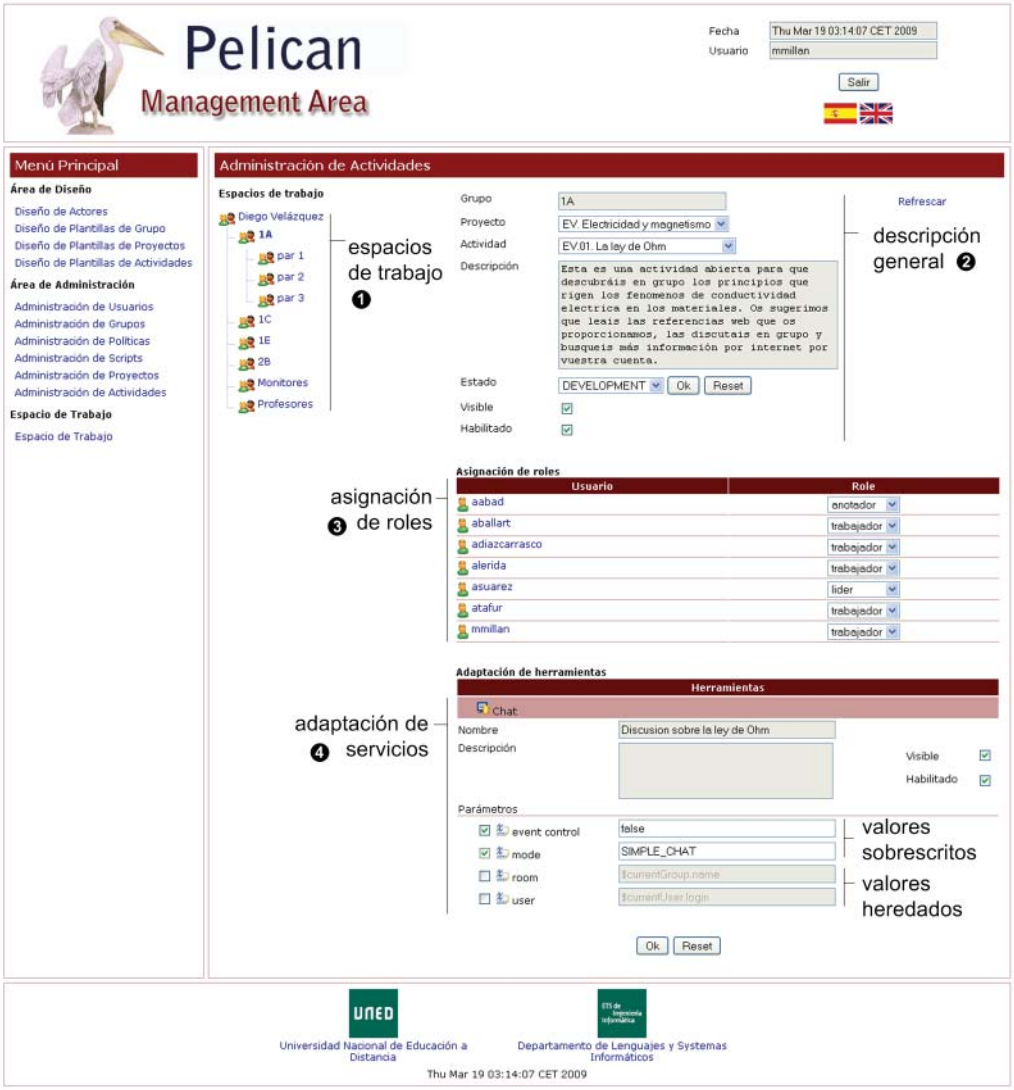

Figura 3.15. Interfaz para la administración de actividades.

Debajo se muestra la descripción de la plantilla de actividad asociada y se puede indicar el estado de la misma mediante una caja de selección. Aquí aparecen todos los posibles estados de la actividad y no se tienen en cuenta las restricciones de transición que dependen de los permisos concedidos a los actores según se explicó con anterioridad. Esto es debido a que esta es una interfaz de administración que debe permitir realizar libremente cualquier cambio.

Finalmente, las actividades también tienen sendos selectores para indicar el nivel de visibilidad y acceso que deben presentar éstas desde los espacios de trabajo. Nuevamente enfatizamos aquí que éstos son típicos valores que se alteran fácilmente de forma automática haciendo uso de las capacidades del subsistema de intervención según avanza el flujo de trabajo instruccional asociado al escenario pedagógico que se está desarrollando.

Interfaz del espacio de trabajo

Los espacios de trabajo constituyen la parte de las interfaces de la plataforma más delicada de diseñar ya que están dirigidas a dar soporte a las experiencias de aprendizaje llevadas a cabo por los estudiantes. Como se ha visto a lo largo de esta sección, su propósito es por un lado proporcionar consciencia del contexto social en el que se desarrollan las experiencias y por otra proporcionar un entorno virtual donde presentar los proyectos y las actividades y dar acceso a las referencias y herramientas integradas que las constituyen. En este sentido, podemos distinguir 3 vistas o perspectivas de los espacios de trabajo: la vista general, la de proyecto y la de actividad .

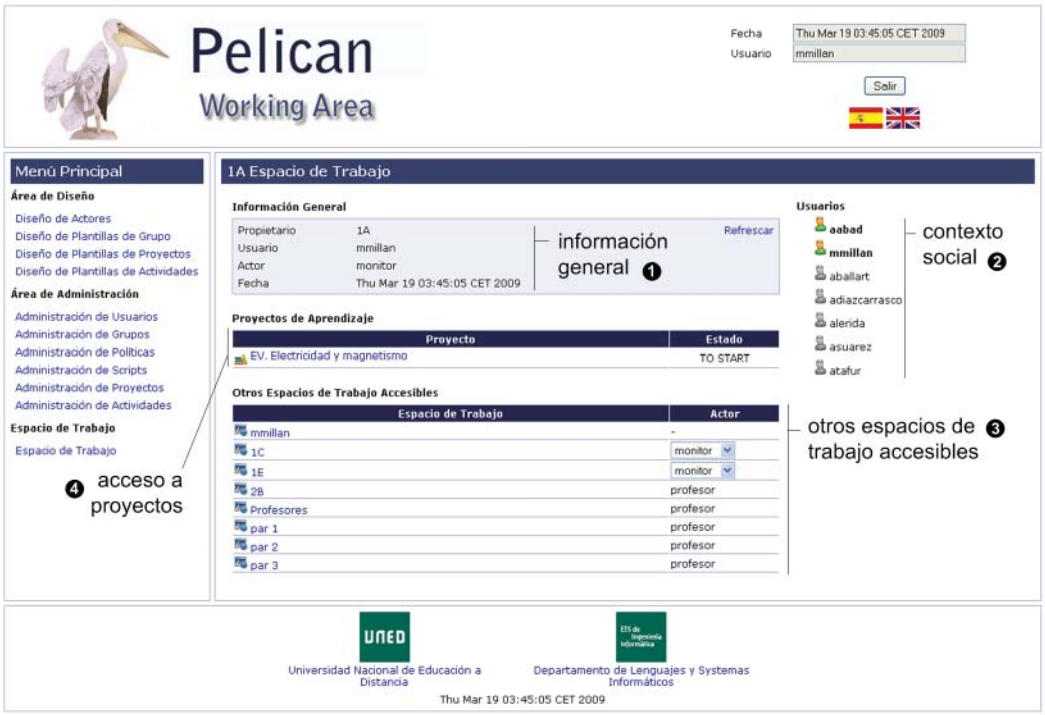

Figura 3.16. Interfaz del espacio de trabajo.

En la perspectiva general del espació de trabajo, que se muestra en la figura 3.16, aparece, en primer lugar, un cuadro que resume la información general del espacio seleccionado (❶). En concreto se indica quién es el propietario del mismo. Si es un espacio privado aparece el nombre del usuario. Si es compartido, aparece el nombre del grupo. También aparece el nombre del usuario que ha accedido al espacio y el actor que está encarnando, dentro del mismo, en ese momento. Y finalmente se presenta la fecha actual del sistema.

A la derecha (❷), se muestra el contexto social del espacio de trabajo. En el caso de los espacios compartidos aquí aparecerán todos los miembros del grupo ordenados lexicográficamente, poniendo primero los conectados (y resaltándolos en negrilla) y luego los no conectados. Esta información es de utilidad ya que saber qué compañeros están actualmente conectados en la plataforma fomenta la interacción con los

mismos. A continuación se muestra la relación de proyectos desplegados sobre el espacio adjuntando el valor de su etiqueta de estado (❸). En el ejemplo de la figura, sólo hay desplegado un único proyecto llamado *EV. Electricidad y Magnetismo,* y cuyo estado es *TO START*. Si se pincha sobre el nombre del proyecto, éste se abre sobre el espacio de trabajo.

Finalmente, aparece la relación de otros espacios de trabajo, tanto los compartidos como el privado asociado al usuario, que también son accesibles desde éste. Nótese, que, desde esta perspectiva, la estructuración jerárquica de los espacios de trabajo es aplanada en aras a la simplicidad para los estudiantes. Además, como puede apreciarse, en la lista aparece junto al nombre de cada espacio de trabajo compartido, el actor con el que el usuario participa si entra en el mismo. Para el caso de que el usuario sólo esté registrado con un actor en el grupo asociado, este campo será texto informativo. En el caso de grupos donde el usuario aparece encarnando más de un actor, éste debe indicar con qué actor en concreto desea entrar a participar en el espacio de trabajo. Para ello se utiliza la caja desplegable con todos los actores asociados al usuario en curso. Este es el caso de los espacios compartidos 1C y 1D de la figura.

Interfaz de proyectos desplegados sobre el espacio de trabajo

Cuando se abre un proyecto en el espacio de trabajo la interfaz se transforma de acuerdo a lo que aparece representado en la figura 3.17 para mostrar toda la información relativa al proyecto.

Figura 3.17. Interfaz de proyectos desplegados sobre el espacio de trabajo.

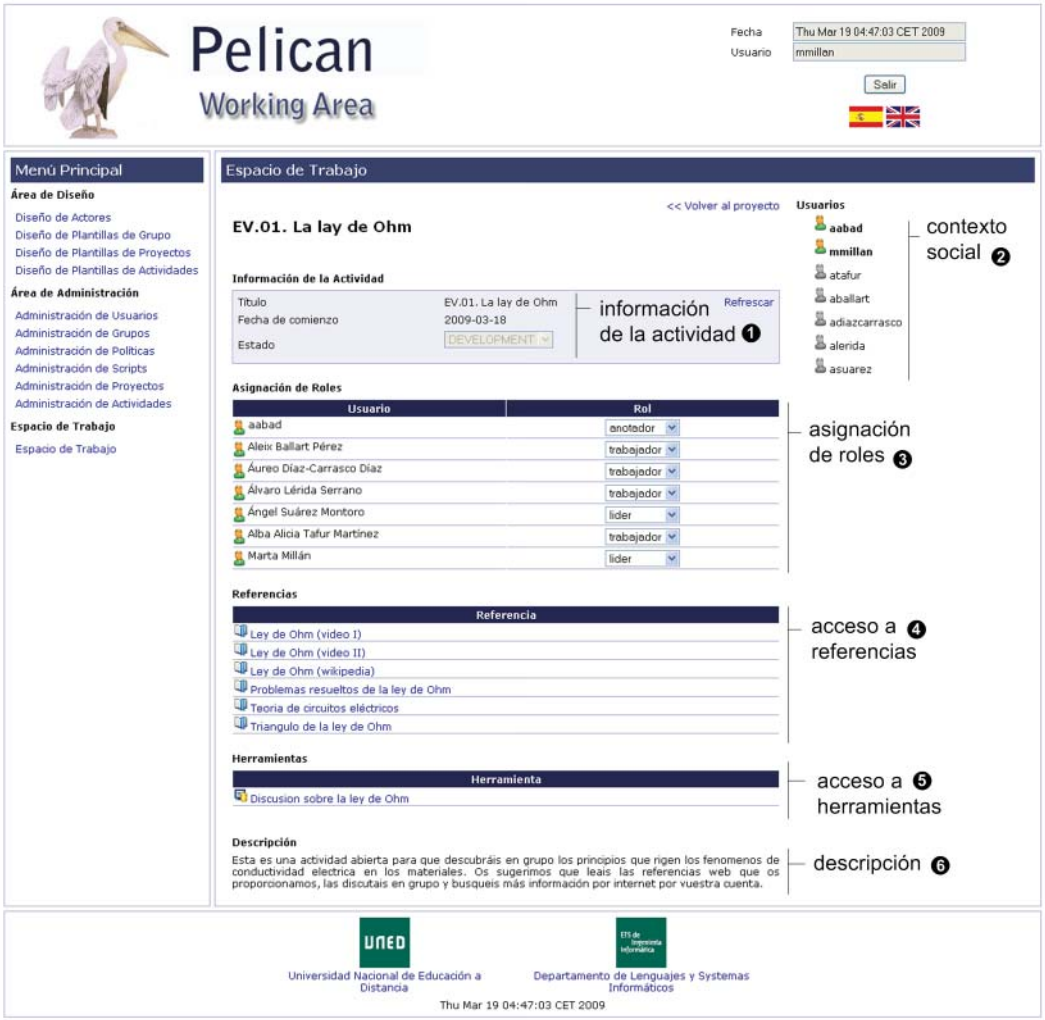

Figura 3.18. Interfaz de actividades desplegadas sobre el espacio de trabajo.

En primer lugar, el cuadro de información general (❶) cambia para presentar el nombre del proyecto, la colección de miembros del grupo implicados, la colección de otros grupos que desarrollan el mismo proyecto en sus respectivos espacios de trabajo y la fecha de despliegue. La información relativa al contexto social no se ve alterada (❷). Sin embargo, ahora aparece la colección de actividades desplegadas junto con su estado de desarrollo de acuerdo a los valores establecidos en el ciclo de vida de la actividad (❸). Al pinchar sobre el nombre de una actividad, ésta pasa a abrirse sobre el espacio de trabajo. Al final del todo (❹), se muestra una descripción del proyecto que proviene del texto que fue introducido por los diseñadores instruccionales en la plantilla de proyecto durante la fase de diseño. Como puede apreciarse en la figura, el proyecto mostrado consta de dos actividades. La primera de ellas esta en estado de desarrollo y la segunda aún no ha comenzado. Ambas son visibles pero tan sólo la primera ha sido configurada como accesible. Nótese que esta es una manera muy común de articular flujos de trabajo instruccionales que impliquen un secuenciamiento en el desarrollo de las actividades.

Interfaz de actividades desplegadas sobre el espacio de trabajo

Al acceder a una de las actividades desplegadas, ésta se abre y el espacio de trabajo se modifica para incorporar los elementos definidos en ella. Como puede verse en la figura 3.18, la información de contexto (❷), es la única parte de esta interfaz que permanece contante a lo largo de estas tres vistas. Ahora, la información general (❶) incluye el título de la actividad abierta, la fecha en la que ésta fue desplegada y un a caja de selección que indica el estado actual de la misma y que permite cambiarlo a otro de los estados subsiguientes siempre y cuando se tenga permiso para ello. En el ejemplo de la figura, el estado de la actividad es *en desarrollo*, y dado que esta pantalla corresponde al profesor, la transición al único estado siguiente (*pendiente de evaluación*) no esta permitida para este actor.

Dentro de esta interfaz, los miembros del grupo también pueden acceder a las referencias definidas en la plantillas de actividad. Al pinchar sobre cada una de las entradas de la lista de referencias (❹) se abre una nueva ventana del navegador y se muestra allí el contenido del recurso Web apuntado por dicha referencia. Al final del todo, también aparece la descripción que fue incluida en la plantilla de actividad por los diseñadores instruccionales para proporcionar a los estudiantes directrices y orientación acerca de qué trabajo deben hacer y cómo deben abordarlo

3.5. Soporte a la intervención adaptativa

En las dos secciones anteriores, se presentaron el subsistema social y el subsistema de colaboración de Pelican que permiten dar soporte a dos dimensiones fundamentales de la especificación de escenarios de aprendizaje. El subsistema social, proporciona un lenguaje de modelado para especificar la estructura de una comunidad virtual de aprendizaje así como mecanismos para construirla y mantenerla a lo largo del tiempo. El subsistema de colaboración, por su parte, proporciona también un lenguaje de modelado para la especificación del trabajo colaborativo de las experiencias y ofrece la infraestructura tecnológica necesaria para llevar a cabo el desarrollo de las mismas sobre ciertos contextos sociales.

Ambos subsistemas dan soporte a la descripción de los elementos estructurales de los escenarios de aprendizaje. Sin embargo, la mayoría de estos escenarios también incluyen aspectos dinámicos, tanto en la dimensión social como colaborativa, que no son soportados por estos subsistemas. En este sentido, cabe distinguir dos aspectos complementarios de la especificación de experiencias de aprendizaje:

- **Aspectos estáticos.** Los aspectos estáticos de la especificación de un escenario describen aquellos elementos que son conocidos en la fase inicial de diseño, previa al desarrollo de la experiencia. En la dimensión social, por ejemplo, la colección de actores y plantillas de grupo que se necesitan para desarrollar un escenario son típicos aspectos estáticos del mismo. Por su parte, en lo colaborativo, la colección de plantillas de proyecto y actividad asi como sus referencias, herramientas y roles permiten organizar el trabajo colaborativo y son frecuentemente considerados aspectos estáticos.

- **Aspectos dinámicos.** En contraposición, los aspectos dinámicos son aquéllos que se expresan a través de directrices de actuación en tiempo de diseño y que se aplican en tiempo de desarrollo para generar o transformar elementos dentro del escenario cuya existencia o estructura interna no se conoce apriorísticamente en tiempo el diseño. En la dimensión social, el método pedagógico de Jigsaw supone un claro ejemplo en este sentido, ya que los grupos expertos emergen en tiempo de desarrollo como consecuencia de una determinada decisión colaborativa lo que imposibilita que su composición sea conocida durante el diseño. Similarmente, los escenarios de aprendizaje en que los estudiantes determinan por consenso una división del trabajo para descomponer cada actividad en otras más sencillas requieren de la creación en tiempo de desarrollo de plantillas de actividad para dar soporte a las nuevas actividades fruto de tal descomposición.

Es necesario advertir, que la frontera entre estos dos tipos de aspectos es difusa, en el sentido de que el carácter estático o dinámico de los mismos depende del escenario en cuestión que se esté diseñando. En efecto, en la descripción de los aspectos estáticos pusimos como típico ejemplo las plantillas de actividad. Sin embargo, en la descripción de los aspectos dinámicos, precisamos que, para algunos escenarios, la construcción de las mismas puede tener que realizarse dinámicamente en tiempo de desarrollo.

Para hacer frente a la captura de los aspectos dinámicos que aparecen inherentemente vinculados a la especificación de la mayoría de los escenarios de aprendizaje colaborativo se utilizan las capacidades de adaptación dinámica proporcionadas por el subsistema de intervención, cuyo estudio abordaremos en esta sección. De esta manera, las facilidades de este subsistema se aplican transversalmente al resto de los subsistemas para completar la especificación de los escenarios de aprendizaje colaborativo en sus aspectos dinámicos. En concreto, las competencias de este subsistema para con el resto de subsistemas se resumen a continuación:

- **Intervención sobre el subsistema social.** El subsistema de intervención permite especificar todos aquellos aspectos dinámicos que aparecen relacionados con la dimensión social de los escenarios de aprendizaje. La creación, eliminación o redefinición de actores, la creación y borrado de plantillas de grupo, el registro de nuevos usuarios, la creación, borrado y fusión de grupos, la asignación de usuarios a grupos etc., son algunos de los ejemplos más característicos de operaciones de intervención que pueden llevarse a cabo mediante las prestaciones de este subsistema.

- **Intervención sobre el subsistema de colaboración.** Con respecto al trabajo colaborativo, durante la fase de desarrollo de la experiencia, también es frecuente aplicar operaciones de intervención que fundamentalmente tienen por objeto alterar el flujo de trabajo instruccional. Entre otras cosas, es posible crear y borrar plantillas de proyecto y actividad, desplegarlas o replegarlas sobre los espacios de trabajo de la plataforma y configurar dinámicamente los proyectos y actividades así resultantes para adaptarlos puntualmente a los requerimientos del escenario.

Como apuntamos en la sección anterior, un típico ejemplo de esto último consiste en la modificación de las propiedades de visibilidad y accesibilidad que presentan ciertos artefactos. No obstante, no hay que olvidar que este es tan sólo un caso de adaptación frecuente pero elemental.

- **Intervención sobre el subsistema de integración.** El subsistema de integración, que permite incorporar herramientas externas a los espacios de trabajo de la plataforma, también hace un uso intensivo de las prestaciones de intervención, tal y como se explicará más adelante en la sección 3.6. En efecto, la incorporación de herramientas requiere frecuentemente de la especificación de protocolos de integración específicos, que dependen de la naturaleza tecnológica de las mismas y de los objetivos concretos del escenario, para alcanzar un grado de interoperabilidad y cohesión apropiados. Los mecanismos de adaptación proporcionados por este subsistema resultan idóneos para describir dichos protocolos.

El subsistema de intervención puede ser conceptualizado de esta manera como un mecanismo que permite definir y aplicar esquemas adaptativos en la plataforma. En efecto, gran parte de la descripción de los aspectos dinámicos de un escenario que comentábamos anteriormente, puede expresarse en términos de una cadena de transformaciones que, aplicadas en el punto preciso dentro del flujo de trabajo instruccional, consiguen adaptar el entorno para atender a los requerimientos de la experiencia. De esta manera, las tareas relacionadas con el uso de este subsistema se distribuyen a lo largo de dos fases. En la fase de diseño los diseñadores instruccionales definen las transformaciones que deben ser llevadas a cabo durante el desarrollo de la experiencia e indican en qué momento deben ser aplicadas dentro del flujo de trabajo instruccional. Durante el desarrollo, dichas transformaciones son aplicadas dentro de un contexto pedagógico específico para adaptar el entorno a los requerimientos del flujo de instrucción. A lo largo de los dos apartados subsiguiente entramos a describir a nivel técnico estos dos tipos de tareas.

Especificación de transformaciones

Durante la fase de diseño, los diseñadores instruccionales definen todas aquellas estrategias de transformación que deban ser potencialmente aplicadas sobre el resto de los modelos internos de la plataforma para conseguir que ésta presente un comportamiento reactivo adecuado en relación a los requerimientos de cambio del escenarios de aprendizaje. Estas transformaciones se definen como **scripts de adaptación** expresados en P#, un lenguaje de scripting, de dominio y orientado a objetos puro que es propio de la plataforma Pelican. Posteriormente, y de manera opcional, los diseñadores instruccionales pueden definir **reglas políticas,** también descritas en este lenguaje, para establecer el momento exacto, dentro del flujo de trabajo instruccional, donde se desea que se apliquen automáticamente las transformaciones definidas en los scripts. Otras propuestas como [Coi et al., 2008] también usan el término política para referirse a reglas que gestionan permisos de acceso al entorno. En nuestro caso no obstante, las reglas políticas son usadas como mecanismo para articular la definición de estrategias adaptativas.

Sintaxis de P#

Los scripts de adaptación en P# constituyen una representación computable de un algoritmo que expresa cierta lógica de trasformación aplicada sobre los modelos subyacentes de la plataforma. Desde el punto de vista tecnológico, cada script consiste en una secuencia ordenada de algunas de las construcciones gramaticales que se exponen a continuación. P# es una extensión orientada al dominio de aprendizaje colaborativo del lenguaje de scripting Groovy. Para una revisión más profunda de su sintaxis consulte [Groovy] o [König et al., 2007]:

- **Comentarios.** Los comentarios son fragmentos de texto ignorados por el lenguaje que son utilizados dentro de los scripts para aclarar partes del código, delimitar zonas del mismo y declarar los propósitos de las construcciones definidas por los programadores tales como clases o funciones. En concreto podemos distinguir dos tipos básicos de comentarios:
  - *Comentarios de línea.* Los comentarios de línea son frases cortas que vienen a documentar instrucciones en una sola línea o bloques de instrucciones que comienzan en la misma. Para indicar al interprete de P# que comienza un comentario de línea se precede al texto que lo forma de un doble carácter de barra inclinada //.
  - *Comentarios de bloque.* Los comentarios de bloque permiten acotar un conjunto de líneas que serán ignoradas por el interprete del lenguaje y suelen ser utilizados para hacer documentaciones de algoritmos o indicar el propósito de clases o procedimientos. Un comentario de bloque es un texto encerrado entre los caracteres /* y */.
- **Declaraciones.** El lenguaje P# permite definir variables que serán utilizadas potencialmente a lo largo de la ejecución de los scripts para almacenar valores temporales o finales. En concreto es posible distinguir cuatro tipos de declaraciones en P#:
  - *Declaraciones simples.* Las declaraciones simples permiten definir variables que referencian objetos de clases reconocidas por el lenguaje. Como puede verse en el patrón sintáctico adjunto, una declaración comienza bien con la palabra reservada *def* o con un nombre de tipo de los permitidos por el lenguaje. En el primer caso, la declaración no asignará un tipo a la variable declarada sino que éste será inferido posteriormente en función del valor que adquiera en los contextos gramaticales donde aparezca referenciada. En el segundo caso, por el contrario, se especifica su tipo y si la variable es utilizada dentro de una construcción que maneja tipos incompatibles se generará un error de tipos. El siguiente elemento consiste en el nombre de la variable, un identificador único que permitirá referenciar a la variable dentro del ámbito gramatical donde la declaración se haya producido. Finalmente, y de manera opcional, puede asignarse un valor inicial a la variable a través del operador de asignación (=) que, en el segundo caso, deberá ser compatible con el tipo especi-

ficado. Los tipos de una declaración pueden corresponderse con el nombre de una clase o con alguno de los tipos primitivos permitidos por el lenguaje. En concreto, éstos últimos coinciden con los definidos de la especificación de [Java]. Así los tipos *byte, short, int* y *long* son utilizados para representar enteros de diferente longitud. Los tipos *float* y *double* para representar números en coma flotante. Y *char* y *boolean* sirven para representar caracteres y valores lógicos respectivamente. Si se trata de un objeto de clase se deberá usar el operador *new* seguido del nombre de un método constructor de la clase con los parámetros actuales pertinentes.

```
      def nombre-var [= valor | new Constructor (p1...pN)]
nombre-tipo nombre-var [= valor | new Constructor (p1...pN)]
```

– *Declaraciones de rangos.* Los rangos sirven para representar una secuencia de valores enteros consecutivos comprendidos entre una cota inferior y otra superior. Es posible distinguir entre dos tipos de rangos: los crecientes, cuando la cota inferior es menor que la superior, y los decreciente, cuando ocurre al revés. Sintácticamente una variable de tipo rango se define asignando un valor inicial a la misma formado por sendas cotas separadas por el delimitador dos puntos (..). Si éste delimitador además incluye el carácter <, entonces el conjunto de valores excluye al valor de la cota superior.

```
def nombre-var = 1..10
def nombre-var = 1..<10
```

– *Declaraciones de listas.* Las listas dinámicas son utilizadas para definir conjuntos de valores ordenados y con posibilidad de elementos repetidos. Para definir una variable de tipo lista es necesario asignarle una lista. Las listas en P# se representan como una colección de valores separados por comas y encerrados entre corchetes. Para obtener una lista vacía se escribe una lista sin elementos formada únicamente por los corchetes.

```
def nombre-var = [1, 2, 3]
   def nombre-var = []
```

– *Declaraciones de tablas.* Estas estructuras de datos son tablas de dispersión utilizadas para asociar un valor único a cada elemento de un conjunto. Para declarar una variable como una tabla de dispersión es necesario asociarle un valor de tipo tabla. En P# las tablas se escriben como una secuencia de entradas separadas por comas y delimitada por corchetes, donde cada entrada consiste en un par de objetos clave–valor separados por el delimitador de dos puntos (:). La tabla vacía son unos corchetes encerrando a un delimitador de dos puntos.

```
def nombre-var = ['I':1, 'II':2]
      def nombre-var = [:]
```

– *Declaraciones de funciones.* En P# es posible definir funciones ya sea como métodos miembro de una clase o como subrutinas auxiliares . La definición de

una función responde a alguno de los patrones sintácticos mostrados a continuación. El primero de ellos permite definir una función sin especificar el tipo de retorno ni el de sus parámetros, mientras que el segundo sí. La cabecera de la función incluye un nombre para poder invocarla a lo largo del script, un conjunto de parámetros formales encerrados entre paréntesis, separados por comas y posiblemente vacío que serán utilizados dentro de la función para representar los argumentos actuales de la llamada y, finalmente un bloque de sentencias encerradas entre llaves que contiene el cuerpo de la función.

```
def nombre-función (def p11,...pkm) { ... }
tipo-retorno nombre-función (tipo1 p11,...p1n, ...
                             tipok pk1,...pkm) { ... }
```

– *Declaraciones de clases.* En P# también es posible definir abstracciones de datos en forma de clases. Desde el punto de vista sintáctico, una clase es una agrupación de atributos y métodos miembros semánticamente cohesionados entre sí. La declaración de una clase se hace de acuerdo al patrón sintáctico mostrado a continuación. Tras las palabras claves *def* y *class*, que indican el comienzo de declaración, sigue el nombre de la misma y opcionalmente el de la clase padre precedido de la palabra *extends* así como la colección de nombres de interfaces separados por comas que implementa precedidos de la palabra clave *implements*. El cuerpo de la clase es un bloque de declaraciones de atributos y métodos miembro entre los que destacan sus constructores. Hay que señalar que en P# cada vez que se declara un atributo *x* dentro de una clase, automáticamente se asume la existencia de los métodos de acceso y mutación *getX ()* y *setX (x)* sin necesidad de que éstos se definan explícitamente.

```
def class nombre-clase [extends clase-padre]
                       [implements if1,.. ifn] {
      // declaración de atributos miembros
      // declaración de métodos miembros
}
```

– *Declaraciones de interfaces.* En P# también es posible definir interfaces. La definición de un interfaz es similar a la de una clase, con la diferencia de que ahora la palabra reservada para hacerlo es *interface* y que el cuerpo está formado únicamente por declaraciones de cabeceras de funciones. Dado que las interfaces también pueden heredar de otras interfaces previamente definidas es posible utilizar la palabra *extends* para indicar a P# el nombre de la interfaz padre de la que ésta hereda sus propiedades.

```
def interface nombre-interfaz [extends interfaz-padre] {
      // declaración de prototipos de métodos miembros
}
```

• **Expresiones.** Las expresiones son construcciones sintácticas que devuelven un determinado valor al contexto dentro del cual éstas son evaluadas. Sintácticamente, cualquier objeto es una expresión, la referencia a los métodos y atributos

miembros de un objeto también lo son, así como la llamada a funciones auxiliares, cualquier expresión entre paréntesis o combinación de expresiones utilizando operadores soportados por el tipo de las mismas. En general, podemos distinguir en P# diferentes tipos de operadores:

– *Operadores aritméticos.* Los operadores aritméticos combinan por lo general dos expresiones que dan lugar a otra más compleja alterando su semántica. Éstos están vinculados comúnmente a las expresiones numéricas que devuelven valores enteros o de coma flotante y son la suma (+), la resta (-), el producto (*), la división (/), el resto de la división entera (%) y el cambio de signo (-).

– *Operadores relacionales.* Los operadores relacionales se utilizan para combinar expresiones y generar otra cuya evaluación devuelva al contexto de uso un valor de verdad (cierto o falso). Los operadores relacionales son: mayor (>), mayor o igual (>=), menor (<), menor o igual (<=), igual (==) y distinto (!=) y en conjunto (*in*). El tipo de las expresiones involucradas suelen ser valores numéricos (enteros o de coma flotante) o de verdad (entendiendo que *false* < *true*). No obstante, estos operadores se pueden sobrecargar para que sean aplicables a otro tipo de abstracciones. Así por ejemplo, en P# el tipo fecha (*Date*) sobrecarga estos operadores entendiendo que una fecha es menor que otra si ésta hace referencia a un instante de tiempo anterior.

– *Operadores lógicos.* Los operadores lógicos se utilizan para combinar expresiones lógicas que devuelven valores de verdad aplicando las reglas de la lógica proposicional. Estos operadores son: y lógica (&&), o lógica (||), exclusión lógica (^) y negación lógica (!).

– *Operadores de acceso.* Los operadores de acceso permiten acceder a diferentes elementos de una expresión. Así, para un objeto (o una expresión que devuelve un objeto) se puede utilizar el operador de punto (.) o interrogante–punto (?.) para acceder a los métodos y atributos miembro definidos dentro de la clase a la que pertenece. Mientras que el primero genera un error si el elemento sobre el que se aplica el operador es nulo, el segundo en ese caso devuelve la referencia nula pero no genera un error. En una expresión de tipo rango o lista, los corchetes con un índice entero en su interior *[n]* permiten hacer referencia al elemento n-1–esimo de la misma, y para una tabla, la construcción *[clave]*, donde clave es el nombre de una entrada de la tabla, devuelve el valor asociado a la misma dentro de ella.

– *Operadores de asignación.* En P#, las asignaciones, aparte de sentencias que asignan a la referencia de la izquierda el valor de la expresión situada a la derecha del igual, también se comportan como expresiones puesto que devuelven al contexto sintáctico donde aparecen el valor resultante de evaluar la expresión de la derecha. Esto permite por ejemplo, asignar a una variable el resultado de una asignación permitiéndose realizar asignaciones encadenadas del estilo *def a = b = 3.*

- *Operaciones de cadena.* En P# existen dos operadores espaciales que se utilizan sobre las cadenas de caracteres, definidas a través de un texto encerrado entre comillas dobles o simples. El operador de concatenación (+) permite unir dos cadenas colocando la segunda inmediatamente detrás de la primera. El operador de resolución de referencias ($) puede insertarse dentro de un texto de cadena justo antes de una referencia, lo que provocará que ésta se evalúe y se sustituya por su valor dentro del texto.

- *Cláusulas.* En P# también es posible definir un fragmento de código parametrizado como si fuera una expresión para poder asignárselo a una variable o pasarlo como parámetro a una función, por ejemplo. Este tipo de construcciones se llaman cláusulas y están formadas por la colección de parámetros separados por comas, una flecha (->) y el conjunto de sentencias, todo ello encerrado entre llaves. Las cláusulas pueden ser invocadas similarmente a como ocurre con las funciones, sustituyendo los parámetros formales por argumentos reales. Estas construcciones proporcionan un azúcar sintáctico que tiende a hacer el código de los scripts más sintético.

- *Operadores de expresiones regulares.* P# también permite definir expresiones regulares para obtener patrones léxicos y utilizarlos en problemas de procesamiento de textos. Una expresión regular en este lenguaje debe ir siempre encerrada entre barras inclinadas (/). Los operadores que se utilizan para definir este tipo de expresiones son muy numerosos y aquí citaremos solamente los más relevantes: cualquier carácter (.), comienzo de línea (^), final de linea ($), dígito (/d), espacio (/s), palabra (/w), alternativa (|), opcionalidad (?), 0 o muchos (*), 1 o muchos (+).

- *Operador condicional.* El operador condicional permite describir expresiones que toman un valor calculado de entre dos expresiones en función del valor de una tercera expresión. Sintácticamente este operador está compuesto por el token (?) y (:). Así por ejemplo, en la asignación *x = (a>b)? a : b*, el valor de la variable x dependerá del de a y b. Si a es mayor que b, x recibe el valor a mientras que si no lo es entonces tomará el valor de b. Es decir, esta sencilla expresión implementa la función mayor qué entre a y b.

- **Sentencias.** P# proporciona una colección de sentencias iterativas, condicionales, de control de errores y de entrada–salida que permiten a los administradores y diseñadores instruccionales escribir scripts más cómodamente. A continuación las describimos brevemente:

  - *Sentencia If–else.* La sentencia *if–else* permite definir fragmentos de código que se ejecutan sólo si se satisface cierta condición. Tras la palabra clave *if* se sitúa entre paréntesis una expresión que se evalúa como un valor de verdad. Si este valor es cierto entonces, y sólo entonces, pasa a ejecutarse la sentencia o bloque de sentencias que le sigue. Opcionalmente, tras esto puede incluirse la palabra reservada *else* y añadir a continuación otra sentencia o bloque de sentencias para que se ejecute en caso de que el resultado de la evaluación

sea falso. El flujo de control continuará después por la instrucción siguiente a la sentencia *if–else*.

```
if (expresión-lógica) { bloque-sentencias }
[ else { bloque-sentencias } ]
```

– *Sentencia witch–case.* La sentencia switch–case permite definir una batería de fragmentos de código, llamados casos, que se ejecutarán si el valor de una determinada expresión coincide con el patrón definido en el caso. A diferencia de en otros lenguajes, los patrones pueden ser una expresión de cualquier tipo compatible con aquélla con la que se compara el caso y no simplemente constantes numéricas. Una vez que el flujo de control entra por un caso, se ejecutan todos los casos que le siguen secuencialmente hasta que se encuentre una sentencia de corte (*break*). Además también es posible incluir al final un caso por omisión que se ejecutara si no se encontró ningún caso previo. Para ello se utiliza la palabra reservada *default*.

```
switch (expresión) {
 case expresión-caso1 : bloque-sentencias1; break
   ...
 case expresión-casoN : bloque-sentenciasN; break
[ default : bloque-sentencias-default ]
 }
```

– *Sentencia while.* La sentencia *while* se utiliza para iterar una sentencia o bloque de sentencias mientras se satisfaga una expresión lógica. Si la expresión nunca llega a hacerse cierta la expresión no se ejecuta jamás y si nunca se falsifica la iteración, y por ende la ejecución del programa, nunca terminará. Sintácticamente el patrón para escribir sentencias *while* es el siguiente:

```
while (expresión-logica) { bloque-sentencias }
```

– *Sentencia do–while.* En ocasiones interesa que un determinado fragmento de código se someta a iteración cierta cantidad de veces en función de la evaluación de una expresión lógica pero garantizando que al menos se ejecutará una vez dicho fragmento. Para ello se usa la sentencia *do–while* cuyo patrón sintáctico se muestra a continuación. En este caso se ejecutará siempre primero, en cada iteración, el bloque de sentencias y luego se comprobará el valor de la expresión:

```
do { bloque-sentencias } while (expresión-logica)
```

– *Sentencia for.* La sentencia de control de flujo iterativo *for* admite tres tipos de formulaciones sintácticas. La primera de ellas – la convencional – permite iterar la ejecución de una sentencia o bloque de sentencias un número determinado de veces controlado por un índice. En este caso, tras la palabra reservada *for* se escriben entre paréntesis tres expresiones: la primera para asignar un valor inicial al índice, la segunda para indicar la condición de continuidad

de la iteración y la tercera para indicar el incremento del índice en cada vuelta. La segunda formulación del bucle for es similar a la anterior pero el objetivo ahora es iterar tantas veces como elementos haya en una expresión de rango y asignando en cada paso un valor consecutivo dentro del rango a la variable de índice. En este caso, tras la palabra *for* se escribe entre paréntesis el nombre de la variable de índice, la palabra reservada *in* y la expresión de rango para indicar el rango de valores que recorrerá a lo largo de la iteración. La tercera forma gramatical del *for* sirve para iterar sobre los elementos de una colección. Es decir, la variable de índice irá tomando consecutivamente los valores de una colección, tal como una lista o un conjunto, y sintácticamente es similar a la anterior a excepción de que ahora la expresión no referencia a un rango sino a una colección. A continuación se muestran los patrones sintácticos para estas tres formas de la sentencia *for*.

```
for (index=valor; exp-index; incremento-index)
 { bloque-sentencias }
for (index in min..max) { bloque-sentencias }
for (index in colección) { bloque-sentencias}
```

– *Sentencia try–catch–finally.* En P# muchas sentencias y expresiones son susceptibles de generar errores en tiempo de ejecución que aunque no es obligatorio tratar, siempre resulta conveniente. Si un fragmento de código, susceptible de fallo, no es tratado puede ocurrir que la ejecución del script, en ciertas condiciones de fallo, provoque una terminación anormal lo que conduzca a corromper el estado de la plataforma. Para controlar los fragmentos susceptibles de fallo se utiliza la sentencia *try–catch–finally*. Como se puede ver en el patrón sintáctico siguiente, tras la palabra reservada *try* se encierra entre llaves el código susceptible de error y después la palabra *catch* y entre paréntesis el objeto que recoge la información de la excepción potencialmente producida seguida del código de tratamiento del mismo, que suele expresarse en términos de ese objeto. A veces se utiliza la terminación *finally* para incluir entre llaves un nuevo código que se ejecutará siempre que se atraviese la sentencia, se haya alcanzado o no el bloque *catch*.

```
  try { bloque-susceptible-error }
  catch (Exception e) { bloque-tratamiento-error }
[ finally { bloque-final } ]
```

– *Sentencia assert.* Para controlar la ejecución de código a veces interesa comprobar que en determinados puntos de un script se verifican determinadas propiedades. La sentencia *assert* permite comprobar esto mediante la especificación de una expresión lógica que caso de evaluarse a falso lanzará una excepción abortando el flujo normal de ejecución. Generalmente esta sentencia se usa en combinación con la anterior mediante el siguiente patrón sintáctico.

```
assert (expresión-lógica)
```

– *Sentencias de salida.* P# incluye la familia de procedimientos sobrecargados *print* y *println* para imprimir sin y con salto de línea respectivamente el resultado de evaluar la expresión que se le pasa como argumento que puede ser de cualquier tipo. El objeto devuelto por la evaluación de la expresión será convertido a una cadena de caracteres invocando automáticamente al método *toString* de la clase a la que pertenezca el mismo. El patrón sintáctico de estas sentencias permiten que la expresión se pase entre paréntesis o simplemente a continuación de la palabra clave.

```
print[ln] expresión
print[ln] (expresión)
```

Variables implícitas de dominio en P#

Uno de los elementos que hace de P# un lenguaje específico de dominio [Fowler, 2005] orientado a la especificación de los aspectos dinámicos de escenarios de aprendizaje colaborativo en Pelican es la existencia de una colección de **variables implícitas de dominio** que referencian elementos relacionados con los artefactos de los subsistemas de la plataforma. Este conjunto de variables constituyen, para cada usuario, una representación de su estado dentro del entorno de aprendizaje indicando aspectos tales como el grupo actual al que pertenece, el proyecto y la actividad en que están inmersos, etc. El carácter implícito de estas variables indica que los diseñadores instruccionales pueden asumir su existencia dentro de P# sin necesidad de declararlas ya que son mantenidas y actualizadas por el entorno de ejecución, tal y como se discutirá más adelante. De hecho, la declaración de variables locales con el mismo nombre que una variable implícita supone un error de programación. Estos objetos resultan fundamentales para expresar estrategias de transformación adaptativas que dependan del estado de los usuarios de la plataforma.

En la tabla 3.2 aparece un resumen que describe la colección de las variables implícitas de dominio. La primera columna muestra el nombre de la variable, la segunda contiene una breve descripción de su significado y la tercera indica el tipo al que pertenece la misma. Como puede apreciarse, cada variable es, en realidad, un objeto P# que contienen una colección de atributos y métodos miembros. Desde las construcciones de este lenguaje estos elementos son accedidos con la notación de punto tal y como se describió en el subapartado anterior.

```
expresión-implícita.atributo
expresión-implícita.getAtributo ()
expresión-implícita.método (parámetros)
```

En este contexto, una expresión implícita es una expresión que comienza por el nombre de una variable implícita de dominio y que utiliza el operador de punto para acceder a nuevos valores del objeto que referencia. Cada atributo puede accederse escribiendo su nombre tras el punto o, en notación funcional, con el prefijo *get* y los paréntesis vacios detrás. Similarmente para invocar a los métodos también se usa la notación de punto.

Nombre	Descripción	Tipo
Sistema		
systemDate	Esta variable devuelve la fecha actual del sistema. Es útil cuando se quieren implantar scripts en cuya descripción interviene el tiempo	java.util.Date
systemTimestamp	Esta variable un entero que representa una marca de tiempo para el sistema. Suele utilizarse para controlar intervalos de tiempo	java.lang.Long
Sociedad		
currentUser	Representa al usuario que está utilizando la plataforma. Esta variable es utilizada para poder referenciar, de forma abstracta, al usuario sobre el cual se aplica el script	UserPolicyVariable
currentUserLocation	Esta variable representa la localización del usuario que esta utilizando la plataforma. Útil para implantar lógica adaptativa de internacionalización	java.lang.String
currentUserLoginDate	Esta variable contiene la fecha en la que el usuario actual hizo login en el sistema. Suele ser utilizado para controlar la duración de la sesión en Pelican	java.util.Date
currentActor	Representa al actor que está encarnando actualmente el usuario en curso. Esta información permite aplicar diferentes estrategias en función del actor que encarnan los usuarios	ActorPolicyVariable
currentPartners	Contiene la colección de usuarios que encarnan el mismo actor en el grupo que el usuario en curso. También accesible a través de currentGroup	java.util.List <UserPolicyVariable>
currentGroup	Referencia al grupo en el cual el usuario en curso se encuentra, actualmente trabajando. Se utiliza para aplicar scripts a los miembros de un grupo en su conjunto	GroupPolicyVariable
currentGroupTemplate	Representa a la plantilla de grupo asociada al grupo en curso. También accesible desde currentGroup. Útil para implantar scripts de adaptación del modelo de sociedad	GroupTemplatePolicyVariable
Colaboración		
currentActivity	Representa la actividad que se encuentra desarrollando actualmente el usuario en curso. Empleado para alterar el ciclo de vida de la actividad y la distribución de roles	ActivityPolicyVariable
currentActivityPartners	Contiene la colección de usuarios que encarnan el mismo role dentro de la actividad que el usuario en curso esta actualmente desarrollando. También accesible a través de currentActivity	java.util.List <UserPolicyVariable>
currentProject	Referencia el proyecto dentro del cual se inscribe el trabajo actual del usuario en curso. Se utiliza para conocer y controlar el estado del proyecto y para adaptar sus actividades al grupo	ProjectPolicyVariable
currentActivityTemplate	Representa a la plantilla de actividad asociada a la actividad actual. Se utiliza para alterar la definición de una actividad. También accesible desde currentActivity	ActivityTemplatePolicyVariable
currentProjectTemplate	Hace referencia a la plantilla de proyecto donde el usuario está actualmente implicado. Pueden declararse políticas para alterar dinámicamente el conjunto de plantillas de actividad que constituyen su composición interna. También accesible desde currentProject	ProjectTemplatePolicyVariable

Tabla 3.2. Colección de variables implícitas de dominio.

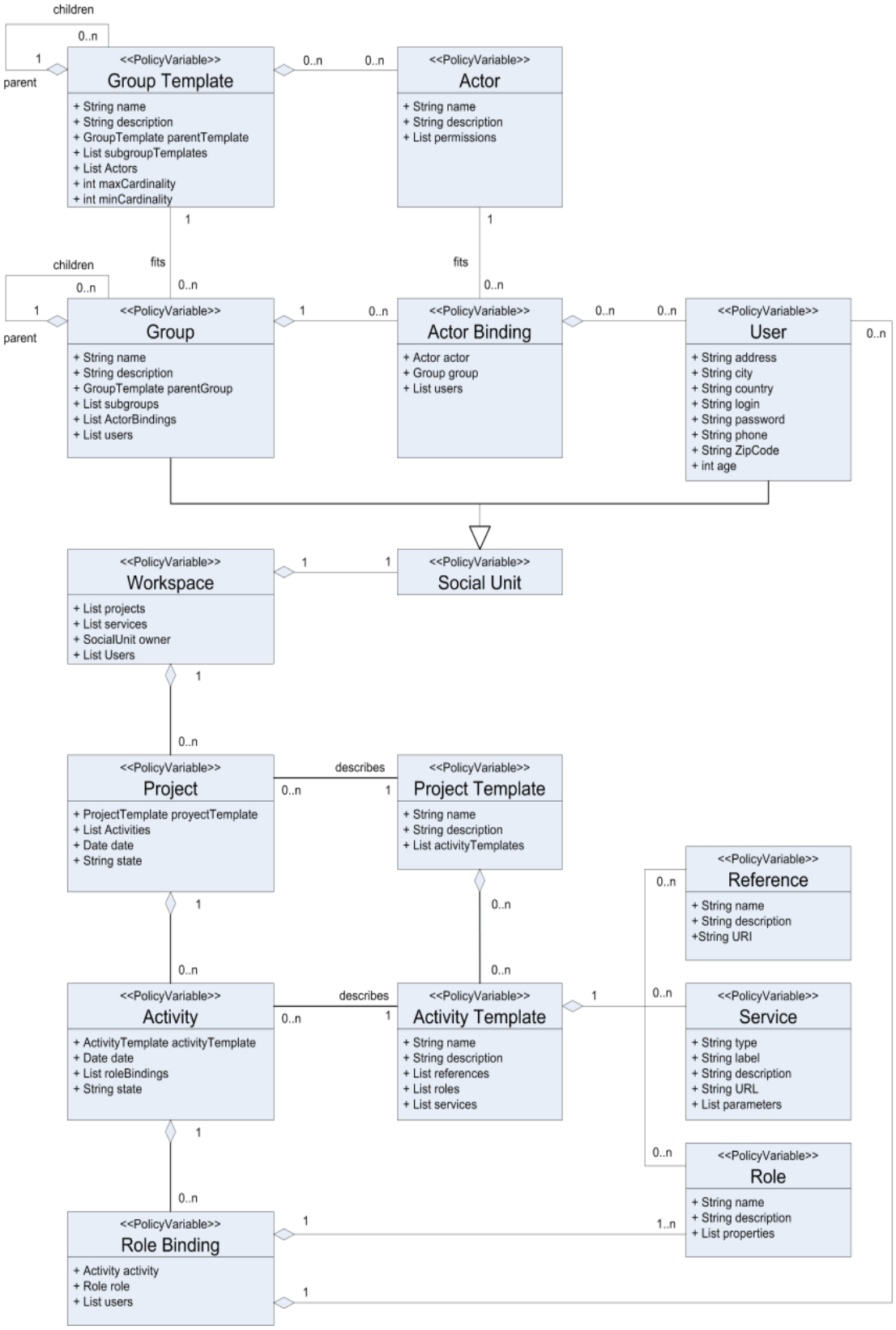

Figura 3.19. Modelo de objetos de las variables implícitas de dominio.

En realidad, las variables implícitas de dominio constituyen puntos de entrada de sólo lectura a los elementos de los modelos subyacentes de la plataforma referidos al usuario en curso al que hacen referencia. Para poder explotar toda esta información, los diseñadores instruccionales deben conocer, de forma precisa, estos modelos. La figura 3.19 es una representación en UML global y detallada del modelo de objetos de la plataforma, accesible desde las variables implícitas de dominio, que incluye la colección de atributos propios de cada entidad.

En una inspección preliminar de este diagrama puede verse que el modelo queda dividido en dos partes: la social y la colaborativa. Los principales puntos de entrada a la parte social son *las variables currentUser* y *currentGroup*. La primera hace referencia al usuario en curso y da acceso al modelo de usuario. Con la segunda se puede acceder al grupo en curso y navegar por la jerarquía de grupos y plantillas de grupo así como descubrir las asignaciones de actores a usuarios dentro de cada grupo. Ambas entidades son unidades sociales que conectan con un espacio de trabajo privado o compartido, punto de contacto con la parte colaborativa. En efecto desde allí pueden explorarse los proyectos, actividades, plantillas de proyecto y actividad, servicios, roles y recursos implicados en cada momento. No obstante, las variables *currentActivity* y *currentProject* también son puntos de entrada a esta parte del modelo.

Familia de tareas de dominio en P#

El otro elemento que hace de P# un lenguaje específico de dominio, es la existencia de una familia de tareas. En efecto, para facilitar a los diseñadores instruccionales la especificación de scripts de adaptación, P# proporciona una colección de primitivas de dominio llamadas **tareas** que llevan a cabo transformaciones atómicas, elementales y recurrentes sobre los modelos internos de los subsistemas de la plataforma. La invocación de estas tareas, con argumentos actuales específicos, combinadas mediante el uso de sentencias de control de flujo de programa iterativas y condicionales permite expresar esquemas de transformación elaborados lo que convierte a este lenguaje en un mecanismo flexible para el tratamiento de los aspectos dinámicos inherentemente asociados a los escenarios de aprendizaje colaborativo.

P# proporciona un total de 117 tareas de dominio para realizar transformaciones sobre la plataforma. La mayoría de ellas constituyen una versión computacional de un procedimiento de intervención accesible manualmente desde las interfaces Web de usuario. No obstante existe algunos otras que ofrecen un control más fino y preciso de determinados aspectos de Pelican sólo accesibles a los programadores de scripts. En concreto las tareas de P# se encuentran organizadas en cuatro librerías especializadas:

- **Librería common.** La librería *common* incluye una colección de tareas de carácter general que permiten realizar ciertas operaciones básicas fundamentales como por ejemplo invocar scripts desde otros scripts (lo que permite seguir un procedimiento constructivo incremental de los mismos), enviar mensajes y correos electrónicos a los usuarios o definir cronómetros para automatizar la ejecución de otros scripts.

- **Librería society.** La librería *society* contiene todas las tareas relacionadas con la gestión de los aspectos sociales de la plataforma tales como la creación, borrado, modificación o consulta de grupos, usuarios, actores, o plantillas de grupo.
- **Librería collaboration.** La librería *collaboration* permite realizar transformaciones elementales sobre los elementos del subsistema de colaboración tales como las plantillas de proyecto y actividad, los proyectos y actividades, los servicios, referencias y roles.
- **Librería tools.** La librería *tools* proporciona una colección de métodos que permiten establecer una comunicación directa y fluida con las herramientas integradas en la plataforma Pelican. En concreto esta librería está constituida por una tarea que permite invocar servicios Web publicados por las herramientas y otras que permiten interpretar flujos de datos de las herramientas en diferentes formatos.

La invocación de las tareas de dominio dentro de los scripts de P# es similar a la invocación de los métodos miembro de un objeto convencional. La sintaxis, representada a continuación indica que primero debe escribirse el nombre de la librería (*common, society, collaboration* o *tools*) seguido de un punto (.) y después la tarea en notación funcional:

```
nombre-librería.nombre-tarea (parámetros)
```

La tabla 3.3 muestra una relación del contenido de todas estas librerías. Para cada una de ellas se muestra, en la columna de la izquierda, la signatura de la tarea en notación funcional indicando primer el nombre de la tarea, la colección de parámetros separados por comas y después, tras dos puntos, el tipo de objeto de retorno que será algunos de los objetos definición en la figura 3.19 o alguno propio de P#. Adviértase que el significado de alguna de estas tareas puede no quedar claro en este punto puesto que aún no hemos terminado de presentar la arquitectura completamente.

Librería *common*	
Tarea	**Descripción**
callScript (id, properties): Object (1)	Permite invocar un script sobre el contexto en curso. Devuelve los resultados
callScript (id, user, properties): Object (2)	Permite invocar un script sobre el contexto de un usuario
callScript (name, properties): Object	Igual que en (1) antes pero el script es referenciado por nombre
callScript (name, user, properties): Object	Igual que en (2) antes pero el script es referenciado por nombre
evaluate (script): Object	Igual que en (1) pero el código del script es pasado como argumento
evaluate (script, user): Object	Igual que en (2) pero el código del script es pasado como argumento
expand (expression): String	Resuelve expresión y devuelve un String. Útil en esquemas de invocación
expand (expression, user): String	Resuelve una expresión sobre un contexto de usuario
load (type, id): Object	Recupera un objeto de la base de datos conocido su tipo e identificador
load (type, name): Object	Recupera un objeto de la base de datos conocido su nombre e identificador
loadFromContext (variable): Object (3)	Equivalente a referenciar directamente la variable
loadFromContext (variable, value): Object (4)	Recupera una variable del contexto. Si no se encuentra devuelve *value*
loadFromContext (user, variable): Object	Igual que en (3) pero para un contexto de otro usuario
loadFromContext (user, variable, value): Object	Igual que en (4) pero para un contexto de otro usuario
configLogger (name, level, file): void	Configura un traceador *name* para el código de scripts
logTask (loggerName, level, message): void	Vuelca a la traza del traceador *name* un mensaje nuevo

Librería common (continuación)	
Tarea	**Descripción**
remove (type, id): void	Borra el objeto de clase *type* e identificador *id*
remove (type, name): void	Borra el objeto de clase *type* e nombre *name*
removeFromContext (variable): void (5)	Borra una variable de contexto. Sin efectos sobre variables implícitas
removeFromContext (login, variable): void	Igual que en (5) pero sobre el contexto del usuario identificado por login
removeFromContext (user, variable): void	Igual que en (5) pero sobre el contexto de otro usuario
save (value): void	Persiste el objeto *value* en el sistema
saveInContext (name, value): void (6)	Persiste el objeto *value* en el contexto con nombre *name*
saveInContext (user, name, value): void	Igual que en (6) pero sobre el contexto de otro usuario
sendMail (from, to, subject, body): void	Envía un correo electrónico a un usuario del sistema
sendMessage (user, message): void	Envía un mensaje instantáneo a un usuario del sistema
sendMessage (user, title, type, message): void	Envía un mensaje instantáneo a un usuario del sistema
startTimer (user, name, script, delay): void	Ejecuta un script transcurridos *delay* ms
startTimer (user, name, properties, delay): void (7)	Lanza un evento time-over transcurridos *delay* ms
startTimer (user, name, properties, delay, period): void	Igual que en (7) pero repetido con periodicidad *period* ms
startTimer (user, name, propreties, date): void (8)	Igual que en (7) pero en una fecha determinada
startTimer (user, name, properties, date, period): void	Igual que en (7) pero repetido con periodicidad *period* ms
stopTimer (name): void	Para el timer periódico de nombre *name*
update (type, id, property, value): Object (9)	Actualiza el valor de la propiedad property un objeto del sistema
update (type, name, property, value): Object	Igual que en (9) pero el objeto es referido por nombre
Librería *society*	
Tarea	**Descripción**
addActorPermission (actor, permission): Actor	Añade un permiso a un actor
addActor (actor, groupTemplate): GroupTemplate	Añade un actor ya creado a una plantilla de grupo
addChildGroup (childGroup, parentGroup): Group	Añade un grupo a otro grupo como hijo
addChildGroupTemplate (childTemplate, parentTemplate): GroupTemplate	Añade una plantilla de grupo a otra como hija
bindUserToActor (user, actor, group): Group	Asigna un actor a un usuario dentro de un grupo
createActor (name, description): Group	Crea un nuevo actor
createGroup (name, description, groupTemplate, parentGroup): Group	Crea un nuevo grupo
createGroupTemplate (name, descr, min, max, template): GroupTemplate	Crea una nueva plantilla de grupo
createUser (login, password, actor): User	Crea un nuevo usuario
deleteActor (actor): void	Borra un actor del sistema
deleteAllActors (): void	Borra todos los actores definidos en el sistema
deleteAllGroups (): void	Borra todos los grupos definidos en el sistema
deleteAllGroupTemplates (): void	Borra todas las plantillas de grupo definidas en el sistema
deleteAllUsers (): void	Borra todos los usuarios registrados en el sistema
deleteGroup (group, cascade): void	Borra un grupo. Si *cascade* es true borra también sus hijos
deleteGroupTemplate (groupTemplate, cascade): void	Borra una plantilla de grupo. Si *cascade* es true borra también sus hijas
deleteUser (user): void	Borra un usuario
dropActorPermission (actor, permission): Actor	Niega un permiso a un actor
dropActor (actor, groupTemplate): GroupTemplate	Borra un actor de una plantilla de grupo
dropChildGroup (childGroup, group, cascade)	Borra un grupo hijo de otro grupo. Si *cascade* es true borra también sus hijos
dropChildGroupTemplate (childTemplate, template, cascade):GroupTemplate	Borra una plantilla hija de otra. Si *cascade* es true borra también sus hijas
getActor (id): Actor	Recupera un actor por identificador
getActor (name): Actor	Recupera un actor por nombre
getAllActors (): List <Actor>	Recupera la lista de todos los actores del sistema
getAllGroups (): List <Group>	Recupera la lista de todos los grupos del sistema
getAllGroupTemplates (): List <GroupTemplate>	Recupera la lista de todas las plantillas de grupo del sistema
getAllUsers (): List <User>	Recupera la lista de todos los usuarios del sistema
getGroup (id): Group	Recupera un grupo por identificador
getGroup(name): Group	Recupera un grupo por nombre

Librería *society* (continuación)	
Tarea	**Descripción**
getGroupTemplate(id): GrouTemplate	Recupera una plantilla de grupo por identificador
getGroupTemplate (name): GroupTemplate	Recupera una plantilla de grupo por nombre
getRootGroup (): Group	Recupera el grupo raíz de la jerarquía de grupos del sistema
getRootGroupTemplate (): GroupTemplate	Recupera la plantilla raíz de la jerarquía de plantillas de grupo del sistema
getUser (id): User	Recupera un usuario por identificador
getUser (login): User	Recupera un usuario por nombre
unbindAllUsersFromActor (actor, group): Group	Borra todas las asignaciones a actores dentro de un grupo
unbindUserFromActor (user, actor, group): Group	Borra la asignación de un usuario a un actor dentro de un grupo
Librería *collaboration*	
Tarea	**Descripción**
addActivity (activity, project): Project	Añade una actividad a un proyecto
addActivityTemplate (activityTemplate, projectTemplate): ProjectTemplate	Añade una plantilla de actividad a una plantilla de proyecto
addReference (reference, activityTemplate): ActivityTemplate	Añade una referencia a una plantilla de actividad
addRoleProperty (role, property, value): Role	Añade una propiedad a un rol ya definido
addRole (Role, activityTemplate): ActivityTemplate	Añade un rol ya definido a una plantilla de actividad
addService (service, activity): Activity	Añade un servicio ya definido a una actividad
addServiceTemplate(service, activityTemplate): ActivityTemplate	Añade un servicio a una plantilla de actividad
bindUserToRole (user, role, activity): Activity	Asocia un rol a un usuario dentro de una actividad
createActivity (ActivityTemplate): Activity	Crea una nueva actividad
createActivityTemplate (name, description): ActivityTemplate	Crea una nueva plantilla de actividad
createProject (ProjectTemplate, group): Project	Despliega una plantilla de proyecto sobre el espacio de trabajo de un grupo
createProjectTemplate (name, description): ProjectTemplate	Crea una nueva plantilla de proyecto
createReference (name, description, uri): Reference	Crea una nueva referencia
createRole (name, description): Role	Crea un nuevo rol
createService (type, label, description, parameters)	Crea un nuevo servicio
deleteActivity (activity, project): Project	Borra una actividad de un proyecto
deleteActivityTemplate (activityTemplate, projectTemplate): ProjectTemplate	Borra una plantilla de actividad de una plantilla de proyecto
deleteAllActivities (): void	Elimina todas las actividad del sistema
deleteAllActivityTemplates (): void	Elimina todas las plantillas de actividad del sistema
deleteAllProjects (): void	Elimina todos los proyectos del sistema
deleteAllProjectTemplates (): void	Elimina todas las plantillas de proyecto del sistema
deleteAllReferences (): void	Elimina todas las referencias del sistema
deleteAllRoles (): void	Elimina todos los roles del sistema
deleteProject (project): void	Elimina un proyecto del sistema
deleteProjectTemplate (projectTemplate): void	Elimina una plantilla de proyecto del sistema
deleteReference (reference, activity): void	Elimina una referencia de una actividad
deleteRole (role, activityTemplate): ActivityTemplate	Elimina un rol de una plantilla de actividad
deleteRoleProperty (role, property): Role	Elimina una propiedad de un rol
deleteService (service, activityTemplate): ActivityTemplate	Elimina un servicio de una plantilla de actividad
getActivity (id): Activity	Recupera una actividad por identificador
getActivity (name, project): Project	Recupera una actividad por nombre asociada a un proyecto
getActivityTemplate (id): ActivityTemplate	Recupera una plantilla de actividad por identificador
getActivityTemplate (name): ActivityTemplate	Recupera una plantilla de actividad por nombre
getAllActivities (): List <Activity>	Recupera una lista con todas las actividades desplegadas en el sistema
getAllActivityTemplates (): List <ActivityTemplate>	Recupera una lista con todas las plantillas de actividad definidas
getAllProjects (): List <Project>	Recupera una lista con todos los proyectos desplegados en el sistema
getAllProjectTemplates (): List <ProjectTemplate>	Recupera una lista con todas las plantillas de proyecto definidas
getProject (id): Project	Recupera un proyecto por id
getProjectTemplate (id): ProjectTemplate	Recupera una plantilla de proyecto por id
getProjectTemplate (name): ProjectTemplate	Recupera una plantilla de proyecto por nombre

Librería *tools*	
Tarea	**Descripción**
callService (url, operation, parameters): Object	Invoca el servicio de una herramienta externa publicado como servicio Web
readBin (bytes): String	Convierte una cadena de bytes de un fichero a una cadena de texto
readCsv (cvs)	Interpreta ficheros en formato CSV (Valores Separados por Comas)
readXml (xml)	Interpreta ficheros en formato XML
readZip (zip)	Interpreta ficheros comprimidos según el estándar ZIP

Tabla 3.3. Familia de tareas de dominio.

Reglas y eventos en P#

Hasta ahora hemos descrito todos los elementos que es necesario conocer para escribir scripts de adaptación en P#. Sin embargo, como apuntábamos al inicio de esta sección, en la especificación de transformaciones, también es posible indicar los momentos exactos, dentro del flujo de trabajo instruccional, en los cuales se deberán aplicar dichos scripts. Esta característica permite establecer formalmente un plan de ejecución temporal de las transformaciones adaptativas que automatiza, en gran medida, el proceso de desarrollo de las experiencias de aprendizaje.

Para dar soporte a esta característica el subsistema de intervención hace uso de un mecanismo de especificación basado en **reglas** y dirigido por **eventos**. Los eventos vienen a tipificar cada una de las situaciones pedagógicamente relevantes que pueden ocurrir dentro del flujo de trabajo de instrucción y a las cuales puede interesar asociar un script de adaptación para su tratamiento. Esta asociación se expresa en términos de reglas. Una regla es un artefacto que indica el script que debe lanzarse a ejecución cuando se produce un evento dentro del flujo de instrucción y ciertas condiciones ambientales son encontradas. De esta manera, puede considerarse que toda regla del subsistema de intervención relaciona tres elementos fundamentes. A continuación describimos brevemente cada uno de ellos:

- **Evento asociado a la regla.** Las reglas se encuentran asociadas a tipos de eventos que representan, en su mayoría, situaciones prototípicas de interacción que los usuarios hacen sobre la plataforma, como por ejemplo cambiar el estado de una actividad, acceder a un espacio de trabajo, o entrar a formar parte de un grupo. Cada evento, proporciona cierta información que caracteriza la situación que produjo el disparo del mismo. Para acceder a esta información, P# proporciona, como variable implícita de dominio, el objeto *event* cuyos atributos, llamados variables implícitas de evento, son accesible a través del operador de punto. Nótese que la variable *event* sólo está disponible en P# cuando se dispara un evento. La colección de tipos de eventos gestionados por Pelican se muestra en la tabla 3.4.
- **Condiciones de disparo de la regla.** Las condiciones de disparo de una regla permiten identificar las situaciones ambientales que deben satisfacerse para que el script de adaptación asociado a la misma se lance a ejecución. Estas condiciones se especifican en Pelican a través de una expresión lógica cuyos elementos constituyentes son, comúnmente, valores constantes, variables implícitas o variables implícitas de evento.

Eventos de *Pelican*		
Tipo	**Lanzado cuando...**	**Atributos**
pelican.core.startUp	Se arranca el sistema	
pelican.core.shutDown	Se detiene el sistema	
pelican.core.timeOver	Termina un cronometro	*Parámetros proporcionados en la tarea startTimer*
pelican.core.login	El usuario entra en el sistema	
pelican.core.logout	El usuario sale del sistema	
pelican.design.actor.new	Se crea un nuevo actor	- actor: El nuevo actor creado
pelican.design.actor.delete	Se elimina un actor	- actor: El actor eliminado
pelican.design.actor.update	Se actualiza un actor	- actor.old: estado del actor antes de la actualización - actor.new: estado del actos tras la actualización
pelican.design.groupTemplate.new	Se crea una nueva plantilla de grupo	- groupTemplate: La plantilla de grupo creada
pelican.design.groupTemplate.delete	Se elimina una nueva plantilla de grupo	- groupTemplate: La plantilla de grupo eliminada
pelican.design.groupTemplate.update	Se actualiza una plantilla de grupo	- groupTemplate.old: Plantilla antes de la actualización - groupTemplate.new: Plantilla tras la actualización
pelican.design.groupTemplate.addActor	Se añade un actor a una plantilla de grupo	- actor: El actor añadido - groupTemplate.old: Plantilla antes de añadir - groupTemplate.new: Plantilla después de añadir
pelican.design.groupTemplate.dropActor	Se quita un actor a una plantilla de grupo	- actor: El actor quitado - groupTemplate.old: Plantilla antes de quitar actor - groupTemplate.new: Plantilla después de quitar actor
pelican.management.user.new	Se crea un nuevo usuario	- user: El nuevo usuario
pelican.management.user.delete	Se elimina un usuario	- user: El usuario borrado
pelican.management.user.update	Se actualiza un usuario	- user.old: El usuario antes de la actualización - user.new: El usuario tras la actualización
pelican.management.group.new	Se crea un nuevo grupo	- group: El nuevo grupo creado
pelican.management.group.delete	Se elimina un grupo	- group: El grupo borrado
pelican.management.group.update	Se actualiza un grupo	- group.old: El grupo antes de la actualización - group.new: El grupo tras la actualización
.management.group.addUser	Se añade un usuario a un grupo	- user: El usuario añadido al grupo - actor: El actor que encarna el usuario - group.old: El grupo antes de añadir al usuario - group.new: El grupo tras añadir al usuario
.design.projectTemplate.new	Se crea una nueva plantilla de proyecto	- projectTemplate: La nueva plantilla de proyecto
pelican.design.projectTemplate.delete	Se borra una plantilla de proyecto	- projectTemplate: La plantilla de proyecto creada
pelican.design.projectTemplate.update	Se actualiza una plantilla de proyecto	- projectTemplate.old: La plantilla antes de actualizar - projectTemplate.new: La plantilla tras actualizar
pelican.design.projectTemplate.addActivityTemplate	Se añade una plantilla de actividad a una plantilla de proyecto	- activityTemplate: La plantilla de actividad añadida - projectTemplate.old: La plantilla antes de añadir - projectTemplate.new: La plantilla tras añadir
pelican.design.projectTemplate.dropActivityTemplate	Se quita una plantilla de actividad a una plantilla de proyecto	- activityTemplate: La plantilla de actividad quitada - projectTemplate.old: La plantilla antes de quitar - projectTemplate.new: La plantilla tras quitar
pelican.design.activityTemplate.new	Se crea una nueva plantilla de actividad	- activityTemplate: La nueva plantilla creada
pelican.design.activityTemplate.delete	Se elimina una plantilla de actividad	- activityTemplate: La plantilla eliminada
pelican.design.activityTemplate.update	Se actualiza una plantilla de actividad	- activityTemplate.old: La plantilla antes de actualizar - activityTemplate.new: La plantilla tras actualizar
pelican.design.activityTemplate.newReference	Se crea una nueva referencia en una plantilla de actividad	- activityTemplate.old: La plantilla antes de la creación - activityTemplate.new: La plantilla tras la creación - reference: La nueva referencia creada
pelican.design.activityTemplate.deleteReference	Se elimina una referencia en una plantilla de actividad	- activityTemplate.old: La plantilla antes de la eliminación - activityTemplate.new: La plantilla tras la eliminación - reference: La referencia eliminada
pelican.design.activityTemplate.newRole	Se crea un nuevo rol en una plantilla de actividad	- activityTemplate.old: La plantilla antes de la creación - activityTemplate.new: La plantilla tras la creación - role: El rol creado

Eventos de *Pelican* (continuación)		
Tipo	**Lanzado cuando...**	**Atributos**
pelican.design.activityTemplate.deleteRole	Se elimina un rol en una plantilla de actividad	- activityTemplate.old: La plantilla antes de la eliminación - activityTemplate.new: La plantilla tras la eliminación - role: El rol eliminado
pelican.design.activityTemplate.updateRole	Se actualiza un rol de una plantilla de actividad	- role.old: El rol antes de la actualización - role.new: El rol tras la actualización - activityTemplate: La plantilla de actividad
pelican.design.activityTemplate.newService	Se crea un nuevo servicio dentro de una plantilla de actividad	- activityTemplate.old: La plantilla antes de la creación - activityTemplate.new: La plantilla tras la creación - service: El nuevo servicio creado
pelican.design.activityTemplate.deleteService	Se elimina un servicio de una plantilla de actividad	- activityTemplate.old: La plantilla antes de la eliminación - activityTemplate.new: La plantilla tras la eliminación - service: El servicio eliminado
pelican.design.activityTemplate.updateService	Se actualiza un servicio de una plantilla de actividad	- service.old: El servicio antes de la actualización - service.new: El servicio tras la actualización - activityTemplate: La plantilla de actividad
pelican.design.activityTemplate.manageService	En servicios que se integran a nivel D, cuando se arranca o para un servicio	- service.old: El servicio antes de la acción - service.new: El servicio después de la acción - activityTemplate: La plantilla de actividad
pelican.management.project.new	Se despliega un nuevo proyecto sobre el espacio de trabajo de un grupo	- project: El nuevo proyecto creado - group: El grupo que desarrolla el proyecto
pelican.management.project.delete	Se repliega un proyecto del espacio de trabajo de un grupo	- project: El proyecto replegado - group: El grupo que desarrolló el proyecto
pelican.management.project.update	Se actualiza el estado de un proyecto	- project.old: El proyecto antes de la actualización - project.new: El proyecto tras la actualización
pelican.management.project.addActivity	Se despliega una actividad de un proyecto	- project.old: El proyecto antes del despliegue - project.new: El proyecto tras el despliegue - activity: La nueva actividad creada
pelican.management.project.dropActivity	Se repliega una actividad de un proyecto	- project.old: El proyecto antes del repliegue - project.new: El proyecto tras el repliegue - activity: La actividad replegada
pelican.management.activity.update	Se actualiza una actividad	- activity.old: La actividad antes de la actualización - activity.new: La actividad tras la actualización
pelican.workspace.accessToWorkspace	El usuario cambia de espacio de trabajo	- fromWorkspace: Espacio de trabajo de origen - toWorkspace: Espacio de trabajo de destino
pelican.workspace.accessToProject	El usuario accede a un proyecto	- project: El proyecto al que accede el usuario
pelican.workspace.accessToActivity	El usuario accede a una actividad	- activity: La actividad a la que accede el usuario
pelican.workspace.addService	Se añade un servicio a un espacio de trabajo	- workspace: El espacio sobre el que se añade el servicio - service: El servicio añadido
pelican.workspace.deleteService	Se elimina un servicio de un espacio de trabajo	- workspace: El espacio sobre el que se borra el servicio - service: El servicio eliminado
pelican.workspace.updateService	Se actualiza un servicio	- workspace: El espacio donde reside el servicio - service.old: El servicio antes de la actualización - service.new: El servicio tras la actualización
pelican.workspace.manageService	En servicios que se integran a nivel D, cuando se arranca o para un servicio	- service.old: El servicio antes de la acción - service.new: El servicio después de la acción - activityTemplate: La plantilla de actividad
pelican.workspace.invokeService	El usuario invoca un servicio dentro de un espacio de trabajo	- workspace: El espacio donde reside el servicio - service: El servicio invocado
pelican.workspace.activity.invokeReference	El usuario abre una referencia	- activity: La actividad donde se abre la referencia - reference: La referencia abierta
pelican.workspace.activity.invokeService	El usuario invoca un servicio dentro de una actividad	- activity: La actividad donde reside el servicio - service: El servicio invocado
pelican.workspace.activity.updateState	Se actualiza el estado de una actividad (su ciclo de vida)	- activity.old: La actividad antes de la actualización - activity.new: La actividad tras la actualización
pelican.workspace.activity.updateRole	Se actualiza la asignación de roles de una actividad	- activity.old: La actividad antes de la actualización - activity.new: La actividad tras la actualización

Tabla 3.4. Colección de tipos de eventos gestionados por Pelican.

- **Script de adaptación de la regla.** La regla también hace referencia a un script de adaptación previamente definido. Este script se lanza a ejecución automáticamente cada vez que se produzca, dentro del flujo de trabajo instruccional, un evento del tipo definido en la regla y que las condiciones de disparo se verifiquen.

Hay que advertir que los scripts de adaptación asociados a las reglas permiten llevar a cabo transformaciones sobre los modelos de la plataforma. Sin embargo, estas transformaciones no implican en ningún caso el disparo de nuevas reglas ya que para que una regla se dispare es necesario que la plataforma lance un evento y, por construcción, los eventos sólo se lanzan cuando se producen interacciones manuales sobre la plataforma a través de sus interfaces de usuario Web.

Para ilustrar las capacidades de transformación adaptativa del lenguaje P# consideremos el ejemplo del listado 3.1. En él se expresa un script en P# que tiene por objeto enviar un correo electrónico a todos los miembros de un grupo instándoles a que continúen con el desarrollo de la siguiente actividad una vez que la actual ya ha sido aprobada.

Listado 3.1. Ejemplo de script en P# para dar soporte a la evaluación.

```
def subject = 'Actividad aprobada'
def body = 'La actividad $activity ha sido aorobada.' +
           'Podéis comenzar la siguiente actividad'
def students = group.getActorBinding ('estudiante').users
def teacher = group.getActorBinding ('profesor').users[0]
for (student in students)
  common.sendMail (teacher, student, subject, body)
```

Como se puede apreciar, este sencillo script utiliza dos parámetros, *activity* y *group*, instancias de los artefactos del mismo nombre presentados en las secciones 3.4.1 y 3.3.1. Estos parámetros deberán ser resueltos en tiempo de desarrollo, durante la aplicación del script, para referenciar, respectivamente, la actividad recién aprobada y el grupo que la desarrolló. Las cuatro primeras instrucciones del script declaran variables locales que se utilizarán a lo largo del mismo. Como puede apreciarse, ninguna de ellas ha sido explícitamente tipificada. P# es capaz de inferir su tipo evaluando el tipo de la expresión a la derecha de la asignación. El parámetro *activity* es utilizado en la cadena asignada a la variable *body* para ser convertida a texto e incluido dentro de la misma. Esto se consigue anteponiendo el operador *$* al nombre del parámetro en la cadena. Por su parte la variable *students* recoge todos los usuarios que encarnan el actor *estudiante* en el grupo que es pasado como parámetro. Similarmente, *teacher* es la variable utilizada para representar al profesor. Una vez que disponemos de la colección de los estudiantes, recorremos la misma para ejecutar la tarea *sendMail* para cada uno de ellos que permite enviar un correo electrónico desde un remitente, el profesor, a un destinatario, cada estudiante. Como se desprende del ejemplo anterior, los estudiantes en concreto que reciben el correo electrónico, el profesor que aparece como remitente y el cuerpo del mismo depende de los objetos vinculados a los parámetros del script. Esto pone de manifiesto nuestra afirmación anterior

que definía a P# como un mecanismo flexible para especificar en términos abstractos scripts de adaptación cuya intervención resultante dependen del contexto donde se apliquen.

Supongamos ahora que deseamos que el script del listado 3.1, se ejecute automáticamente cada vez que el profesor cambie el estado de cualquier actividad desde *pendiente de evaluación* hasta *aprobada*. Para llevar esto a cabo, podemos declarar una regla cuyos elementos constituyentes son detallados en el listado 3.2. Como se puede ver, el tipo de evento al que está asociada la regla identifica una situación de cambio de estado de cualquier actividad. Es decir, esta regla será revisada para su potencial ejecución cada vez que un usuario en la plataforma lleve a cabo una transición de estado de una actividad. La condición de disparo expresa que el script de adaptación asociado sólo debe aplicarse si la transición de estados de la actividad indica que ésta ha sido aprobada. Esto es, si se ha transitado desde el estado *pendiente de evaluación* hasta *aprobada*. Como se ve, toda la información proveniente del evento está accesible a través de la variable implícita *event.* En concreto, para este tipo de evento, se proporcionan dos objetos de datos (oldActivity y newActivity) que capturan el estado de la actividad antes y después de realizan la transición.

Listado 3.2. Regla que aplicar el script 3.1 al terminar cada actividad.

```
Evento: pelican.workspace.activity.updateState
Condicion: event.oldActivity.state == 'PENDING' &&
           event.newActivity.state == 'PASSED'
Script: def activity = event.newActivity ❶
        def group = event.newActivity.project.workspace.owner ❷
        def subject = 'Actividad aprobada'
        def body = 'La actividad activity ha sido aprobada.' +
                   'Podéis comenzar la siguiente actividad'
        def students = group.getActorBinding ('estudiante').users
        def teacher = group.getActorBinding ('profesor').users[0]
        for (student in students)
          common.sendMail (teacher, student, subject, body)
```

En relación al script de adaptación asociado a la regla, los cambios son menores con respecto al que aparecía en el listado 3.1. Simplemente, en este caso hemos contextualizado su uso asignando un valor real a los parámetros del script conectándolo con información proveniente del evento. En efecto, la actividad ahora se corresponderá con newActivity (❶) mientras que el grupo puede ser calculado navegando sobre los modelos internos de la plataforma. Desde la actividad accedemos al proyecto que la contiene, de ahí al espacio de trabajo donde éste se ha desplegado y de ahí obtenemos el grupo preguntando por el propietario del mismo (❷).

Estos dos artefactos en su conjunto – scripts de adaptación y reglas políticas – permiten capturar los aspectos dinámicos en una especificación computable de un escenario de aprendizaje, tal y como se mostrará en el capítulo siguiente al discutir los escenarios de prueba de esta tesis. En cierto sentido, puede pensarse que el papel

que juegan estos elementos es complementario. Los scripts por un lado constituyen un mecanismo para expresar procedimientos de transformación adaptativos reutilizables en diferentes contextos socio–pedagógicos mientras que las reglas son los elementos encargados de adaptar la aplicación de los mismos a cada contexto específico. Sin embargo, como se verá en la subsección 3.5.1, donde se entrará a discutir en profundidad los elementos de este subsistema, las reglas también son elementos parametrizables y por tanto, candidatos para la reutilización.

Aplicación de transformaciones

Durante la fase de desarrollo de las experiencias colaborativas todas las especificaciones de carácter dinámico deben ser aplicadas puntualmente para adaptar la plataforma a los requerimientos de cambio del flujo de trabajo instruccional. En este sentido, Pelican proporciona los mecanismos pertinentes para llevar a cabo esta aplicación. En concreto podemos distinguir dos mecanismos alternativos. Por un lado es posible realizar una aplicación manual de los scripts de adaptación y por otro se pueden aprovechar los mecanismos de automatización del subsistema de intervención que, como vimos, utilizan un conjunto de reglas políticas establecidas en tiempo de diseño, para conocer el momento exacto dentro del flujo de instrucción en el que los scripts deben ser aplicados. A lo largo de los siguientes subapartados describimos estos mecanismos en detalle.

Aplicación manual de scripts.

La manera más sencilla de llevar a cabo la aplicación de los scripts de adaptación consiste en la ejecución manual de los mismos. De esta manera, si en un determinado momento del desarrollo de una experiencia, el monitor detecta que se ha producido un acontecimiento relevante que requiere cierto tratamiento, éste puede seleccionar, de entre la colección de scripts de adaptación predefinidos, aquél más apropiado y lanzarlo a ejecución manualmente a través de las facilidades que para ello proporciona el subsistema de intervención. Por contrapartida, el inconveniente que supone esta forma de trabajo es que se requiere gran cantidad de personal monitorizando las actividades que se están desarrollando dentro de cada grupo en la plataforma.

Para dar soporte a la aplicación manual de scripts de adaptación, Pelican proporciona un entorno de ejecución que los monitores pueden utilizar discrecionalmente. La ejecución de un script en este entorno es un proceso de cuatro fases consecutivas que se repite iterativamente para cada una de las instrucciones del mismo. Adviértase que P# es un lenguaje interpretado. A continuación describimos brevemente cada una de estas fases de ejecución, si bien todo ello resulta transparente para los usuarios de la plataforma:

- **Fase de selección de la siguiente instrucción.** El primer paso para proceder con la ejecución de un script es determinar la siguiente instrucción que, dentro del ciclo iterativo de interpretación, debe ser procesada. Este trabajo es realizado internamente por el analizador sintáctico de P# que es capaz de descomponer los scripts de acuerdo a una secuencia ordenada de instrucciones.

- **Fase de contextualización de la instrucción.** Una vez identificada la instrucción que debe ser procesada, la fase de contextualización se encarga de resolver a valores reales cada una de las referencias abstractas que aparecen dentro la misma. Para dar soporte a esta fase Pelican mantiene un **contexto de usuario** asociado a cada uno de los usuarios registrados en la plataforma. Un contexto de usuario es un almacén que contiene información actualizada del estado en que se encuentra un usuario dentro del entorno. Para ejecutar un script, lo único que debe hacer el monitor sobre el entorno de ejecución es seleccionar el contexto de usuario sobre el cuál se desea contextualizar el mismo. La información mantenida en los contextos de usuario se expresa en términos de dos tipos de variables:
  - *Variables implícitas de dominio.* El contexto de usuario es el encargado de mantener actualizada la colección de variables implícitas de dominio que se describieron en el apartado anterior y que, como dijimos, son una representación del estado del usuario dentro del entorno (ver tabla 3.2).
  - *Variables definidas por el usuario.* La aplicación de scripts sobre un determinado contexto de usuario puede suponer la aparición de nuevas variables dentro del mismo. En efecto, el lenguaje P# contiene algunas primitivas que permiten almacenar las variables locales de un script dentro de un contexto de usuario (ver tabla 3.4).
- **Fase de interpretación de la instrucción.** Finalizada la fase de contextualización de la instrucción es posible pasar a interpretar la semántica subyacente a la misma. Este proceso consiste en identificar la colección de tareas específicas que es necesario realizar. Así por ejemplo, en la instrucción de iteración del listado 3.1, la instrucción requiere enviar un correo electrónico a cada estudiante del grupo.
- **Fase de ejecución de la instrucción.** Una vez que las instrucción ha sido interpretada, ésta puede ser lanzada a ejecución para que produzcan los efectos oportunos. En el caso particular del listado 3.1, estos resultados serán, como hemos mencionado, el envío de un correo electrónico a cada uno de los estudiantes del grupo referido a través del proceso de contextualización.

La selección del contexto utilizado en la ejecución de un script puede tener una incidencia directa en el resultado obtenido. Por ejemplo, si aplicamos el script del listado 3.1 sobre algún miembro del grupo A de la figura 3.8, la contextualización provocará que se envíen 3 correos electrónicos a Luis, María y Pedro. Por el contrario, si se selecciona un miembro del grupo B, entonces los alumnos que recibirán el correo serán Oscar, Pedro e Isaac.

Además la capacidad de almacenar variables definidas por el usuario dentro de un contexto resulta muy ventajosa en la práctica, ya que permite a los scripts aplicados secuencialmente sobre un mismo contexto comunicarse entre sí a través de la compartición de variables. Para ilustrar el papel que juegan los contextos de usuario en este sentido, considérese el caso de un escenario formado por tres actividades (A1, A2 y A3) pertenecientes a un proyecto P. Un requerimiento de este escenario es que

los estudiantes desarrollen estas actividades en el orden en que aparecen declaradas en el proyecto. Para hacer frente a ello, el monitor tiene que activar y desactivar las mismas según éstas se vayan desarrollando para garantizar que en cada momento los estudiantes solamente tengan acceso a una actividad. En lugar de gestionar este problema manualmente a través de la interfaz de administración de actividades (figura 3.15) el monitor decide utilizar los scripts de adaptación de la figura 3.20 para que realicen esta labor automáticamente. En concreto, el script S1 desactiva todas las actividades del proyecto menos la primera (A1) e introduce en el contexto un índice i con valor 0 para indicar que la actividad accesible en ese momento es la primera de la lista de actividades del proyecto. Por su parte, el script S2, cada vez que se aplica, desactiva la actividad con índice i en la lista de actividades del proyecto y activa la actividad con índice i + 1, incrementando además este valor de i en una unidad.

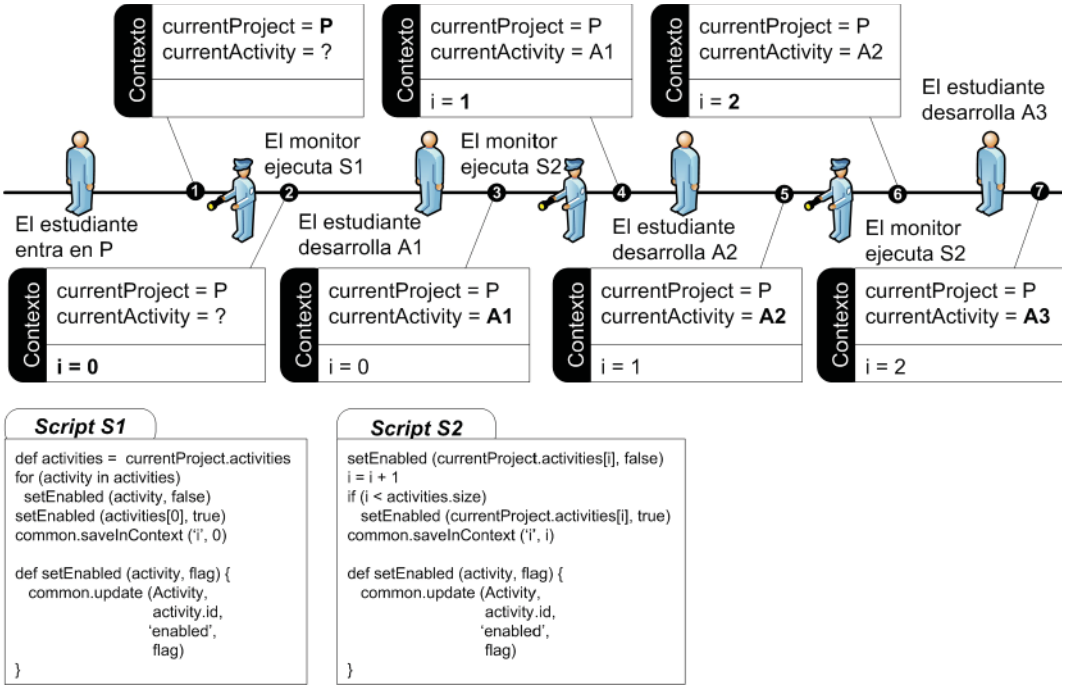

Figura 3.20. Proceso de ejecución de scripts dentro de un contexto.

La imagen de la figura también ilustra la secuencia de acontecimientos que tienen lugar a lo largo del transcurso del escenario y el estado del contexto de uno de los estudiantes involucrados durante el desarrollo de la experiencia. La línea gruesa horizontal representa el eje de tiempo desde el inicio hasta el final de la misma. Como se puede apreciar, el primer acontecimiento relevante consiste en que el estudiante objeto de estudio abre el proyecto dentro del espacio de trabajo compartido. Esto hace que la variable currentProject adquiera el valor P (❶). En ese momento, la variable currentActivity no tiene valor puesto que el estudiante aún no ha abierto ninguna actividad ni tampoco existen variables definidas por el usuario en el contexto puesto que todavía no se ha ejecutado ningún script. Antes de que el estudiante seleccione cualquier actividad, el monitor ejecuta el script S1 con lo que ya solamente A1 resulta accesible. Tras esto, la variable currentActivity sigue sin valor ya que el estudiante aún

no ha abierto ninguna actividad pero el contexto incluye ahora la variable i con valor 0 (❷). Tras esto, el estudiante abre la actividad A1 y la desarrolla colaborativamente con el resto de sus compañeros. En ese momento (❸), la variable currentActivity es A1 e i sigue valiendo 0 ya que nadie ha alterado su valor en el contexto. Cuando el monitor detecta que los estudiantes han terminado de desarrollar la actividad A1, por ejemplo porque observa que éstos han transitado el estado de la misma de *en desarrollo* a *pendiente de evaluación*, lanza manualmente a ejecución el script S2 por primera vez. Este script utiliza el valor de i para saber qué par de actividades consecutivas debe desactivar y activar respectivamente e incrementa en una unidad el valor de esta variable sobre el contexto para la próxima vez que S2 sea ejecutado. En ese momento (❹), la variable implícita currentActivity sigue apuntando a A1 puesto que el estudiante aún no ha abandonado esta actividad en su espacio de trabajo. Sin embargo, el script ha desactivado A1, ha activado A2 y ha puesto el valor de i a 1 para que apunte a la actividad segunda de la lista de actividades del proyecto (A2). Esto permite al estudiante entrar a desarrollar la actividad A2 (❺). Este proceso se repite posteriormente para desactivar A2 y activar A3 (❻ y ❼).

Este no es más que un sencillo ejemplo de cómo se pueden utilizar las prestaciones del subsistema de intervención para implantar en Pelican un escenario con una lógica de secuenciamiento lineal entre sus actividades. En el capítulo siguiente, donde presentaremos diferentes resultados de la pruebas realizadas para este trabajo de tesis, se pueden encontrar mejores soluciones para este problema. No obstante, el ejemplo pone de manifiesto aquí dos consideraciones relevantes. En primer lugar, cómo las variables implícitas de un contexto de usuario (*currentProject* y *CurrentActivity*) se ven afectadas por los movimientos e interacciones que éste haga sobre la plataforma. Y en segundo lugar, cómo las variables definidas por el usuario (*i*) permiten poner en contexto la ejecución de un script para que se ajuste a la situación actual e incluso suponen un mecanismo de transferencia de información indirecta entre scripts ejecutados secuencialmente sobre el mismo contexto de usuario. En este sentido cabe afirmar que los algoritmos complejos de adaptación, que requieran de la aplicación de una familia de transformaciones secuenciales en distintos momentos del tiempo, se distribuyen en una colección de scripts que permanecen cohesionados gracias a las transferencias de información soportadas por la compartición de variables definidas por el usuario sobre el mismo contexto.

En estas situaciones, es necesario hacer uso siempre del mismo contexto para lanzar los scripts a ejecución. Volviendo al ejemplo anterior, si el monitor hubiera ejecutado los scripts S1 y S2 en un principio sobre el contexto de un miembro del grupo y luego hubiera decidido realizar el resto de aplicaciones del script S2 sobre un contexto de usuario diferente, se produciría un error porque la variable i no estaría definida dentro de este segundo ya que sobre él no se ejecutó S1. En situaciones como esta, donde la gestión del trabajo de un grupo se lleva a cabo a través de la aplicación de scripts, es conveniente que el monitor elija un miembro del grupo como representante para trabajar en exclusividad con su contexto de usuario o modificar el script para sincronizar todas las variables de usuario en los contextos de los miembros del grupo.

En definitiva podemos afirmar que esta infraestructura de contextos de usuario facilita el trabajo de contextualización de los scripts de adaptación. En efecto, los monitores sólo tienen que seleccionar un script y lanzarlo a ejecución sobre un determinado contexto de usuario a través del entorno de ejecución. La contextualización sobre contextos de usuarios no supone una limitación en la interpretación de scripts sino más bien es una consecuencia lógica de diseño. No parecería razonable resolver cada referencia a una variable dentro de un script a valores reales arbitrarios. En este sentido, los contextos de usuarios proporcionan el grado de cohesión necesario entre las variables del mismo para que la contextualización tenga un valor semántico apreciable.

Aplicación automáticas de scripts mediante reglas.

Cuando las especificaciones de los aspectos dinámicos definidas por los diseñadores instruccionales incluyen una colección de reglas que determinan exactamente cuándo, y bajo que condiciones debe aplicarse cada script definido, la aplicación de los mismos se puede automatizar. Este mecanismo es más ágil en su aplicación pero conlleva más esfuerzos durante la fase de diseño ya que es preciso, como discutimos anteriormente, alinear la aplicación de cada script de adaptación con el tipo de evento correspondiente de aquéllos proporcionados por la plataforma. En efecto, en la aplicación manual, los scripts son lanzados a ejecución cuando los monitores lo consideran oportuno mientras que en la aplicación basada en reglas, los scripts asociados a las mismas se lanzan a ejecución automáticamente cuando la plataforma dispara un evento del tipo reflejado en las reglas y siempre que se cumplan las condiciones ambientales expresadas en las mismas.

En este sentido, el valor diferencial más relevante es que en la aplicación manual de scripts la observación y tratamiento de los acontecimientos del flujo de trabajo instruccional se lleva a cabo por los monitores mientras que en la aplicación basada en reglas, los acontecimientos se encuentran tipificados en tipos de eventos para poder asociar reglas de tratamiento a los mismos. Esta diferencia trae consigo ciertas consecuencias relacionadas con el proceso de ejecución automática de scripts. Nuevamente aquí podemos enumerar una colección de fases que son atravesadas cada vez que se produce un acontecimiento pedagógicamente relevante dentro del flujo de instrucción. A continuación describimos en detalle cada una de estas fases:

- **Fase de escucha.** El proceso de ejecución de scripts de forma automática comienza en la fase de escucha. En esta fase, la plataforma se encuentra a la espera de que se produzca un acontecimiento relevante dentro del flujo de trabajo instruccional. Como ya dijimos con anterioridad estos acontecimientos hacen referencia, esencialmente, a cualquier tipo de interacción que los usuarios hagan sobre la plataforma. Cuando un determinado acontecimiento tiene lugar, el subsistema de intervención se encarga de construir un evento que lo represente incluyendo dentro de él la familia de variables implícitas de evento requeridas para caracterizar el contexto socio–colaborativo donde éste se produce. Hay que advertir que aunque esta es la primera fase del proceso de ejecución automática, la plataforma

no abandona nunca este estado de escucha permanente para atender ordenadamente a todos los acontecimientos que puedan surgir durante el desarrollo de una experiencia.

- **Fase de selección de reglas.** Una vez que un evento es disparado, el subsistema de intervención lo recoge para su tratamiento. Este tratamiento consiste, en primer lugar, en recuperar de la batería de reglas definidas en el sistema aquéllas que se encuentren vinculadas al tipo del evento que está siendo procesado. Como resultado de esta fase, se obtiene una colección de reglas que es necesario evaluar para su potencial ejecución.

- **Fase de planificación de la ejecución.** Por motivos de seguridad, la ejecución de la colección de reglas seleccionadas en la fase anterior se realiza de forma secuencial y no paralela. Nótese que cada regla lleva asociado un script cuyas transformaciones podrían colisionar frontalmente si se ejecutasen en concurrencia. El orden exacto en que las reglas son lanzadas a ejecución queda especificado por el plan de ejecución determinado durante esta fase. Este plan se construye teniendo en cuenta el orden de prioridad que es definido en cada regla en tiempo de diseño. En este sentido, una labor importante de los diseñadores para garantizar la consistencia del sistema tras la aplicación de las reglas es asignar cuidadosamente estas prioridades ya que los efectos de la aplicación de una reglas podrían contrarrestarse con los de la aplicación de otras.

- **Fase de ejecución de una regla.** Una vez establecido un plan de ejecución, éste puede llevarse a la práctica, seleccionando por orden, una a una, todas las reglas del plan y lanzándolas a ejecución. La fase de ejecución de una regla está compuesta a su vez de dos subfases. La contextualización requerida en cada una de ellas se hace sobre el contexto que se indica en la información del evento:

  – *Evaluación del disparador.* En primer lugar es evaluada la expresión P# que expresa las condiciones de disparo de la regla lo cual incluye su contextualización sobre el contexto del usuario que provocó el evento. Si el resultado es cierto, se pasa a la subfase siguiente. La evaluación de cada disparador se realiza justo en este momento y no durante la fase de planificación para asegurarse de que la ejecución se hace atendiendo al estado actual de la plataforma.

  – *Ejecución del scripts.* Una vez que las condiciones de disparo han sido verificadas, se procede con la ejecución del script asociado contextualizándolo sobre el contexto de usuario que provocó el evento, tal y como fue descrito previamente para el caso de la ejecución manual.

El proceso anterior aparece esquematizado en la figura 3.21. En ella se observan cada una de las fases que atraviesa el subsistema de intervención cuando se produce un acontecimiento sobre las interfaces de la plataforma. En primer lugar, se recoge el evento (❶), en segundo lugar se buscan sus reglas asociadas (❷), después de construye un plan de ejecución (❸) y por último se lleva a cabo dicho plan para transformar, potencialmente, el estado de la plataforma (❹).

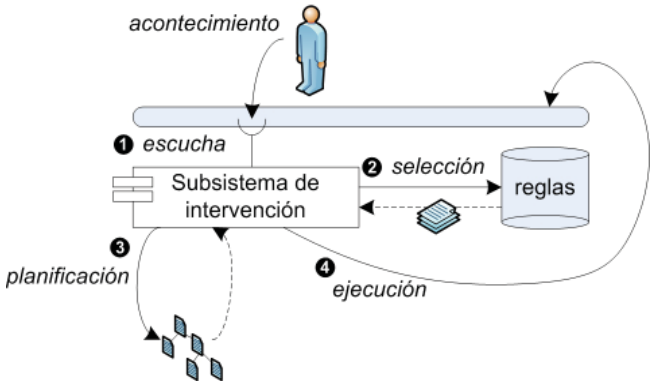

Figura 3.21. Proceso de ejecución automática de scripts basada en reglas.

De manera similar a como ocurre con los dos subsistemas anteriores, el trabajo de especificación y tratamiento de los aspectos dinámicos vinculados a los escenarios de aprendizaje colaborativo también puede ser conceptualizado en tres niveles de abstracción:

- **Nivel de lenguaje.** El nivel de lenguaje está formado por los artefactos del subsistema de intervención que permiten llevar a cabo las tareas de diseño para capturar los aspectos dinámicos del escenario. En concreto en este nivel encontramos el lenguaje P# en sí mismo y la regla de planificación que permiten la definición de scripts adaptativos y la especificación de un plan temporal de ejecución para los mismos respectivamente.

- **Nivel de modelo.** Haciendo uso de los elementos del nivel de lenguaje, los diseñadores se encargan, a este nivel, de construir un **modelo de intervención** que contemple todos los aspectos dinámicos del escenario. Este modelo está constituido por una colección de scripts que expresan algoritmos de transformación adaptativa y, opcionalmente, de una colección de reglas que permiten especificar la aplicación temporal de los mismos.

- **Nivel de aplicación.** A nivel de aplicación, los monitores, en el caso de la aplicación manual de transformaciones o la propia plataforma, en el caso de aplicación automática, ejecutan scripts adaptativos para transformar la misma en respuesta a los requerimientos de cambio del flujo de trabajo instruccional del escenario de aprendizaje.

La figura 3.22 muestra esta descomposición en tres niveles de abstracción para el caso automatizado con reglas del ejemplo que presentamos anteriormente relativo al secuenciamiento de actividades. A nivel de lenguaje disponemos de los artefactos script de adaptación y regla política. La regla R1 se utiliza para disparar la ejecución del script S1 de la figura 3.20 durante la fase de despliegue. R2 por su parte, ejecuta S2 en cada cambio de estado (ambos scripts han sido adaptados para que referencien variables implícitas del evento). A nivel de aplicación puede verse como las reglas R1 y R2 se aplican puntualmente de forma automática en respuesta a los eventos que emite la plataforma asociados a los acontecimiento de despliegue y fin de la actividad.

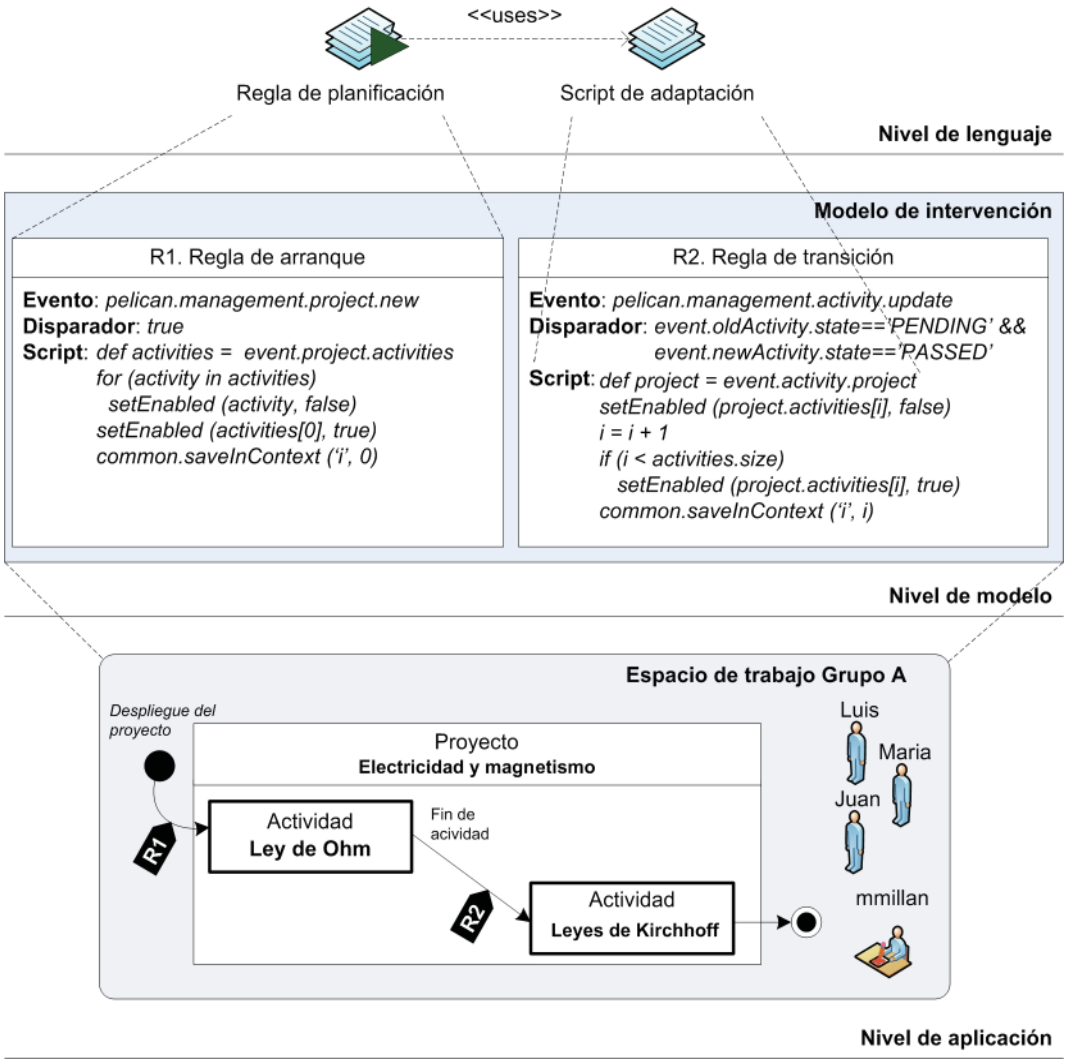

Figura 3.22. Niveles de especificación del subsistema de intervención.

La principal aportación del subsistema que presentamos en esta sección es que permite la captura de los aspectos dinámicos que aparecen inherentemente vinculados a la mayoría de escenarios de aprendizaje colaborativo. Para mantener el paralelismo con las secciones anteriores precisaremos en este punto cuales son las ventajas principales ofrecidas por los artefactos de este subsistema:

- **Obtención de un modelo de intervención abstracto.** En concreto, en la fase de diseño, los diseñadores instruccionales construyen un modelo de intervención formado por una colección de scripts de adaptación y reglas de planificación que capturan en su conjunto todos los aspectos dinámicos del escenario. Esta especificación se realiza en términos abstractos e independientemente del contexto socio–colaborativo sobre el que posteriormente se aplicarán las transformaciones gracias a las facilidades del nivel de lenguaje. Tanto es así que un mismo modelo de comportamiento puede aplicarse a diferentes contextos pedagógicos obteniéndose resultados potencialmente diferentes, como discutimos anteriormente.

- **Reutilización de los modelos de intervención.** Los elementos del modelo de intervención constituyen una colección de especificaciones que, por si mismos, resultan candidatos idóneos para la reutilización en diferentes situaciones pedagógicas. En efecto, la familia de reglas y scripts que describen los aspectos dinámicos de un escenario pueden ser utilizados para definir otros escenarios con características y requerimientos de adaptación similares.

- **Simplificación de las tareas de administración.** La expresión de las intervenciones, a través de las interfaces de usuario, facilita la labor de los administradores para mantener la plataforma constantemente adaptada a las necesidades de cada escenario. En efecto, el lenguaje de scripting proporciona un extenso conjunto de primitivas que permiten cubrir muchas de las necesidades de cambio y configuración lo que hace innecesario hacer frente a las mismas mediante costosas tareas de desarrollo realizadas por expertos tal y como ocurre en otros sistemas. Esto es especialmente notable en integración como veremos en la sección 3.6.

- **Flexibilidad en la alteración de comportamientos reactivos.** Los modelos de intervención descritos con las facilidades de este subsistema prescriben el comportamiento reactivo que debe mostrar la plataforma ante determinados estimulos. Dado que estos modelos se expresan a través de las interfaces Web, es factible alterar rápida y eficazmente estos comportamientos. Piénsese como ejemplo en la definición de dos modelos de intervención distintos que describen comportamientos alternativos de la plataforma para el mismo escenario. La conmutación entre uno y otro, posible a través de las interfaces, permitiría cambiar dinámicamente, el comportamiento de la plataforma durante la fase de desarrollo de la experiencia de manera instantánea.

- **Opcionalidad del modelo de intervención.** El modelo de intervención forma parte, junto con el de sociedad y de colaboración, de la especificación de un escenario de aprendizaje. Si estos dos últimos productos se encargan de describir dimensiones complementarias del escenario, el primero tiene por objeto completarlas en sus aspectos dinámicos de manera que puede prescindirse de su uso si se apuesta por una gestión completamente manual de las adaptaciones.

3.5.1. El subsistema de intervención

Las facilidades proporcionadas por el subsistema de intervención de Pelican son soportadas por un complejo entramado de artefactos software interrelacionados entre sí. A lo largo de esta subsección presentaremos en detalle cada uno de ellos y trataremos de justificar qué responsabilidades tienen de cara a la especificación y desarrollo de los aspectos dinámicos de un escenario [Vélez & Verdejo, 2007] [Vélez & Verdejo, 2007b] [Vélez et al., 2006] [Vélez el al., 2005b]. En concreto, podemos dividir esta familia de artefactos en 2 tipos, aquéllos que permiten la especificación y posterior ejecución de los scripts de adaptación y aquéllos que hacen posible el establecimiento de un plan para la ejecución automática de los mismos a lo largo del tiempo. Comenzaremos, en primer lugar, por el primer tipo de elementos:

- **Policy Context.** Esta entidad hace referencia a los contextos de usuario que se utilizan para llevar a cabo el proceso de contextualización de los scripts de adaptación tal y como se discutió con anterioridad. Desde el punto de vista tecnológico, un contexto de usuario es un almacén de información que contiene una colección de variables con nombres que referencian diferentes entidades de los modelos internos de cada uno de los subsistemas de la plataforma así como variables definidas por el usuario que son puntualmente insertadas en el mismo mediante la ejecución de scripts de adaptación. La característica fundamental de los contextos es que éstos disponen de un método para la actualización automática de las variables implícitas contenidas en los mismos. La estrategia de actualización por parte de la plataforma consiste en invocar a dicho método cada vez que un usuario realiza una interacción sobre ella a través de las interfaces de usuario. De esta manera, cualquier cambio motivado por dicha interacción, como puede ser la entrada en un espacio de trabajo compartido, la apertura de un proyecto o de una actividad, se ve inmediatamente reflejado en el valor de las variables implícitas. Esto es importante para que la contextualización de scripts nunca se lleve a cabo sobre una colección de variables obsoletas.

- **Policy Context Factory.** Esta entidad es una factoría que tiene como misión la construcción actualizada de contextos de usuario. De esta manera, los contextos generados por esta entidad contienen una versión actualizada de la familia de variables implícitas descritas en la tabla 3.2. Para llevar a cabo esta construcción, este artefacto toma como argumento el usuario para el que hay que construir el contexto y recolecta de la plataforma toda la información relacionada con él. El uso de la factoría de contextos está asociado a dos situación distintas. Por un lado, cuando un usuario se autentifica en Pelican es necesario crear un contexto nuevo para él y asociarlo a su sesión de trabajo. Por otro, cada vez que la plataforma invoca al método de actualización de un contexto, también se requiere hacer uso de este artefacto. En efecto, en este segundo caso, la implementación interna del método de actualización delega en la factoría de contextos la responsabilidad de obtener un nuevo contexto con valores actualizados para la variables implícitas, después copia el resto de variables del contexto antiguo sobre el nuevo contexto y finalmente reemplaza el primero por el segundo.

- **Policy Variable.** Cada una de las variables implícitas que residen en el contexto de usuario hace referencia a un elemento de los modelos de información de la plataforma (actor, usuario, grupo, actividad, proyecto, etc.). Sin embargo, introducir en el contexto directamente estos objetos de dominio resultaría inapropiado ya que viola ciertos principios de seguridad elementales. En efecto, las implementaciones de estos artefactos proporcionan mecanismos para cambiar su estado. Esta capacidad, utilizada con irresponsabilidad desde los scripts de adaptación podría causar daños irreparables en la integridad en la plataforma. Recuérdese, en este sentido, que las variables implícitas de contexto sólo tienen por objeto proporcionar información de estado para dar soporte a la contextualización de los scripts. La responsabilidad de la entidad Policy Variable es precisamente garantizar la se-

guridad en el uso de las variables implícitas. Para resolver este problema, lo que la fábrica de contexto produce e inserta en el contexto no son instancias de los elementos de dominio sino objetos de tipo Policy Variable que encapsulan a los primeros para ofrecer una interfaz de acceso de sólo lectura al programador de scripts de adaptación. En este sentido, hay que señalar que Policy Variable es tan sólo una interfaz que define un tipo común para una colección de tipos específicos asociados a cada uno de los objetos de dominio. Es decir, para cada tipo de objeto de dominio manejado por los subsistemas de la plataforma existe un tipo de entidad Policy Variable que implementa la lógica de acceso protegido al mismo.

- **Policy Task.** Dado que, por motivos de seguridad, las transformaciones sobre los subsistemas de la plataforma no se pueden hacer por modificación directa de las variables implícitas de contexto, es necesario proporcionar a los diseñadores instruccionales un mecanismo para llevarlas a cabo. La responsabilidad de esta entidad es precisamente dar soporte a estas transformaciones. Policy Task es, en realidad, la entidad padre de una extensa jerarquía de tareas implementadas que tienen por objeto realizar una operación de transformación elemental sobre los subsistemas de la plataforma (ver tabla 3.3). Por ejemplo, existe una tarea para crear un grupo a partir de una plantilla, otra para ingresar a un usuario dentro de un grupo, otra para desplegar un proyecto sobre un espacio de trabajo, etc. En general puede afirmarse que para cualquier tarea de administración que puede llevarse a cabo manualmente a través de las interfaces de la plataforma existe una tarea homóloga que permite realizarla de forma automática. Estas tareas son accesibles desde los scripts a partir de 4 objetos que las incluyen como funciones: *common*, que engloba las de propósito más general; *social*, que incluye las relacionadas con el subsistema social; *collaboration* que agrupa las del subsistema de colaboración y *tools* que contiene las de integración (consulte tabla 3.3). El propósito de estas entidades es doble. Por un lado, proporcionan una implementación probada de operaciones de transformación no triviales lo que simplifica la labor de los diseñadores a la hora de definir scripts de adaptación y aumenta la orientación a dominio del lenguaje P#. Por otro lado, las tareas son implementadas de acuerdo a un modelo de transaccionalidad lo que garantiza la consistencia de la plataforma tras la aplicación de las transformaciones ya que éstas se llevan a cabo atómicamente y en aislamiento concurrente.

- **Policy Script.** Las entidades Policy Script representan los scripts de adaptación que son definidos por los diseñadores instruccionales y lanzados a ejecución posteriormente en diversos contextos socio–colaborativos. Desde el punto de vista tecnológico, un script está constituido por un nombre y una descripción que se utiliza para facilitar las labores de administración en el repositorio de scripts mantenido por la plataforma y que es accesible desde las interfaces de usuario. Además, esta entidad también contiene una referencia al texto del código fuente del scripts que será interpretado puntualmente a través de un proceso de contextualización en un contexto determinado. Aunque estos son los tres únicos elementos constituyentes del artefacto, conviene recordar que el texto de un script hace

referencia, potencialmente, a una colección de variables de contexto (implícitas o definidas por el usuario) así como una colección de tareas para su invocación con unos parámetros actuales específicos. El uso de estos dos tipos de referencias suele venir confinado a otro tipo de construcciones sintácticas tales como definiciones de clases, clausuras y funciones o sentencias de control de flujo de programa iterativas o condicionales tal y como se describió en el primer apartado de la sección introductoria 3.5.

- **Policy Engine.** Este artefacto representa el elemento central del entorno de ejecución sobre el que se ejecutan los scripts de adaptación. Los métodos de esta entidad permiten proporcionar un script (instancia de Policy Script) y un contexto de usuario (instancia de Policy Context) para llevar a cabo varias operaciones relacionadas con la interpretación, depuración y ejecución del script. La interpretación y depuración es utilizada en Pelican para proporcionar consolas en las interfaces de usuario que asistan a los diseñadores instruccionales a encontrar los errores existentes en el script. Por su parte, la ejecución contextualiza e interpreta secuencialmente el mismo de manera que al llegar a cada tarea referenciada, ésta se ejecuta invocándose al método *execute* de la misma. Al final del proceso es posible obtener un informe de la ejecución que indique el número de tareas ejecutadas y el tiempo total transcurrido.

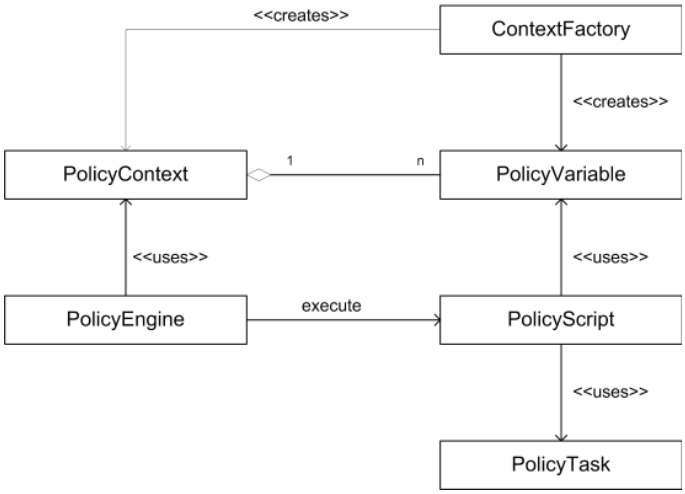

Figura 3.23. Modelo en UML de la ejecución de scripts de adaptación.

En la figura 3.23 se presenta un diagrama en UML que muestra todos los artefactos software que acabamos de describir con anterioridad. Como dijimos, éstos son todos aquéllos que resultan necesarios para dar soporte a la ejecución de scripts de adaptación. Como puede verse, las fabricas de contextos se utilizan para crear las variables implícitas que luego serán insertadas dentro de los contextos de usuario. El proceso de ejecución de scripts por su parte, está dirigido por el motor de ejecución que utiliza un contexto para contextualizar el mismo y que resuelve las referencias a variables en objetos de datos especificos y lanza a ejecución las tareas referenciadas desde el script.

Una vez estudiados los elementos que permiten dar soporte a la ejecución de scripts de adaptación estamos en disposición de extender esta familia de artefactos para considerar aquéllos necesarios en la ejecución temporal automática de los mismos. A continuación pasamos a describir en detalle todos ellos y a relacionarlos con los anteriores:

- **Policy Event Type.** Como explicamos en la introducción de esta sección los mecanismos de planificación de la ejecución automática de scripts están dirigidos por eventos. Es decir, para la especificación de un plan de ejecución, los diseñadores instruccionales deben indicar qué script desean lanzar a ejecución como respuesta a cierto tipo de acontecimientos pedagógicamente relevantes surgido durante el desarrollo de la experiencia. La responsabilidad de esta entidad es precisamente caracterizar la colección de todos los tipos de acontecimientos – en lo venidero tipo de evento – identificados por la plataforma. Para ello, este artefacto contiene, básicamente, un campo que identifica unívocamente el tipo de evento de que se trata. Este campo sigue, por convenio, una estructura sintáctica de identificadores separados por puntos lo que permite establecer espacios de nombre para garantizar la ausencia de colisiones (ver tabla 3.4). Como descubriremos en la sección 3.6, la colección de tipos de eventos reconocibles por la plataforma no está cerrada ya que, a los tipos de eventos propios de Pelican, se debe sumar la de todos los tipos de eventos que potencialmente disparan las herramientas integradas en el entorno. Por eso el uso de espacios de nombre es necesario. Adicionalmente, esta entidad contiene un nombre y una descripción que sirve para facilitar las tareas de administración en las interfaces de usuario. Finalmente, este artefacto dispone de un campo que permite indicar si los eventos de cierto tipo deben ser atendidos o no por la plataforma. Esto resulta de especial interés para mejorar la eficiencia de este subsistema. Hay que señalar que la especificación de los tipos de eventos que reconoce la plataforma pueden ser indicados editando los ficheros de configuración de Pelican. (Consulte el apéndice B para obtener detalles sobre la configuración).

- **Policy Event.** Cuando surge, dentro de la plataforma o en una herramienta externa integrada en el entorno, un acontecimiento de aquéllos representados por algún tipo de evento, el subsistema de intervención construye una instancia de la entidad Policy Event para representarlo. Un evento incluye una referencia a la entidad Policy Event Type que identifica el tipo de evento de que se trata, una marca de tiempo para indicar el momento exacto en que se produjo y el nombre de la fuente que originó el evento. Para el caso de la plataforma el valor de ese campo es Pelican mientras que en otro caso es un nombre que identifica la herramienta. Además el evento también incluye el usuario de la plataforma que originó el acontecimiento asociado al mismo así como un contexto formado por una colección de variables implícitas de evento que lo caracterizan. Estas variables pueden ser utilizadas dentro de los scripts de adaptación y de los disparadores de las reglas similarmente a como ocurre con las variables del contexto de usuario, aunque en el caso de la definición de reglas es más frecuente expresar estos elementos en

términos de la información proveniente del evento que se está atendiendo. En la tabla 3.4 puede encontrarse una descripción detallada de la colección de todos los tipos de eventos de Pelican y en el apéndice A se describen aquéllos generados por las herramientas que han sido utilizadas en nuestras pruebas descritas en el capítulo 4.

- **Policy Rule.** Para permitir a los diseñadores instruccionales indicar el script de adaptación que debe ser aplicado para tratar cada tipo de evento en una situación particular se utiliza un mecanismo de especificación basado en reglas. Como ya presentamos en la introducción, estas reglas relacionan tres tipos de entidades: un tipo de evento, un script de adaptación y un disparador que indica las condiciones ambientales que deben darse para que se produzca la ejecución del script ante la llegada de un evento del tipo asociado a la regla. La entidad policy rule es la encargada de representar este concepto dentro del subsistema de intervención. De acuerdo a lo que acabamos de decir, ésta incluye una referencia a un objeto de tipo Policy Event Type, otro a un objeto Policy Script para representar el script de adaptación asociado a la misma y una referencia a un objeto policy Trigger, que como veremos, se encarga de representar las condiciones de disparo. Adicionalmente, esta entidad también incluye un campo para indicar la prioridad de la ejecución de la misma cuando ésta forma parte de un plan de ejecución constituido por otras reglas. Cuando los scripts asociados a las reglas de un plan de ejecución son no concomitantes entre sí, esto es, cuando codifican trasformaciones que se aplican sobre distintas partes de los modelos internos de la plataforma sin interferencia entre ellas, el orden en el que se ejecuten las reglas es indiferente. No obstante, en otras situaciones, existe, por diseño, una dependencia temporal en la ejecución de los scripts que requiere cierto orden en la ejecución de los mismos. La existencia de estas dependencias suele deberse al hecho de que las trasformaciones del script dependiente se apoyan o requieren de un estado en la plataforma que sólo es alcanzado cuando se ejecuta el script del que éste depende y se gestionan a través de la información de prioridad de las reglas.

- **Policy Trigger.** El disparador de una regla es una expresión lógica de P# que indica las condiciones ambientales que deben satisfacerse para que el script asociado a la misma se lance a ejecución. Los disparadores suelen venir expresados precisamente en términos de las variables implícitas del evento que se está tratando aunque también pueden referenciar cualquier variable del contexto (variables implícitas de dominio o variables definidas por el usuario). De esta manera, ante la llegada de un evento, solamente serán ejecutados los scripts de aquellas reglas cuyos disparadores sean evaluados a cierto. El concepto de disparador juega un papel fundamental dentro de la especificación de la planificación de la ejecución. En efecto, estas entidades sirven en la práctica para indicar unívocamente el escenario colaborativo al que hace referencia la regla. Por ejemplo supongamos que estamos implantando en Pelican un escenario de Jigsaw y deseamos ejecutar un script para la creación automática de grupos expertos tras la realización de la primera actividad. El tipo evento asociado a la regla debería ser aquel que indique

un cambio de estado de actividad. Sin embargo el disparador debe precisar que la regla solamente debe ejecutarse si el evento de cambio de estado proviene de alguno de los grupos que están desarrollando la experiencia ya que potencialmente podría haber muchos eventos concurrentes que respondan a acontecimientos acaecidos en otros contextos sociales. Es decir, los disparadores son expresiones de filtro para caracterizar el momento en que una regla debe aplicarse.

- **Policy.** Comúnmente, para capturar la colección de aspectos dinámicos que aparecen relacionados con el desarrollo de un determinado escenario de aprendizaje colaborativo es preciso utilizar una colección de reglas. En efecto, los escenarios relativamente complejos que requieren aplicar diferentes transformaciones a lo largo del tiempo, se expresan a partir de una colección de reglas que se encuentran conceptualmente vinculadas entre sí, en tanto que todas ellas capturan en su conjunto un modelo de intervención para el escenario. En este sentido, se utiliza el concepto de política como unidad organizativa que permite agrupar todas las especificaciones que forman parte de cada modelo de intervención. El nombre proviene del hecho de que los aspectos dinámicos suelen relacionarse con estrategias de gestión de diferentes tipos de procesos propios de una experiencia. En este sentido se puede hablar, para cada escenario, de la política de gestión social, la de monitorización y control, la de evaluación, etc. Desde el punto de vista tecnológico una política es una entidad que agrupa, básicamente, una colección de reglas y proporciona ciertos mecanismos para parametrizar su expresión. Esto convierte a las políticas en candidatos idóneos para la reutilización ya que, como acabamos de apuntar las reglas individuales de la misma no suelen tener ningún valor semántico aislado.

- **Policy Event Dispatcher.** Los eventos que son lanzados por la plataforma como consecuencia de alguna acción llevada a cabo sobre las interfaces de la misma o las herramientas externas integradas, son recogidos por este artefacto para su tratamiento. El despachador de eventos se encarga de llevar a cabo la secuencia de 4 fases que se describió con anterioridad (escucha, selección, planificación y ejecución). Las solicitudes de tratamiento de eventos son encoladas para garantizar un procesamiento de los mismos de forma aislada, secuencial y por orden de llegada y asegurar así la integridad de la plataforma en todo momento.

La figura 3.24, ilustra un diagrama en UML que representa el modelo de clases que es utilizado para dar soporte a la planificación temporal de la ejecución de scripts de adaptación de forma automática. Como se puede ver, la entidad central es la regla. Una regla está contenida, con exclusividad, dentro de una política y cada política agrupa una colección de reglas relacionadas entre sí. Desde el punto de vista estructural, la regla contiene un tipo de evento y cada tipo de evento puede ser asociado a una colección de reglas. Esto permite clasificar a las mismas en función del tipo de evento al que atienden. Por otro lado, cada regla dispone de un disparador único asociado con exclusividad a la misma. Y finalmente, cada regla dispone de un script que puede ser compartido por varias reglas, para permitir su reutilización en diferentes situaciones y escenarios.

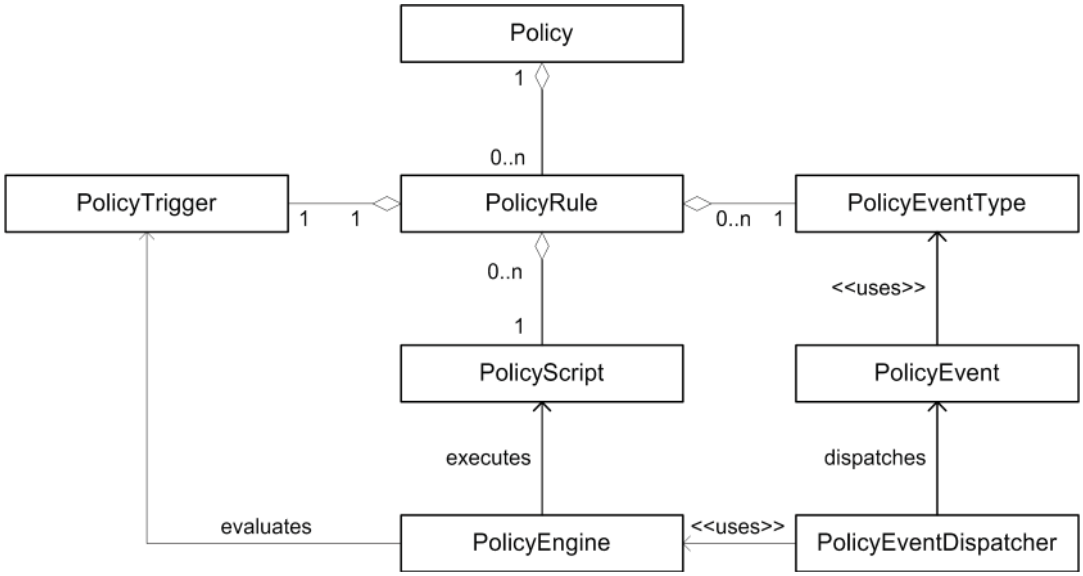

Figura 3.24. Modelo en UML de ejecución automática de scripts basada en reglas.

Con respecto a la ejecución automática de scripts, el proceso comienza, como dijimos por la construcción de un evento para representar el acontecimiento. Éste utiliza la entidad Policy Event Type para indicar el tipo de evento de que se trata y la entidad Policy Event Dispatcher se encarga de procesarlo. Para ello hace uso del motor de ejecución que se encargará de evaluar el disparador de la regla y de lanzar a ejecución script de adaptación si procede.

3.5.2. La interfaz del subsistema de intervención

Las interfaces de usuario proporcionadas por el subsistema de intervención dan soporte a la especificación de modelos de intervención completos. En concreto, la interfaz para la administración de scripts de adaptación permite gestionar la definición de scripts y acceder al explorador de contextos, un interfaz que permite navegar por los contextos de usuarios con sesión abierta en la plataforma. Adicionalmente, para dar soporte a la especificación de scripts, la plataforma proporciona dos consolas: la consola de depuración y la consola de ejecución. La primera tiene por objeto asistir en las tareas de codificación indicando los errores encontrados cuando los hay. La segunda por su parte, permite lanzar a ejecución los script indicando el contexto de usuario que debe usarse para la contextualización. Finalmente la interfaz para la administración de políticas y reglas políticas permite definir la planificación temporal de la ejecución de scripts y la consola de políticas monitorizar este proceso.

Interfaz para la administración de scripts de adaptación

Desde esta interfaz de usuario, los diseñadores instruccionales y administradores de la plataforma pueden definir scripts de adaptación. Como puede observarse en la figura 3.25, a la izquierda aparece una relación de todos los scripts definidos actualmente (❶).

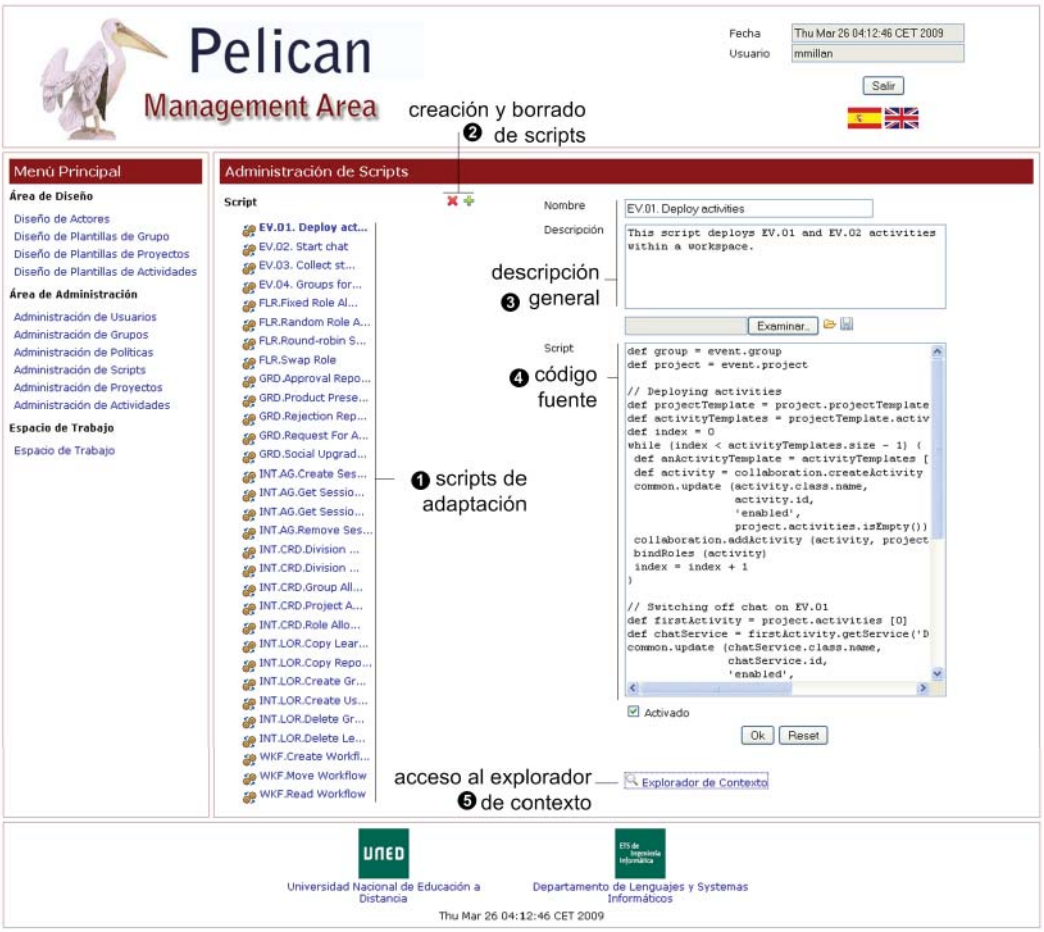

Figura 3.25. Interfaz para la administración de scripts de adaptación.

Los diseñadores instruccionales y administradores de la plataforma puede seleccionar un elemento de esta lista de scripts, lo que provocará que éste aparezca entonces resaltado en negrilla. Al hacer esto, el script de adaptación seleccionado pasa a ser editado sobre el formulario que, a tal efecto, aparece a la derecha. Si se desea eliminar un script debe seleccionarse el mismo y pulsarse sobre el botón de eliminación (❷). Adjunto a éste se encuentra el botón de creación que permite crear un nuevo script de adaptación con un nombre temporal. Este nuevo script pasará a ser editado sobre el formulario para que pueda definirse.

El formulario de edición está formado por una parte que permite indicar un nombre y una descripción para el script (❸). El nombre se utiliza para identificar de manera unívoca el script de adaptación en la plataforma por lo que no debe coincidir con el de ningún otro script previamente definido. Por su parte, la descripción permite almacenar un texto que refleja los propósitos del mismo. Debajo se encuentra un área de texto (❹) donde debe escribirse el código fuente del script de adaptación. El botón inferior de Ok permite salvar el mismo. Finalmente, desde esta interfaz se ofrece un enlace para acceder al explorador de contextos que describiremos en el siguiente apartado (❺).

Interfaz del explorador de contextos de usuario

El explorador de contexto permite seleccionar un contexto de usuario, navegar por el grafo de objetos contenidos dentro de él y realizar consultas sobre el valor de los mismos. Esta interfaz de la plataforma resulta de mucha utilidad para los administradores y diseñadores instruccionales ya que permite conocer la colección de variables que pueden ser referenciadas desde los scripts que se están definiendo.

Como puede verse en la figura 3.26, lo primero que debe hacerse es seleccionar el contexto que deseamos explorar (❶). Esto actualizará automáticamente la interfaz para pasar a representar el mismo. El administrador puede escribir literalmente una expresión sobre las variables del contexto para consultar su valor actual (❷). Este resultado se muestra en el área de texto de debajo de la consulta (❸). En la figura por ejemplo se está pidiendo devolver la colección de permisos asociados al actor en curso. Otra manera de utilizar el explorador es mediante el navegador de contexto (❹). En un principio éste muestra la colección de todas las variables contenidas en el contexto. Cuando se pincha sobre una de ellas, se pasa a mostrar la colección de atributos accesibles desde la variable seleccionada y se actualiza el valor de la expresión de

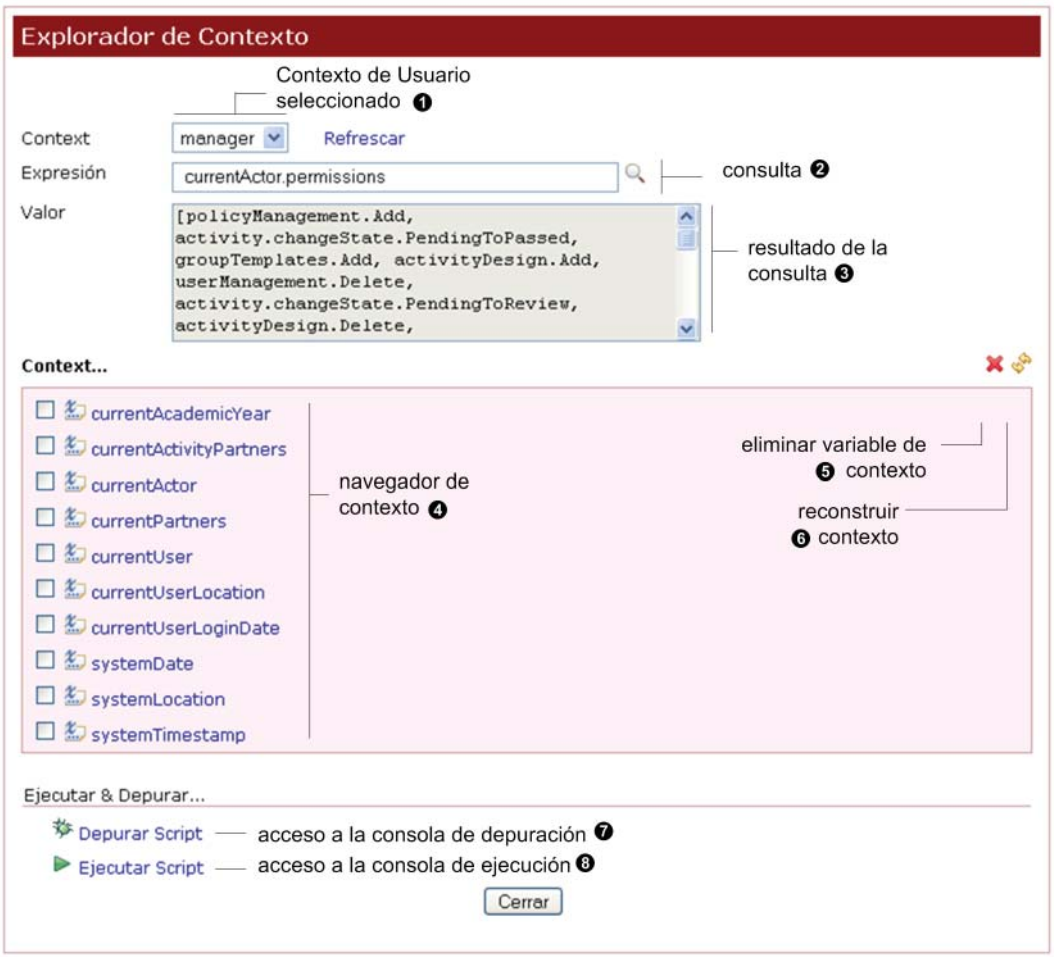

Figura 3.26. Interfaz del explorador de contextos de usuario.

consulta y su resultado para mostrar en todo momento el valor actual de cada elemento del grafo de objetos. Desde la interfaz del explorador también es posible eliminar variables. Para ello deben seleccionarse las variables deseadas sobre el navegador y pulsar en el botón de borrado (❺). Asimismo, también se puede restaurar el contexto completamente de manera que se eliminen todas las variables definidas por el usuario y sólo persista una versión actualizada de las variables implícitas de dominio. Para ello puede usarse el botón de reconstrucción de contexto (❻).

Desde el explorador de contexto también es posible acceder a la consola para la depuración de scripts (❼) y a la consola para la ejecución de los mismos (❽), que pasamos a describir en los dos apartados subsiguientes.

Interfaz de la consola de depuración de scripts de adaptación

La consola de depuración es una herramienta que permite identificar los errores que se encuentran dentro del código fuente del script sobre la interfaz de administración de los mismos. Como puede verse en la figura 3.27, esta consola proporciona información sobre el script en depuración (❶) y el contexto de usuario seleccionado en el explorador (❷). Debajo, se representa el código fuente del script (❸) y, en caso de detectarse errores, éstos se resaltan sobre el mismo (❹). Además la tabla inferior muestra un resumen de dichos errores ofreciendo, para cada uno de ellos, una descripción del problema encontrado y el número de línea y columna donde aparece.

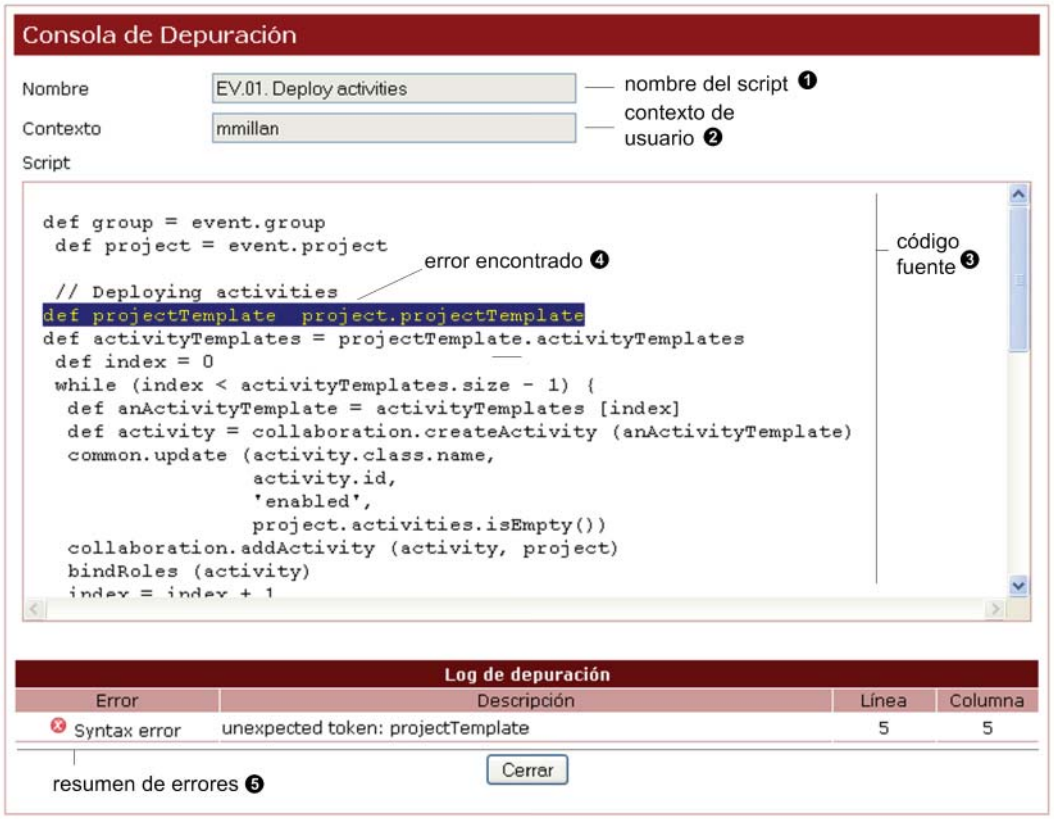

Figura 3.27. Interfaz de la consola de depuración de scripts de adaptación.

Interfaz de la consola para la ejecución de scripts de adaptación

Como dijimos con anterioridad, la ejecución de scripts de adaptación es responsabilidad, típicamente, de los monitores. Para lanzar a ejecución un determinado script manualmente, éstos deben acceder a la interfaz de administración de scripts y seleccionar allí aquél que desean aplicar. Después, deben acceder al explorador de contexto para seleccionar el contexto de usuario que será utilizado en el proceso de contextualización. Y finalmente, deben acceder a la consola de ejecución cuya interfaz se muestra en la figura 3.28. Como puede apreciarse en dicha figura, lo primero que aparece dentro de la interfaz es información relativa al script de adaptación que se va a ejecutar (❶) y el contexto de usuario que ha sido seleccionado para contextualizar el mismo (❷). Para lanzar a ejecución el script seleccionado lo único que es necesario es presionar el botón de ejecución situado a la derecha en la interfaz (❸).

Una vez realizada la ejecución, la consola (❹) muestra una traza detallada de todas las tareas del script que han sido aplicadas indicando para cada una de ellas el nombre de la tarea, el modo de ejecución, los parámetros y el estado de terminación del mismo (éxito o fracaso). El modo de ejecución puede ser real o en modo de prueba, que se puede seleccionar con el selector situado encima del botón de ejecución. Cuando un script es ejecutado en modo de prueba, éste es ejecutado pero las tareas no son lanzadas a ejecución. Esto es una facilidad para asistir en la depuración del script.

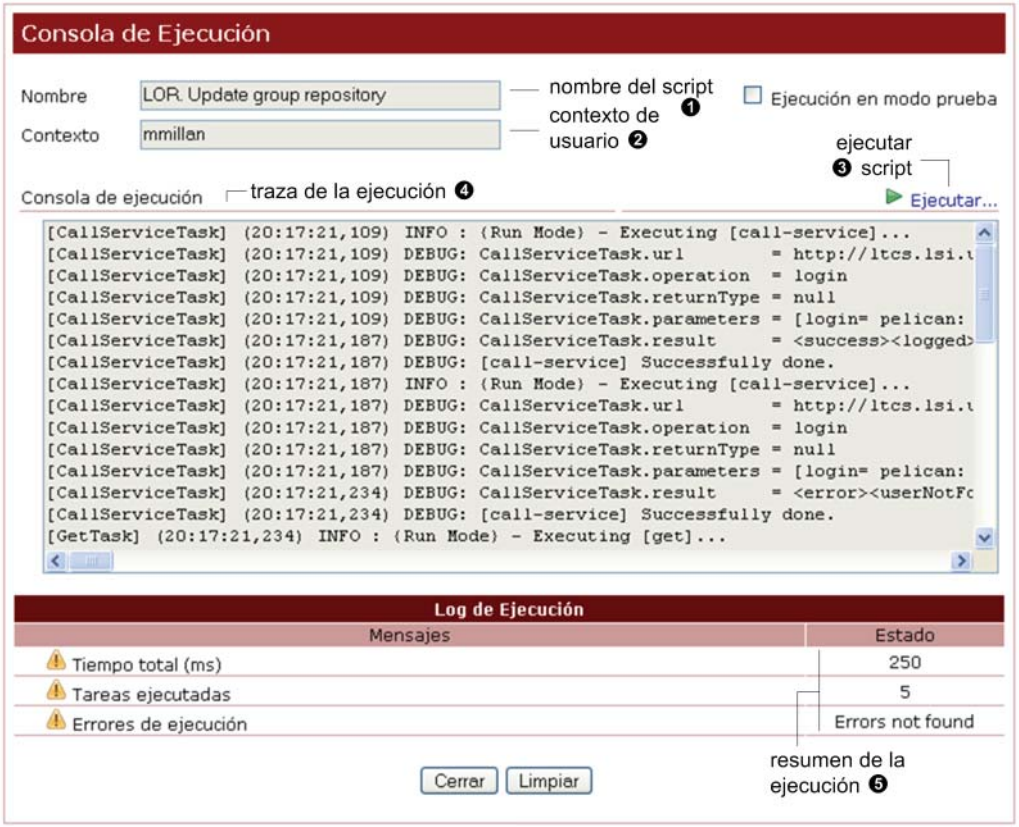

Figura 3.28. Interfaz de la consola para la ejecución de scripts de adaptación.

Interfaz para la administración de políticas y reglas políticas

La interfaz para la administración de políticas y reglas políticas permite establecer una planificación temporal para la ejecución automática de scripts de adaptación. A través de ella, los diseñadores crean una política que típicamente incluye la relación de reglas necesarias para capturar los aspectos dinámicos de un escenario y crear así un modelo de intervención para el mismo.

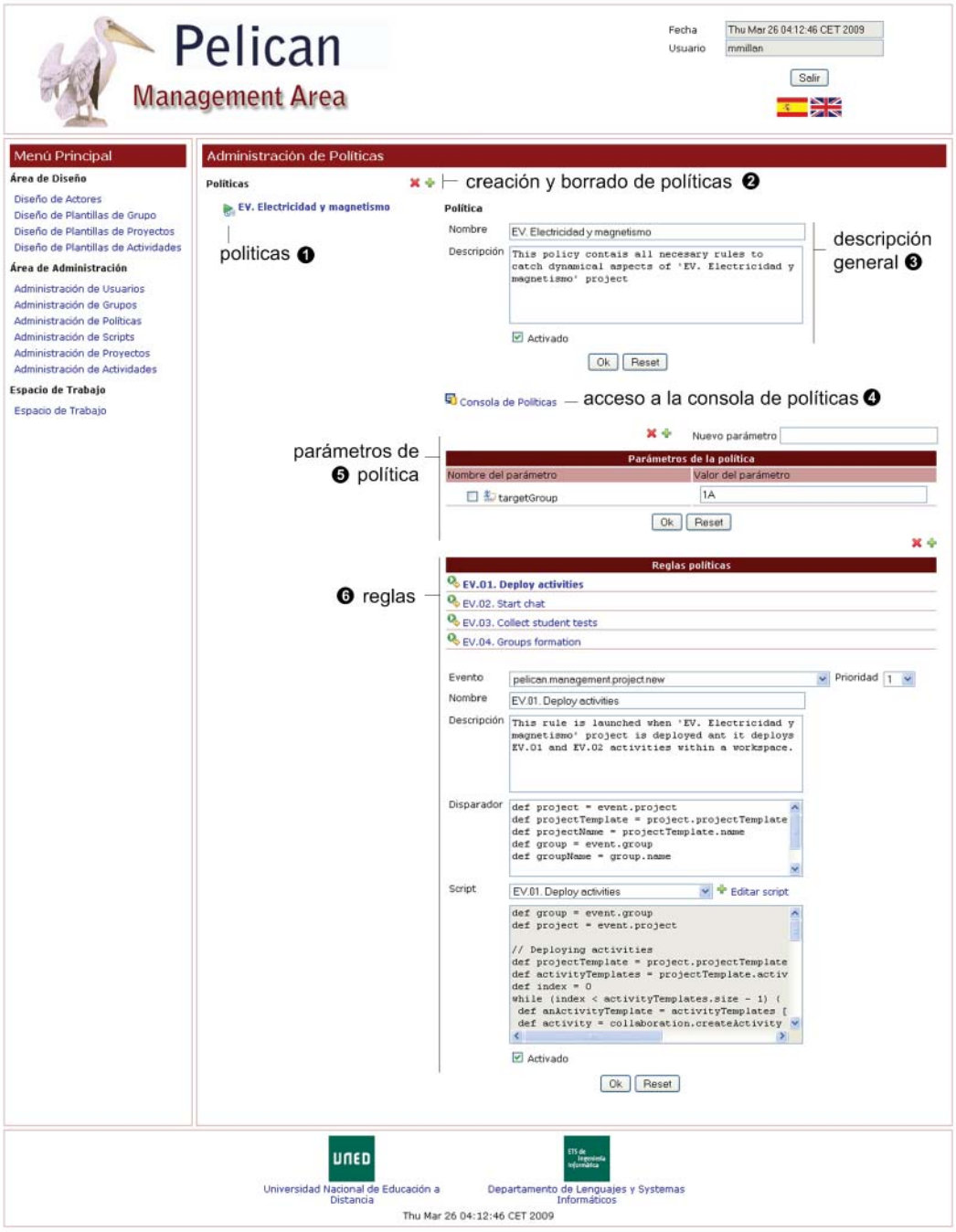

Figura 3.29. Interfaz para la administración de políticas y reglas políticas.

La figura 3.29 muestra esta interfaz de administración. Como puede apreciarse, en la parte de la izquierda aparece una relación de todas las políticas actualmente definidas en la plataforma (❶). El administrador puede seleccionar un elemento de esta lista para editarlo en la parte derecha del interfaz. Si se desea eliminar la política actualmente seleccionada debe pulsarse sobre el botón de borrado, mientras que si lo que se desea es crear una nueva política se pulsa sobre el botón de creación y como consecuencia ésta aparecerá, con un nombre temporal, sobre la lista y se seleccionará automáticamente para su edición (❷).

En el interfaz de la derecha, el área de descripción general (❸) permite establecer un nombre único que identifica la política dentro del sistema y una descripción para detallar el propósito general de todas las reglas contenidas dentro de la misma. Debajo de esta parte aparece una tabla que permite definir los parámetros de la política (❺). Estos parámetros son una colección de variables implícitas compartidas por todas las reglas que aparecen en la política y que se insertan en el contexto de usuario justo antes de la ejecución de cada regla. Como puede apreciarse existen botones para la creación y borrado de parámetros. En concreto la creación de un parámetro requiere proporcionar un nombre para el mismo e indicar el valor que se le asociará. El nombre debe seguir las reglas de identificadores de P# ya que al fin y al cabo se trata de variables referenciadas dentro de los scripts y los disparadores. Por su parte el valor asociado a los mismos puede ser cualquier expresión de este lenguaje. (Los detalles sintácticos de P# fueron descritos en con anterioridad en la sección 3.5).

Tras la sección de parámetros se encuentra la de declaración de reglas (❻). En la parte derecha aparecen dos botones que permiten llevar a cabo las tareas de creación y eliminación de reglas asociadas a la política seleccionada y debajo se muestra una tabla que contiene la colección de todas las reglas actualmente definidas. Para editar una regla debe seleccionarse una entrada de esta tabla y, como consecuencia, ésta pasa a mostrarse en el formulario de edición. Este formulario permite asignar un nombre y una descripción a la regla así como seleccionar el tipo de evento a la que responde, definir un disparador para el mismo, asociarle un script de adaptación y especificar el nivel de prioridad (1 para el máximo, 10 para el mínimo).

Esta interfaz incluye también dos selectores que permiten activar o desactivar la política (debajo de ❸) y la regla (debajo de ❻). Cuando una política se desactiva todas las reglas definidas dentro de ella son ignoradas durante el procesamiento de las reglas. De manera similar, cuando una regla es desactivada ésta no es tenida en cuenta en dicho proceso. Por último, desde aquí se ofrece acceso a la consola de políticas que describimos a continuación (❹).

Interfaz para la consola de políticas

La interfaz mostrada en la figura 3.30 está diseñada para que los administradores de la plataforma puedan seguir las actividades de transformación adaptativa que se están produciendo en la misma. Como puede apreciarse en la figura, a parte de ofrecer información sobre el contexto de usuario (❶) y la fecha actual del sistema (❷), esta consola está formada por 3 áreas de texto donde se presentan, respectivamente, la

traza de todos los eventos que son disparados a lo largo del tiempo en la plataforma (❸), las reglas políticas que son revisadas para tratar dichos eventos (❹) y la colección de tareas que como consecuencia son lanzadas a ejecución (❺).

Figura 3.30. Interfaz para la consola de políticas.

3.6. Soporte a integración de herramientas externas

En las tres secciones anteriores hemos presentado sendos subsistemas que ofrecen facilidades para articular el diseño y despliegue de las experiencias de aprendizaje colaborativo. En concreto, el subsistema de colaboración permitía, a este respecto, organizar la definición de las mismas en términos de proyectos y actividades. Sin embargo, para hacer posible el desarrollo de este tipo de experiencias deben proporcionarse herramientas de soporte a la interacción colaborativa. En efecto, no debemos olvidar que, en definitiva, es en el seno de dicha interacción donde todo el aprendizaje en colaboración tiene lugar [Weinberger, 2003] [Dillenbourg & Scheneider, 1995] [Doyse & Mugny, 1984].

La mayoría de los entornos virtuales de aprendizaje [Blackboard] [Moodle] [Alf] proporcionan, en este sentido, una colección canónica, cerrada y por lo general insuficiente de servicios de interacción empotrados dentro de los mismos tales como sistemas de mensajería instantánea, repositorios de recursos de aprendizaje, herramientas de votación y soporte a la decisión o foros de discusión. Sin embargo, en la práctica se ha comprobado que el tipo de interacciones a las que es necesario dar soporte depende de la naturaleza de la experiencia, del dominio de conocimiento de la misma y de los objetivos pedagógicos perseguidos [Dillenbourg, 1999]. Todo ello demanda un cambio en la manera en que deben ser concebidos estos entornos. En lugar de proporcionar una colección canónica y cerrada de servicios, los entornos deben permitir la incorporación sencilla y eficaz de nuevas herramientas externas de soporte a la interacción potencialmente desarrolladas por terceras partes.

El subsistema de integración de Pelican, que presentamos en esta sección, apunta precisamente en esta dirección. En efecto, desde cierto punto de vista, Pelican puede ser considerada como una plataforma de integración donde una colección de servicios proporcionados por herramientas externas utilizan los mecanismos de este subsistema para incorporarse a los espacios de trabajo de la plataforma. En este sentido, pueden destacarse dos principios fundamentales de integración que se han establecido como objetivos de diseño de nuestra propuesta:

- **Proporcionar mecanismos de Integración no intrusivos.** Los mecanismos de integración provistos por la plataforma deben ser no intrusivos con respecto a la arquitectura interna de las herramientas de interacción incorporadas dentro del entorno. Esto quiere decir que, en la medida de lo posible, las herramientas no deberían tener que cambiar para adaptarse a Pelican ni tampoco deberían desarrollarse adaptadores software para tal fin. Tal y como fue discutido en el estado del arte, esta es una diferencia importante con respecto a la mayoría de las propuestas de integración existentes en la actualidad que se apoyan en estándares de facto a los que deben adscribirse los desarrolladores si quieren integrar sus herramientas al entorno en cuestión [IMS–TI] [OSGi] [OGSA]. Para dar soporte a esta idea Pelican proporciona cuatro mecanismos de integración no mutuamente excluyentes que ofrecen diferentes capacidades e imponen niveles de compromiso tecnológico creciente.

- **Adaptar las herramientas al flujo de trabajo instruccional.** La integración de las herramientas no debe limitarse a incorporar éstas dentro del entorno de la plataforma de manera que sean agnósticas las unas de las otras. Adicionalmente, deben proporcionarse los mecanismos pertinentes para ofrecer un grado de interoperabilidad y cohesión suficiente entre las mismas. Es decir, los servicios proporcionados por las herramientas de un escenario deben ser conectados funcionalmente entre sí para que ofrezcan una sensación de continuidad en el uso de las mismas a lo largo del desarrollo del flujo de trabajo instruccional. Esto implica que el comportamiento mostrado por las herramientas en cada momento dependerá, potencialmente, tanto del estado de la plataforma como de aquél en que se encuentren el resto de las herramientas. Por ejemplo, un determinado escenario de aprendizaje podría requerir que, tras una actividad de discusión mantenida a través de un foro de debate, se transite a otra actividad de resolución donde se utiliza una herramienta de votación para determinar la propuesta ganadora de tal discusión. La integración en este escenario debería permitir que cada hilo de discusión creado en el foro apareciese automáticamente como una opción candidata de la sesión de votación subsiguiente. En este sentido, las capacidades del subsistema de intervención descrito en la sección 3.5, son una solución al respecto.

De la descripción anterior es posible establecer una clara diferenciación entre dos tipos de integración complementarios y ortogonales entre sí. A continuación describimos en detalle cada una ellas:

- **Integración Vertical.** La integración vertical permite mantener cohesionados los modelos de información de Pelican con cada uno de los modelos propios de las herramientas externas y recíprocamente. En este sentido podemos distinguir dos tipos de integración vertical:

  – *Integración vertical descendente.* La integración vertical descendente es la que se produce desde las herramientas hacia los modelos de la plataforma. De esta manera, los cambios ocasionados en las herramientas, como consecuencia de la interacción de los usuarios sobre las mismas, se propagarán descendentemente hacia la plataforma para que sean atendidos por ésta. En el escenario de Jigsaw por ejemplo, la primera actividad requiere que cada uno de los usuarios escoja el grupo al que quiere pertenecer. Esta decisión es soportada, típicamente, a través de una herramienta de votación. El procesamiento del resultado de dicha votación debe desencadenar una adaptación del modelo social de manera que cada usuario aparezca enrolado en el grupo que escogió.

  – *Integración vertical ascendente.* Los cambios ocasionados en el subsistema social y colaborativo de la plataforma deben también propagarse automáticamente hacia las herramientas para que éstas adapten su comportamiento adecuadamente. En el ejemplo de Jigsaw, es necesario asociar a cada nuevo grupo experto un servicio de chat. Este es un ejemplo de integración vertical ascendente, donde la creación de grupos desencadena la configuración de nuevos servicios.

- **Integración horizontal.** En la integración horizontal las herramientas incorporadas en el entorno utilizan la infraestructura de Pelican para establecer una comunicación bidireccional entre las mismas. Esto significa que las interacciones realizadas sobre una herramienta podrían transmitirse horizontalmente a otra u otras herramientas del entorno para que éstas adaptasen su comportamiento en relación a los mismos. Los mecanismos de integración horizontal llevan a cabo la propagación de dichos cambios de forma automática y transparente a los usuarios. El escenario que fue descrito anteriormente al presentar el segundo objetivo de este subsistema – la adaptación de las herramientas al flujo de trabajo instruccional – es un claro ejemplo en este sentido. En efecto, la colección de hilos que se han creado dentro de un foro se transmiten sobre la infraestructura de Pelican para dar lugar a una colección de opciones dentro de una nueva sesión en la herramienta de votación.

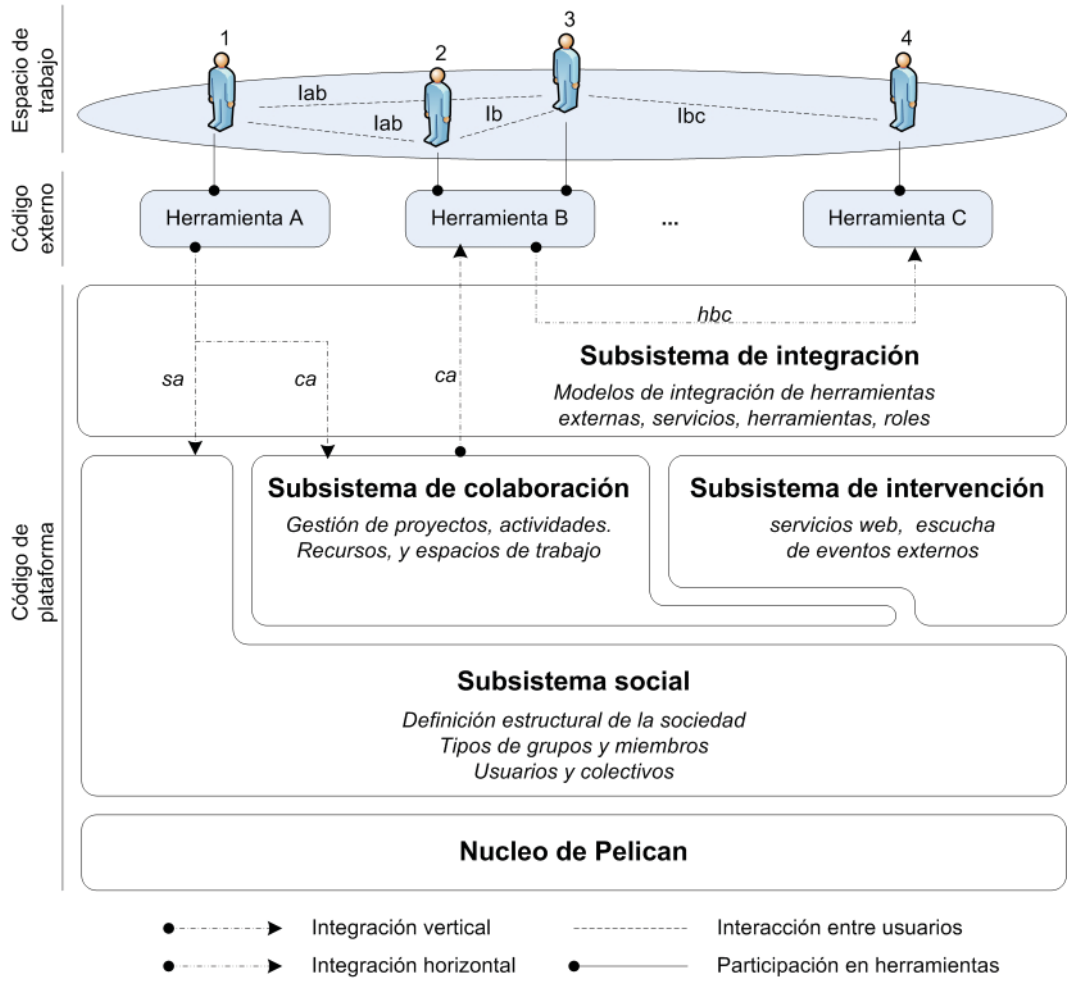

Figura 3.31. Ejemplo de integración de herramientas en Pelican.

Para ilustrar el uso de estos dos tipos de integración consideremos el escenario de ejemplo presentado en la figura 3.31. En ella puede verse cómo cuatro estudiantes comparten un espacio de trabajo participando en el uso de tres herramientas de so-

porte (*A, B* y *C*). Como puede apreciarse, el uso de la herramienta A en el ejemplo genera sendas alteraciones sobre el subsistema social (*sa*) y colaborativo (*ca*) de la plataforma que son propagados a través del subsistema de integración. Por su parte, la herramienta B, atiende a los cambios producidos sobre el subsistema de colaboración (*ca*) y se adapta convenientemente a ellos. Asimismo, el uso de dicha herramienta provoca ciertas alteraciones sobre su modelo interno que se transmiten horizontalmente (*hbc*) hacia la herramienta C para que ésta adapte su comportamiento ajustándose a tales cambios. En este ejemplo aparecen representadas diferentes interacciones entre los estudiantes tal y como se ilustra en el diagrama de la figura con trazos discontinuos . A continuación discutiremos cada una de ellas:

- **Interacción Iab.** En la interacción *Iab*, la participación del estudiante 1 en la herramienta A produce unas alteraciones, *ca*, que se propagan verticalmente hacia el subsistema de colaboración de la plataforma y desde ahí hacia la herramienta B, en cuyo uso participan los estudiantes 2 y 3. Este mecanismo permite coordinar, de manera indirecta, la actuación de los 3 estudiantes. Este es un ejemplo de cómo la integración vertical puede soportar la interacción colaborativa entre estudiantes que participen en diferentes herramientas.

- **Interacción Ib.** En la interacción *Ib*, los estudiantes 2 y 3 participan en el uso de la herramienta B. En este caso, la colaboración queda completamente circunscrita al uso de dicha herramienta y por tanto, no hay ninguna implicación por parte de la arquitectura de Pelican, aparte de la de permitir que la herramienta se incorpore y sea utilizada adecuadamente dentro del espacio de trabajo compartido para que todos los estudiantes tengan acceso a ella.

- **Interacción Ibc.** En este último caso, los estudiantes 3 y 4 mantienen una interacción colaborativa a través de su participación respectiva en las herramientas B y C. En efecto, una de las acciones realizadas por el estudiante 3 sobre B se propaga horizontalmente sobre la infraestructura del modelo de colaboración hacia la herramienta C, y ésta modifica su comportamiento lo cual es percibido por el estudiante 4 que la está utilizando. Este es un ejemplo de cómo la integración horizontal puede dar soporte a la interacción colaborativa entre estudiantes que participen en distintas herramientas ya sea en modo síncrono o asíncrono.

La situación colaborativa expuesta en este escenario es un claro ejemplo de lo que ocurre frecuentemente cuando un grupo de estudiantes lleva a cabo el desarrollo de una actividad de aprendizaje mediante el uso transversal de una colección de herramientas de soporte a la interacción. Este escenario ilustra, de manera general, cómo las herramientas externas pueden sacar partido a la infraestructura proporcionada por la plataforma para integrarse en un flujo de instrucción específico e interoperar con otras herramientas de su entorno.

Desde el punto de vista tecnológico, el subsistema de integración puede ser descrito como un middleware de orquestación que proporciona diferentes mecanismos orientados tanto a adaptar las herramientas a la plataforma como a proveer un canal de comunicación para favorecer la interoperabilidad entre las mismas. En este sentido,

los principios de diseño de este subsistema se pueden resumir en los tres puntos que describimos a continuación:

- **Integración orientada a roles.** La integración vertical ascendente permite que las herramientas externas se mantengan a las escucha de los cambios ocurridos dentro de la plataforma – en particular de los subsistemas social y de colaboración – para adaptar su comportamiento puntualmente a los mismos. En concreto, como discutiremos en la siguiente subsección, el subsistema de colaboración es extendido aquí para permitir la especificación de **roles** vinculados al desarrollo de la actividad. Estos roles representan un conjunto de responsabilidades colaborativas dentro de la misma y confieren al usuario ciertos permisos dentro de las herramientas integradas en el entorno. Así, se dice que la integración en Pelican está orientada a roles porque se espera que las herramientas, con un grado de integración suficiente, adapten su comportamiento especialmente en función del rol que los usuarios estén desempeñando en cada momento, aunque otros aspectos socio–colaborativos tales como el grupo al que pertenecen, el actor que encarnan o el proyecto en el que están implicados también puedan condicionarlo.

- **Integración basada en servicios Web.** Las adaptaciones que es necesario llevar a cabo para soportar la integración horizontal y vertical de las herramientas se realiza a través de los servicios Web que éstas típicamente proporcionan. En el ejemplo que presentábamos al describir el objetivo de adaptación al flujo de trabajo instruccional de la plataforma, donde la herramienta de soporte a la decisión es adaptada para incluir una nueva sesión de votación con los hilos del foro de debate como candidatos, el subsistema de integración utiliza los servicios Web de la misma para crear nuevas votaciones.

- **Integración dirigida por reglas políticas.** De lo descrito hasta ahora se deduce que la integración de herramientas en el entorno de aprendizaje requiere implantar en la plataforma ciertos protocolos de integración que lleven a cabo las tareas de orquestación necesarias para obtener un nivel de interoperabilidad adecuado entre las mismas. Estos protocolos son diferentes para cada escenario colaborativo diseñado en Pelican por cuanto vinculan una colección de adaptaciones específicas sobre los modelos internos de las herramientas de acuerdo a los requerimientos del flujo de trabajo instruccional. Por este motivo, y por tratarse en esencia de transformaciones adaptativas, el subsistema de integración utiliza las facilidades del subsistema de intervención para hacer posible la especificación declarativa de tales protocolos. Volviendo al ejemplo anterior, la creación de una nueva sesión de votación en la herramienta de soporte a la decisión es especificada a través de una regla política cuyo script de adaptación se lanzará a ejecución justo antes de la realización de la actividad donde se lleve a cabo la votación.

Estas tres características esenciales del subsistema de integración ofrecen gran flexibilidad a la hora de hacer frente a los problemas de integración que aparecen en la mayor parte de los escenarios de aprendizaje colaborativo (consulte la sección 4.5 donde se incluye una descripción detallada de la implantación completa de varios

escenarios complejos). No obstante, no debemos olvidar, que uno de los objetivos principales de este subsistema es garantizar una integración poco invasiva con respecto a la arquitectura de las herramientas externas. Precisamente por este motivo, este subsistema proporciona cuatro mecanismos de integración complementarios que ofrecen diferentes niveles de interoperabilidad entre las herramientas y acoplamiento con los modelos de la plataforma. Cada herramienta puede elegir el nivel de integración al que desea adscribirse a la misma. En cualquier caso, la determinación de los mecanismos de integración utilizados dependerá, en última instancia, de los requerimientos del escenario de aprendizaje que se esté diseñando. A continuación describimos en detalle cada uno de ellos:

- **Nivel A Integración ligera.** El mecanismo de integración ligera es el que ofrece un menor nivel de interoperabilidad e implica un grado de acoplamiento más bajo con los modelos internos de la plataforma. De los cuatro mecanismos presentados aquí, éste resulta el más sencillo y todas las herramientas que se integran en Pelican deben utilizarlo. El único requerimiento en este sentido es que la herramienta sea accesible a través de una dirección Web. De esta manera, para incorporar una herramienta externa en la plataforma utilizando integración ligera, lo único que es necesario es definir un servicio. Cada **servicio** se define dentro de un espacio de trabajo o una plantilla de actividad e identifica un uso particular y contextualizado que se hace de una determinada herramienta. Para ello es preciso proporcionar una URI que haga referencia a la herramienta y un **esquema de invocación**. Este esquema indica cómo debe ser invocada la herramienta cuando ésta es accedida desde el espacio de trabajo donde se incorpora. En concreto, el esquema de invocación proporciona valores adecuados para cada uno de los parámetros requeridos en la invocación. En la práctica, estos valores dependen del contexto socio–colaborativo donde las herramientas son accedidas, y su expresión suele ser hecha en términos abstractos a nivel del modelo de colaboración donde se define el servicio para que, tras el despliegue en un espacio de trabajo específico, éstos se resuelvan a valores concretos. Para definir estos valores pueden utilizarse expresiones de P# que referencien potencialmente a variables almacenadas en el contexto de usuario. Supongamos, por ejemplo, que estamos definiendo un escenario de aprendizaje donde se utiliza una herramienta de chat, y que ésta requiere como parámetro un valor que identifique el salón de discusión al que se desea acceder. Si los requerimientos de la experiencia demandan que debe existir un salón de discusión diferente para cada grupo implicado en el desarrollo de la misma, entonces el valor asignado a ese parámetro podría corresponderse, por ejemplo, con el nombre del grupo, sea cual sea éste en cada caso. La expresión de P# *currentGroup.name* permite indicar a Pelican el valor de este parámetro en términos abstractos. En tiempo de despliegue, cuando los servicios son incorporados a los espacios de trabajo, esta expresión será resuelta a valores concretos utilizando para ello el contexto de usuario que invoca la herramienta desde su espacio de trabajo. Este mecanismo atiende al primero de los objetivos que presentamos al inicio de esta sección ya que permite incorporar herramientas externas sin más que conocer la manera de acceder a ella a través de la Web.

- **Nivel B Integración dirigida por la plataforma.** El mecanismo de integración dirigido por la plataforma, requiere que las herramientas externas desarrolladas por terceras partes ofrezcan capacidades adaptativas a través de una colección de servicios Web. De esta manera, las herramientas pueden recibir instrucciones de configuración explicitas que les permitan adaptar su comportamiento bajo demanda de las directrices de la plataforma. Este mecanismo de integración no impone ningún consenso en la naturaleza de los servicios Web proporcionados por las herramientas. En efecto, el proceso de integración consiste, en este caso, en llevar a cabo un protocolo que es expresado dinámicamente a través de las facilidades del subsistema de intervención. Para implantar este protocolo, los administradores y diseñadores instruccionales estudian los requerimientos adaptativos del escenario y las posibilidades de adaptación que ofrecen los servicios Web provistos por las herramientas. Después, definen las reglas políticas pertinentes para alterar el comportamiento de la misma puntualmente según progresa el flujo de trabajo instruccional. A modo de ejemplo, y para ilustrar las oportunidades de integración que ofrece este mecanismo, consideremos un escenario colaborativo de elección de delegados. En él los estudiantes utilizan una herramienta de votación para seleccionar al mejor candidato. Tras la finalización de la votación, Pelican debe consultar a esta herramienta, mediante el uso de servicios Web, cuál ha sido el resultado de la votación y modificar la estructura social del grupo en la plataforma para añadir un nuevo actor delegado y asignárselo al usuario seleccionado. Como puede apreciarse, en este ejemplo, el protocolo de integración consiste en recoger la información de la votación y modificar la estructura social del grupo. La integración dirigida por la plataforma se alinea con el segundo de los objetivos expuestos en esta sección – la adaptación de las herramientas al flujo de trabajo instruccional – y permite articular la integración horizontal y vertical.

- **Nivel C Integración dirigida por la herramienta.** Contrariamente a como ocurre en la integración dirigida por la plataforma, en este mecanismo de integración son las herramientas las encargadas de realizar automáticamente las alteraciones pertinentes en sus modelos internos para adaptarse a los requerimientos cambiantes del flujo de trabajo instruccional. Para poder atender a los cambios en la plataforma, Pelican proporciona a este nivel una colección de servicios Web que permiten extraer información de estado sobre los modelos de la misma. En concreto existe un servicio Web para consultar el estado del subsistema social, otro para el sistema de colaboración, otro para el de intervención y un cuatro que permite realizar consultas relacionadas con los contextos de usuario (consulte el apéndice A para obtener detalles sobre los servicios Web definidos). Las herramientas pueden utilizar estos servicios para sincronizar el estado de sus modelos internos con aquellos mantenidos por Pelican. Por ejemplo, supongamos que deseamos integrar en la plataforma un repositorio de recursos de aprendizaje de manera que esta herramienta conceda facilidades de consulta y descarga de recursos a los estudiantes y funciones de administración, eliminación y adición de nuevos recursos al profesor. Cuando un usuario accede a la herramienta desde la plataforma, ésta debe consultar los servicios Web de la misma para saber cuál es

el actor que está encarnando y asignarle así los permisos que debe concederle. Este es un claro ejemplo de cómo este mecanismo de integración ofrece un nivel elevado de interoperabilidad dando soporte a la integración vertical ascendente. Sin embargo a este respecto cabe destacar dos usos prototípicos de este mecanismo de integración.

– *Orientación a roles.* Los servicios Web proporcionados por la plataforma pueden ser utilizados, en general, para adaptar el comportamiento de las herramientas a los cambios que se produzcan en el flujo de trabajo instruccional. No obstante, como comentamos con anterioridad, uno de los principios de diseño del subsistema de integración aconseja a los desarrolladores de herramientas externas que se integren a este nivel que condicionen el comportamiento de las mismas a los roles que, en cada momento, desempeñen los usuarios que acceden a ellas.

– *Propagación de eventos externos.* Como apuntábamos anteriormente, para alcanzar el segundo los objetivos de esta sección es preciso implantar en Pelican protocolos de integración haciendo uso del subsistema de intervención. Sin embargo, las facilidades de este subsistema sólo permiten vincular reglas a los eventos surgidos dentro de la plataforma. Para implantar con éxito estos protocolos, es preciso adicionalmente que las herramientas informen puntualmente de cualquier acontecimiento pedagógicamente relevante que surja en el seno de las interacciones ocurridas dentro de las mismas. Sólo de esta manera podrá atenderse reactivamente a este tipo de eventos, que por diferenciarlos de los anteriores los denominaremos **eventos externos**. Para permitir a las herramientas comunicar este tipo de eventos a la plataforma se utilizan los servicios Web del subsistema de intervención. Por ejemplo, si el repositorio de recursos de aprendizaje del ejemplo anterior informa puntualmente a la plataforma cada vez que un usuario descarga un recurso, entonces es posible definir reglas para establecer un tratamiento adaptativo para este acontecimiento. Nótese que sin esta característica la integración horizontal no sería posible ya que no se podría determinar cuándo se producen alteraciones en la herramienta del origen de la propagación.

• **Nivel D Integración basada en componentes.** La integración basada en componentes es el mecanismo de integración más fuerte de todos los propuestos. De acuerdo a esta aproximación, para que una herramienta se integre con Pelican, ésta debe implementar un servicio Web cuya interfaz está prescrita por la plataforma. Esta interfaz consiste en una colección de métodos cuyo objeto es el de permitir a la misma controlar el ciclo de vida de las herramienta. Así, en determinados instantes Pelican invocará, a cada uno de estos métodos para indicar a la herramienta qué es lo que debe hacer. El uso de este mecanismo da lugar a una arquitectura de integración basada en componentes donde Pelican se convierte en un orquestador de las herramientas indicando cuándo éstas deben trabajar, mientras que cada herramienta es un componente pasivo que únicamente implementa el significado que tiene cada procedimiento dentro del ciclo de vida para sí

misma y que tiene un comportamiento fundamentalmente reactivo, arquitectónicamente hablando. El nivel de interoperabilidad alcanzado con esta aproximación es el más fuerte de todos los expuestos. Sin embargo, su uso supone un grado de acoplamiento fuerte con los modelos de la plataforma por lo que no suele ser muy utilizado. En efecto, este mecanismo exige a los desarrolladores de las herramientas que implementen sus sistemas de acuerdo a una arquitectura determinada, o, en el mejor de los casos, que elaboren puentes de adaptación para alinear la especificación del ciclo de vida prescrita por este mecanismo con la el modelo interno de la herramienta. En cualquier caso, toda la potencia que puede obtenerse haciendo uso de este nivel de integración puede ser alcanzado aplicando el nivel C sin la penalización de tener que seguir un interfaz. En efecto, si las capacidades del ciclo de vida se implementan en servicios Web de la herramienta, aunque sea con otra sintaxis la plataforma podrá invocarlos haciendo uso de las capacidades del modelo de intervención.

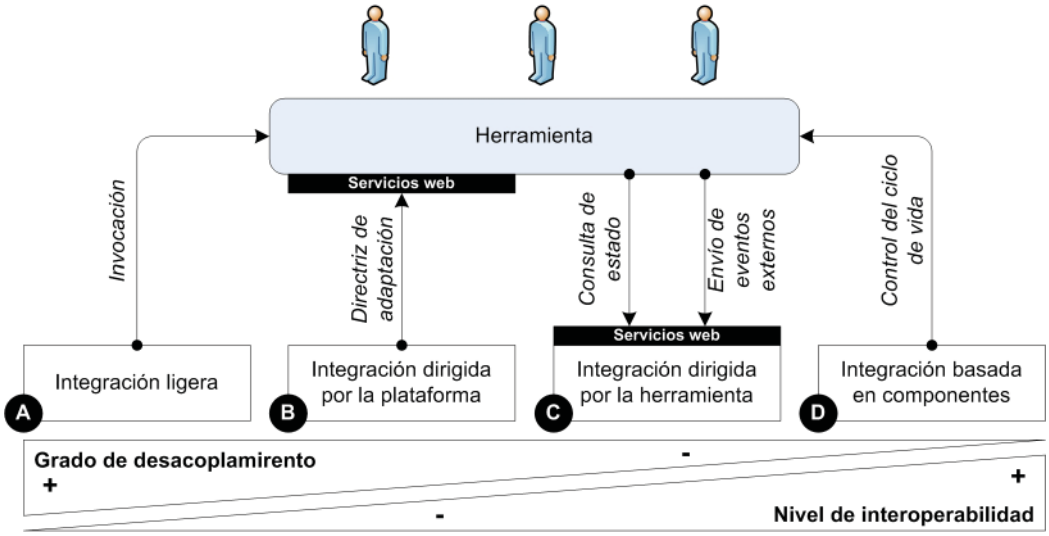

Figura 3.32. Mecanismos de integración de Pelican.

La figura 3.32 resume los cuatro mecanismos de adaptación que acabamos de describir. Como puede apreciarse, el nivel de desacoplamiento con respecto a los modelos internos de la plataforma decrece desde el primero al último. Sin embargo, a la vez se consigue un mayor nivel de interoperabilidad. Cada mecanismo proporciona un conjunto de prestaciones diferentes. Así por ejemplo, la integración ligera es utilizada para poder invocar a las herramientas desde los espacios de trabajo, la integración dirigida por la plataforma permite indicar a las mismas directrices de adaptación, la integración dirigida por la herramienta mantiene a éstas a la escucha de los cambios de estado de los modelos internos de la plataforma y permite informar de acontecimientos externos ocurridos en ellas y finalmente, la integración basada en componentes permite a Pelican controlar el ciclo de vida de las herramientas. Aunque cada mecanismo constituye un enfoque de integración diferente y supone diferentes grados de compromiso tecnológico para los desarrolladores, éstos no deben considerarse incompatibles entre sí sino potencialmente complementarios.

Las tareas de integración de herramientas externas dentro de la plataforma Pelican también son desarrolladas, al igual que en el resto de los subsistemas en las dos fases identificadas en nuestro modelo teórico. A este respecto podemos distinguir entre dos tipos de tareas:

- **Definición de servicios.** Durante la fase de diseño, los diseñadores instruccionales incorporan al modelo de colaboración del escenario colaborativo en construcción la definición de los servicios necesarios para realizar la experiencia incluyendo, para cada uno de ellos, el esquema de invocación apropiado.
- **Despliegue y adaptación de servicios.** Una vez obtenido el modelo de colaboración ampliado, los administradores pasan a desplegarlo sobre los espacios de trabajo donde la experiencia vaya a ser desarrollada. La resolución de los valores de los parámetros de cada esquema de invocación será resuelta dinámicamente cada vez que los usuarios accedan al servicio desplegado. No obstante, tal y como veremos en la sección 3.6.3, donde presentaremos las interfaces de este subsistema, los esquemas de adaptación pueden ser adaptados al contexto específico donde han sido desplegados por la sobrescritura de algunos (o todos) sus parámetros.

El desarrollo de estas tareas puede ser conceptualizado asimismo en tres fases de especificación diferentes. A continuación describimos brevemente cada una de ellas:

- **Nivel de lenguaje.** El nivel de lenguaje está formado por los artefactos del subsistema que se describirán en la siguiente subsección. En concreto los elementos principales de este nivel son la herramienta, que hace referencia a la herramienta externa que se pretende incorporar al entorno y el servicio, que como explicamos anteriormente representa un uso particular y contextualizado de dicha herramienta.
- **Nivel de modelo.** Haciendo uso de los artefactos de modelado del nivel de lenguaje, los diseñadores instruccionales pueden crear servicios de interacción específicos para dar soporte a necesidades concretas circunscritas al desarrollo de las actividades colaborativas. De esta manera, crean un **modelo de integración** que extiende las especificaciones del modelo de colaboración descritas en la sección 3.4. En efecto la especificación de servicios es una tarea de diseño que se hace sobre la definición de las plantillas de actividad, encargadas de dar soporte a dicho modelo.
- **Nivel de interacción.** En este nivel, los administradores de la plataforma despliegan el modelo de colaboración extendido con el modelo de integración lo que desencadena la aparición de un acceso contextualizado a las herramientas de soporte a la interacción desde los espacios de trabajo sobre los que se realiza el despliegue, y potencialmente adaptan estos servicios sobrescribiendo algunos de los valores de los parámetros definidos en los esquemas de invocación.

La figura 3.33, muestra un ejemplo de cómo estos niveles de especificación son utilizados en la puesta en práctica de una experiencia. En ella se utiliza una herramienta

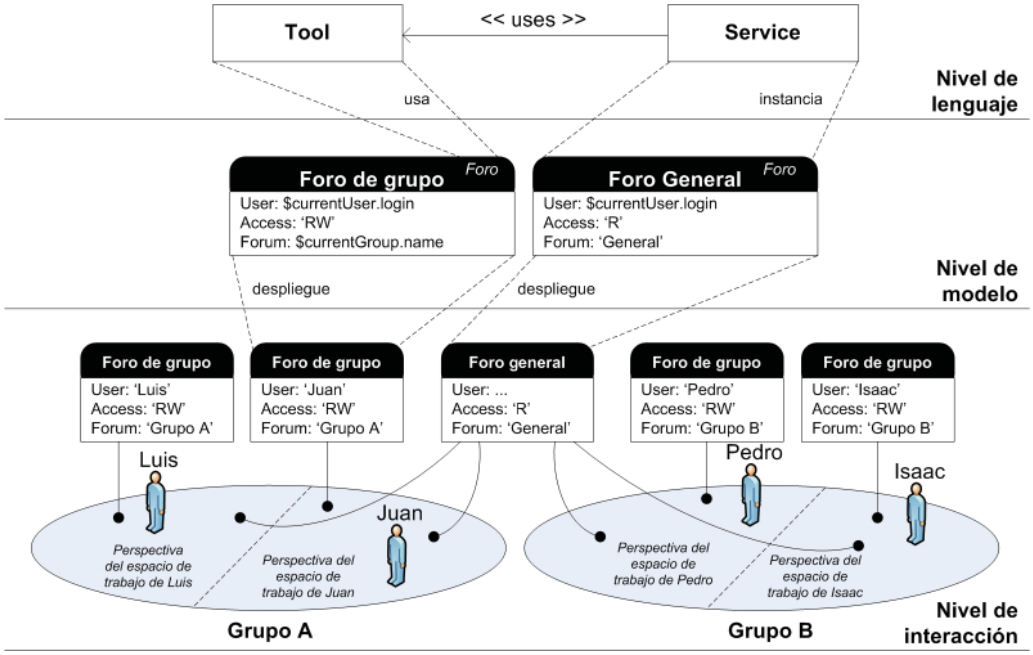

Figura 3.33. Niveles de especificación del subsistema de integración.

Foro para definir 2 tipos de servicios: foros de grupo, asociados a cada uno de los grupos que va a desarrollar la misma y foro general, compartido por todos los grupos. En el nivel de modelo se especifica que un servicio *foro de grupo* debe poder ser invocado, desde los espacios de trabajo proporcionando en el parámetro usuario el login del usuario que accede a la herramienta, en el nivel de acceso la cadena RW que indica permisos de lectura y escritura y en el identificador de foro el nombre del grupo, para que, en tiempo de desarrollo, cada grupo disponga de un foro propio. El otro servicio, *foro general*, ofrece acceso de sólo lectura e introduce la constante *General* para asegurar que durante el despliegue sólo existirá una instancia del mismo. En efecto, a nivel de despliegue puede observarse cómo al desplegar el modelo de colaboración sobre los espacios de trabajo de los grupos A y B se crea un servicio para cada grupo y un servicio compartido para ambos. En la imagen se muestra cada espacio de trabajo dividido en dos para representar la perspectiva que cada uno de los usuarios tienen de éste. Así por ejemplo Luis, desde su espacio de trabajo compartido (Grupo A) tiene acceso al *foro de grupo* de A con permisos RW y el usuario se resuelve en tiempo de acceso a *Luis*. Además dispone de un acceso al *foro general* con acceso de sólo lectura.

Dado que la especificación de la integración se produce en los tres mismos niveles que el resto de los subsistemas, las ventajas de tal descomposición aparecen también en este caso. Para no resultar reiterativos nos limitaremos a recordar que nos estamos refiriendo al desacoplamiento de las tareas de administración y diseño, a la generación de un modelo de integración, que aunque es una extensión del de colaboración constituye un producto potencialmente reutilizable y a las facilidades de administración motivadas por la centralización de la definición de servicios dentro del

modelo de integración. Sin embargo, si cabe destacar dos características propias de este subsistema que resultan de interés:

- **División de responsabilidades entre las herramientas y la plataforma.** El subsistema de integración permite separar claramente dos tipos de responsabilidades dentro de las tareas de soporte computacional al aprendizaje colaborativo. Por un lado, las herramientas son las encargadas de proporcionar servicios de interacción para posibilitar el desarrollo de las actividades de aprendizaje. Por otro, la plataforma se encarga de gestionar los aspectos de estructuración social de la comunidad virtual, de articular el flujo de trabajo instruccional y organizar el trabajo colaborativo en términos de proyectos y actividades, de proporcionar la infraestructura tecnológica necesaria para el desarrollo de las experiencias y de orquestar los protocolos de integración pertinentes que permitan la incorporación de las herramientas con un grado de interoperabilidad y adaptabilidad adecuado al flujo de trabajo instruccional.

- **Especificación de protocolos de integración.** El subsistema de integración permite a los diseñadores instruccionales definir por ellos mismos los protocolos de integración que sea necesario establecer para atender a los requerimientos del flujo de trabajo instruccional y en función de las capacidades adaptativas que ofrezca la herramienta a integrar. Esta es una gran diferencia con respecto a otras propuestas que fueron discutidas en el estado del arte de este trabajo, ya que en Pelican los esfuerzos de integración son realizados sobre las interfaces Web de la plataforma. Ademas estas labores resultan más sencillas en Pelican ya que se hacen con las facilidades del lenguaje de dominio P# y no requieren desarrollar código de integración específico dentro de cada herramienta. Esto conduce a arquitecturas más desacopladas y más fácilmente extensibles dado el menor esfuerzo que requiere la adición de nuevos servicios en la plataforma.

3.6.1. El subsistema de integración

El modelo del subsistema de integración de Pelican, se apoya sobre el modelo social y colaborativo para proporcionar los mecanismos necesarios de incorporación de las herramientas externas a los espacios de trabajo de la plataforma y obtener un alto nivel de interoperabilidad entre las mismas [Vélez, 2006] [Vélez, 2007] [Vélez & Verdejo, 2007b] [Vélez & Verdejo, 2007]. En esta sección discutiremos las responsabilidades que cumplen, a este respecto, cada una de las entidades de estos tres modelos y cómo estas se relacionan entre sí así como con las herramientas externas.

- **Service.** Tal y como se discutió anteriormente, los servicios representan un uso contextualizado de una herramienta dentro de algún espacio de trabajo de la plataforma. De esta manera, para dar acceso a una herramienta desde el entorno virtual de Pelican es preciso definir un servicio que haga referencia a la misma y que indique el valor que debe ser transmitido para cada uno de los parámetros de invocación de la misma. Las entidades pueden ser definidas formando parte directamente de un espacio de trabajo o dentro de la definición de una plantilla

de actividad. En el primer caso se pretende proporcionar acceso general a una herramienta a todos los miembros de un grupo (caso de los espacios de trabajo compartidos) o de manera individual a los usuarios de la plataforma (caso de los espacios de trabajo privados). En el segundo caso, el uso de la herramienta está circunscrito al desarrollo de alguna actividad a la que el servicio pretende dar soporte.

- **Invoking Schema.** Para proporcionar un valor a cada uno de los parámetros que requiere la invocación de un servicio se utiliza la información contenida dentro del esquema de invocación asociado al mismo. Este esquema contiene una colección de pares parámetro–valor donde cada valor es una construcción de P# que expresa, en términos abstractos y haciendo uso de las variables de contexto, el valor de cada parámetro. En el momento de acceso a un servicio desde un espacio de trabajo, la plataforma toma la URL del la herramienta y resuelve a valores reales cada uno de los parámetros indicados en el esquema para componer una URI específica y poder invocar así a la misma.

- **Tool.** Esta entidad se utiliza para referenciar a las herramientas externas desde los servicios. Cada instancia hace referencia a una herramienta externa y constituye un apoderado dentro de la plataforma para ella. Internamente podemos encontrar información para la gestión de la herramienta que se utiliza en el momento de la invocación y durante la fase de despliegue de los servicios asociados sobre los espacios de trabajo. En concreto nos estamos refiriendo a la dirección URL donde se encuentra instalada la aplicación, su icono asociado, su nombre y descripción o si dispone de servicios Web propios implementados. Asimismo, esta entidad es la encargada de controlar la herramienta mediante el ciclo de vida en caso de que ésta se adscriba al nivel de integración D. La definición de nuevas herramientas sobre la plataforma Pelican se realiza mediante un fichero de configuración en XML, de manera que cuando Pelican se arranca, se interpreta dicho fichero y se crea automáticamente una entrada Tool por cada herramienta definida en el fichero (consulte el apéndice B para obtener mas detalles al respecto). Esto permite, posteriormente simplificar la definición de servicios ya que simplemente hay que seleccionar, desde el interfaz de administración Web una herramienta y crear un nuevo servicio asociado a la misma.

- **Life Cycle.** El ciclo de vida de las herramientas es una interfaz que prescribe la colección de métodos que las herramientas externas integradas a nivel D deben implementar como servicios Web y es utilizada por las instancias de la entidad anterior para controlarlas. En concreto los métodos son los siguientes:

  – *Create.* Este método es invocado una única vez cuando de crea un nuevo servicio integrado a nivel D en la plataforma. La implementación del mismo por parte de la herramienta incluye las acciones que deben llevarse a cabo en el momento de la creación de un nuevo servicio. Por ejemplo, en una herramienta de votación podría requerirse que cada vez que se definiese un nuevo servicio se crearse, automáticamente, una nueva sesión de votación en la misma.

  - *Configure.* El método de configuración recibe como parámetro un objeto de propiedades que indican una configuración correcta para el servicio que se acaba de crear con el método anterior. La herramienta debe implementar este método para adaptar el modelo interno de la herramienta de acuerdo a la configuración solicitada. En el ejemplo anterior, la sesión de votación podría tener que configurarse para indicar el tema de la votación, la lista de opciones candidatas, el tiempo de la sesión y el grupo que participará en dicha sesión.

  - *Start.* El método de arranque permite indicar a la herramienta cuando se va a hacer uso del servicio proporcionado. Esto permite que la herramienta pueda mantener servicios inactivos hasta que no sea realmente necesario lo cual es muy útil para satisfacer requisitos no funcionales de optimización en el uso de recursos. Por tanto, la herramienta debe implementar en este método, esencialmente, toda la lógica de reserva de recursos necesarios para dar soporte a la provisión del servicio. La herramienta de votación en este caso podría reservar las conexiones a bases de datos para proceder a la votación.

  - *Stop.* El método de parada es el opuesto al de arranque y se utiliza para indicar a la herramienta que el servicio proporcionado va a dejar de ser utilizado por los usuarios de Pelican. Las herramientas deben implementar este método para incluir toda la lógica de liberación de recursos que fueron reservados durante el arranque. Cuando se desee volver a acceder al servicio será necesario volver a llamar al método de arranque para que vuelva a solicitarse una reserva de recursos. En la herramienta de votación, en este método se liberaría la conexión a la base de datos así como el histórico de interacciones que mantiene la herramienta en memoria.

  - *Destroy.* Cuando el servicio es eliminado de la plataforma, Pelican llama a este método para informar a la herramienta. Ésta debe implementar aquí toda la lógica asociada con la destrucción del servicio que generalmente consiste en eliminar la información que se incorporó en el modelo interno de la herramienta cuando se creó el servicio. En la herramienta de votación deberían eliminarse de la base de datos todos aquellos registros que hayan servido para almacenar información acerca del proceso de votación y los participantes involucrados.

- **Local Web Services.** En el nivel de integración C, las herramientas requieren consultar el estado interno de la plataforma para adaptarse a los cambios ocurridos en la misma así como informar puntualmente a ésta de todos los acontecimientos pedagógicamente relevantes que ocurren durante el uso de las herramientas. Para dar soporte a esta necesidad, la plataforma proporciona una colección de servicios Web que describimos a continuación (ver apéndice A):

  - *Social Service.* Este servicio ofrece una colección de métodos para consultar el modelo del subsistema social de Pelican. A través de él puede solicitarse información acerca de las plantillas de grupo y actores que se encuentran definidas dentro del modelo de sociedad y también acerca de los usuarios y grupos que

existen registrados en la comunidad así como conocer a qué grupo o grupos pertenece cada usuario y qué actor o actores representa.

- *Collaborative Service.* El servicio colaborativo ofrece información acerca de las plantillas de proyecto y actividad que hay declaradas en el sistema y permite tener acceso a cada una de los recursos, roles y servicios allí definidos. Además también es posible acceder a los proyectos y actividades que han sido desplegados sobre diferentes espacios de trabajo y conocer el rol que desempeña cada usuario dentro de cada actividad. De esta manera es posible adaptar dinámicamente el comportamiento de las herramientas en función del rol que los usuarios estén representando en cada momento.

- *Policy Service.* El servicio de políticas permite a las herramientas externas instalar y desinstalar políticas en la plataforma modificando de esta forma los modelos de intervención de la misma. Esto es especialmente interesante para dar soporte a herramientas de modelado que ofrezcan facilidades de diseño instruccional sobre las prestaciones ofrecidas por Pelican. Además, desde este servicio también es posible lanzar a ejecución scripts de adaptación lo cual permite realizar transformaciones que modifiquen los modelos internos de la plataforma. Y finalmente, el servicio de políticas permite a las herramientas externas integradas a nivel C comunicar eventos externos para que sean procesados por Pelican.

- *Session Service.* El servicio de sesión proporciona información acerca del contexto social y colaborativo en el que se encuentra inscrito el usuario. Todos los métodos de este servicio reciben, entre otros, un parámetro que identifica al usuario sobre el cual se quiere realizar la consulta. La información proporcionada permite conocer aspectos tales como la fecha en la que abrió sesión en el sistema, si el usuario está conectado, el grupo dentro del cual se encuentra actualmente, el espacio de trabajo en el que está participando o el proyecto y la actividad que está desarrollando. Asimismo es posible resolver expresiones en P# dentro de un determinado contexto de usuario.

- **Role.** Desde el punto de vista tecnológico el rol es una entidad que consiste en una etiqueta que identifica una colección de responsabilidades dentro de una actividad y en una colección de propiedades para su caracterización semántica. Cuando un servicio se inscribe dentro de una actividad, la herramienta externa que le da soporte puede descubrir, a través del servicio de colaboración, el rol con el que acceden a ella cada uno de los usuarios involucrados y así adaptar su comportamiento a éste . Por ejemplo, la herramienta podría mostrar un comportamiento distintos para cada uno de los tres roles (*moderador*, *votante* y *candidato*) definidos en una actividad.

- **Role Properties.** Utilizar como mecanismo de adaptación directamente el nombre de un rol puede no resultar conveniente. En efecto, si se cambia desde Pelican este nombre, la herramienta debería volver a programarse para hacer frente a este cambio. Con el fin de solucionar este problema, se añade a la definición

de rol la entidad Role Properties. Este elemento contiene una colección de pares propiedad – valor que pretenden caracterizar semánticamente al mismo. De esta manera, cada rol identifica de forma precisa un perfil de acceso para cada una de las herramientas inscritas en la definición de una plantilla de actividad. El conjunto de propiedades de un rol no constituye un vocabulario cerrado sino que es establecido por los diseñadores en tiempo de diseño. Con ello, las herramientas, en lugar de basar su lógica de adaptación en el propio role, utilizan la colección de propiedades a las que se accede desde éste. Así, la entidad Role properties constituye un artefacto que permite desacoplar la descripción sintáctica de un rol de su interpretación semántica. Por ejemplo, en la herramienta de votación los perfiles asociados a cada uno de los 3 roles (*moderador*, *votante* y *candidato*) pueden describirse incluyendo las siguientes dos propiedades booleanas a modo de permisos: *administrar*, para indicar si el rol tiene permiso de administración de la sesión de votación y *votar*, para indicar si éste puede participar en la misma. De esta manera, *moderador* se define como *administrar = cierto, votar = falso*, *votante* como *administrar = falso, votar = cierto* y *candidato* como *administrar = falso, votar = falso*. Para adaptar su comportamiento, la herramienta de votación consultará el valor de estas dos propiedades y en función de ello concederá a los usuarios permisos de administración y participación en la sesión de votación.

- **Role binding.** Para conocer el rol que está desempeñando un usuario en un momento determinado dentro de la actividad las herramientas pueden consultar la colección de entidades Role binding asociadas a la misma. Cada actividad, cuya plantilla incluye la definición de una colección de roles, dispone de una entidad de este tipo para cada rol. Internamente éstas mantienen una referencia al rol y una colección de usuarios que contiene a aquéllos que están desempeñando dicho rol. Nótese que la asignación de usuarios a roles puede alterarse en cualquier momento durante el desarrollo de la actividad, ya sea manualmente desde las interfaces de usuario de la plataforma o a través de las facilidades proporcionadas por el mecanismo de intervención. En cualquier caso, esto significa que las herramientas integradas a nivel C deben consultar el rol que está desempeñando cada usuario con cierta frecuencia para mantenerse sincronizadas con estos cambios.

- **Permission.** Cuando los servicios no se definen dentro de la descripción de una plantilla de actividad sino sobre un espacio de trabajo, la herramienta no tiene acceso a un sistema de roles y, por tanto, no puede hacer uso de los mecanismos de adaptación que se acaban de describir. Para hacer frente a este inconveniente, se propone a los desarrolladores de herramientas de nivel C que incluyan como fuente de consulta adaptativa la colección de permisos asociadas al actor (Permission). Es decir, el comportamiento de la herramienta deberá ser distinto dependiendo del actor que encarne el usuario que accede a la misma. Los permisos de actor pueden ser alterados igualmente tanto a través de las interfaces de la plataforma como mediante el uso de scripts de adaptación. Esta información se puede combinar con la información de propiedades asociada a los roles. Por ejemplo, cuando un usuario accede a un servicio definido en su espacio de traba-

jo, se pueden consultar los permisos del actor que encarna éste para determinar el perfil de acceso a la herramienta. Si el usuario accede a un servicio definido en una actividad, entonces, se puede acceder a las propiedades del rol que representa para combinarlas con las anteriores y determinar así el perfil de acceso. En caso de que exista colisión semántica entre los elementos de una y otra fuente de información pueden aplicarse estrategias para solucionarlo que prioricen, por ejemplo, la información del rol a la del actor. De esta manera el comportamiento que percibiría los usuarios de la herramienta dependería en primer lugar del rol que éstos desempeñen y en segundo lugar del actor que encarnen.

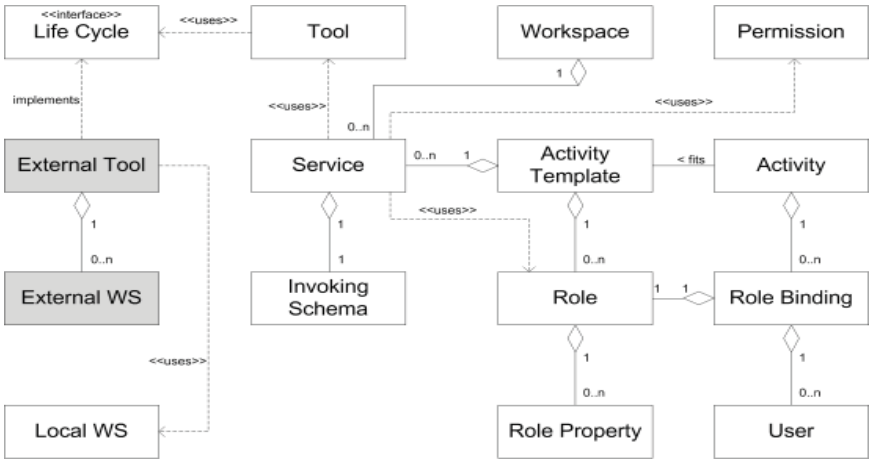

Figura 3.34. Modelo en UML del subsistema de integración.

La figura 3.34 muestra el diagrama en UML del subsistema de integración. El elemento central del modelo es la entidad service. Un servicio contiene un esquema de invocación y referencia exactamente a una herramienta. Esta entidad tiene acceso al ciclo de vida que implementan las herramientas externas a nivel D (en gris en la figura) y a través de él las controla. Además, las herramientas externas hacen uso de los servicios Web de la plataforma para adaptar su comportamiento (nivel C) y proporciona potencialmente (nivel B) una colección de servicios Web externos. Como puede apreciarse los servicios están estrechamente vinculados con el modelo de colaboración. En efecto, un servicio puede ser añadido directamente a un espacio de trabajo o a una plantilla de actividad para que luego se despliegue en la actividad correspondiente. Los mecanismos de adaptatividad basada en roles se articulan con los roles que están vinculados a cada plantilla de actividad y las propiedades (Role Property) que dependen de ésta así como con las entidades role binding que se encargan de mantener la asociación ternaria entre una actividad, un rol y la colección de usuarios que lo representan. De esta manera el comportamiento de cada herramienta dependerá del rol específico que cada usuario adquiera durante el desarrollo. Si esta asignación cambia el usuario percibirá de manera instantánea el cambio en la herramienta. Adicionalmente la adaptación puede hacerse depender de los permisos asociados al actor (Permission). En este sentido hay que recordar que mientras que estos codifican las responsabilidades de un usuario a nivel social, las propiedades de un rol representan las responsabilidades colaborativas dentro de una actividad.

3.6.2. La interfaz del subsistema de integración

Las prestaciones ofrecidas por el subsistema de integración están distribuidas a lo largo de diferentes tipos de interfaces ya presentadas en este capítulo. En concreto podemos realizar cuatro tipos diferentes de tareas desde las mismas que pasamos a describir en los siguientes apartados.

Definición de servicios

La definición de servicios permite a los diseñadores instruccionales crear servicios ofrecidos por las herramientas registradas en la plataforma. Como ya comentamos con anterioridad, esta definición puede incluirse en los espacios de trabajo privados o compartidos o en las plantillas de actividad para circunscribir su uso al desarrollo de las mismas. En los dos primeros casos, debe accederse a la administración de usuarios (figura 3.6) o a la administración de grupos (figura 3.7) y seleccionar el usuario o grupo en cuyo espacio de trabajo privado o compartido se desea incluir el servicio. En el caso de desear incluirlo sobre una actividad debe accederse a la interfaz para el diseño de plantillas de actividad y seleccionar allí la plantilla correspondiente (figura 3.13). En sendas interfaces el trabajo que debe realizarse es el mismo así que centraremos nuestras explicaciones en esta última.

Como puede apreciarse, la parte inferior de la interfaz es la que está reservada para la definición de servicios (❻). Para crear un nuevo servicio el diseñador debe seleccionar del desplegable de la esquina superior derecha el nombre de la herramienta a la que desea dar acceso y después pinchar sobre el botón de adición. Esto trae como consecuencia la creación de un formulario de edición para definir el servicio que tiene dos partes diferenciadas. En la parte superior puede indicarse una etiqueta y una descripción que será utilizada para presentar el servicio sobre el espacio de trabajo. Además pueden seleccionarse el valor de las propiedades de visibilidad que permiten ocultar e impedir el acceso al servicio. En la parte inferior por el contrario, el formulario permite definir el esquema de invocación asignando un valor a cada parámetro. El número y nombre de los mismos depende de la herramienta seleccionada al crear el servicio (consulte apéndice B). Por su parte, para eliminar un servicio sólo hay que pinchar sobre el aspa situada en la esquina superior izquierda del formulario asociado al mismo.

Adaptación de servicios

Los servicios definidos en las plantillas de actividad para permitir el desarrollo de la misma pueden ser adaptados desde la interfaz para la administración de actividades una vez realizado el proceso de despliegue sobre un espacio de trabajo determinado. En la figura 3.15 se mostró esta interfaz al presentar el subsistema de colaboración. En la parte inferior (❹) aparece un formulario de edición muy similar a los descritos en el apartado anterior. La única diferencia es que aquí cada parámetro incluye un selector para indicar si el valor del mismo desea sobrescribirse con respecto a su valor original heredado de la definición en la interfaz para el diseño de plantillas de actividad. Cuando seleccionamos un parámetro para modificarlo, el campo de texto

pasa a ser editable lo que permite introducir un valor diferente para el mismo. Si lo desmarcarmos el parámetro se restaura al valor original y el campo de texto pasa a ser no editable. En el ejemplo de la figura puede observarse como los dos primeros parámetros están sobrescritos y los dos segundos mantienen su valor original.

Acceso a servicios

Los servicios pueden aparecer vinculados a dos sitios distintos dentro de las interfaces de la plataforma: directamente en los espacios de trabajo o en las actividades desplegadas dentro de los mismos. En efecto, al acceder a un espacio de trabajo compartido o privado puede aparecer, abajo del todo, una sección de herramientas que contiene la colección de servicios definidos. Esta sección puede no mostrase si no existe ningún servicio incluido en dicho espacio, tal y como ocurre en el interfaz de la figura 3.16. Por otro lado, como puede verse en la figura 3.18, al acceder a una actividad cuya plantilla haya incluido la definición de servicios también se muestra un área de herramientas para dar acceso a los mismos (❺).

Adaptación de servicios orientada a roles

Como explicamos con anterioridad, uno de los principios de diseño del subsistema de integración es orientar la adaptación de las herramientas hacia el rol que desempeñan los usuarios en cada momento. Desde las interfaces de la plataforma es posible definir la colección de roles que utiliza un determinado tipo de actividad y establecer manualmente una asignación inicial de los mismos a los miembros del grupo así como modificar posteriormente dicha asignación durante su desarrollo.

La definición de roles es llevada a cabo por los diseñadores instruccionales desde la interfaz para el diseño de plantillas de actividad mostrado en la figura 3.13. En el área de definición de roles (❺) aparece, en primer lugar, una relación de todos los roles definidos. El diseñador puede seleccionar uno de ellos para editarlo sobre el formulario que aparece debajo. Para crear y borrar roles se utilizan los botones que aparecen al lado del campo para el nombre en el formulario. Al crear uno nuevo es necesario asignarle un nombre y opcionalmente una descripción. Las propiedades del rol pueden ser creadas, borradas y modificadas con los elementos del resto del formulario. Como puede apreciarse en el ejemplo de la figura, por construcción todos los roles incluyen cuatro propiedades por defecto con prefijo pelican: pelican.references.locked y pelican.services.locked permite indicar si el rol puede acceder a los recursos y los servicios y pelican.roles.changeable y pelican.state.changeable indican si usuario con ese rol puede alterar, desde el espacio de trabajo la asignación de roles y el estado de la actividad.

Para llevar a cabo la asignación inicial de roles, el profesor debe acceder a la interfaz para la administración de actividades mostrada en la figura 3.15. Allí existe un área de asignación de roles (❸) que permite asociar a cada usuario un rol. Durante el desarrollo de la actividad esta asignación puede modificarse directamente sobre la actividad desplegada tal y como se ilustra en la figura 3.18 (❸). Esto permite a los estudiantes realizar esta asignación según crean conveniente.

3.7. Conclusiones

A lo largo de este capítulo hemos presentado la arquitectura de la plataforma Pelican profundizando en cada uno de los cuatro subsistemas que la constituyen. Las funcionalidades ofrecidas por cada uno estos subsistemas permiten dar soporte a la especificación formal y computable de todos los aspectos que es necesario tener en cuenta cuando se van a poner en práctica experiencias formativas de carácter colaborativo.

En efecto, el subsistema social permite definir un modelo de sociedad formado por plantillas de grupo y actores para dar forma a la comunidad virtual de aprendizaje. Con el subsistema de colaboración es posible organizar el trabajo colaborativo del escenario en términos de plantillas de proyecto y actividad y proporcionar la infraestructura tecnológica necesaria para desplegar esta especificación sobre diferentes contextos sociales. El subsistema de integración completa esta definición para permitir la incorporación de herramientas externas que ofrecen servicios de soporte a la interacción colaborativa. La responsabilidad de este subsistema es proporcionar los mecanismos necesarios para ofrecer un nivel de interoperabilidad y adaptabilidad al flujo de trabajo instruccional adecuado. Y finalmente, el subsistema de intervención se encarga de capturar todos los aspectos dinámicos inherentemente vinculados al desarrollo de este tipo de experiencias para permitir transformar tanto los modelos internos de la plataforma como los de las herramientas externas integradas en la misma a tenor de los requerimientos del flujo de trabajo instruccional.

En cada una de las secciones donde se discutieron estos subsistemas se presentaron las ventajas que ofrecen cada uno de ellos. No obstante, si cabe citar aquí dos características diferenciales fundamentales:

- **Especificación de escenarios de aprendizaje por niveles.** La plataforma Pelican articula las tareas de diseño y desarrollo de experiencias de aprendizaje colaborativo en tres niveles de especificación.
  - *Nivel de lenguaje.* Este nivel está formado por todos los artefactos de modelado que son proporcionados por cada uno de los cuatro subsistemas de la arquitectura tales como actores, plantillas de grupo, plantillas de proyecto y actividad, recursos, servicios, roles, reglas, scripts de adaptación, etc.
  - *Nivel de modelo.* En el nivel de modelo, los diseñadores instruccionales utilizan el lenguaje de la plataforma para definir los cuatro modelos constituyentes de la especificación completa de un escenario colaborativo: modelo de sociedad, modelo de colaboración, modelo de integración y modelo de intervención.
  - *Nivel de despliegue.* En el nivel de despliegue los administradores construyen una comunidad virtual, despliegan el trabajo colaborativo sobre los espacios de trabajo, adaptan cada uno de los servicios definidos y realizan la asignación inicial de roles. Además, de manera automática, la plataforma aplicará puntualmente scripts de adaptación para llevar a cabo las transformaciones pertinentes que permitan avanzar el flujo de trabajo instruccional.

- **Obtención de productos reutilizables.** Cada uno de los cuatro modelos que conforman la especificación de los escenarios de aprendizaje colaborativo constituyen productos potencialmente reutilizables en otros contextos y situaciones pedagógicas. Por ejemplo, un escenario de Jigsaw sobre un determinado dominio de conocimiento requerirá el mismo modelo de estructuración social, el mismo tipo de actividades y la misma lógica de transformación adaptativa que otros escenarios basados en el mismo método pedagógico.

Para ilustrar las capacidades proporcionadas por la plataforma Pelican, en la tabla 3.5 se resumen las tareas de diseño y desarrollo que es necesario llevar a cabo, en cada uno de los tres niveles anteriores para poner en práctica un escenario de aprendizaje sobre la electricidad. En concreto se muestran, los elementos del nivel de lenguaje, el uso de los mismos a nivel de modelo y su explotación a nivel de despliegue. Asimismo se alinean estos dos últimos niveles con las fases de diseño y desarrollo asociando a cada una los actores implicados.

	Subsistema Social	Subsistema de colaboración	Subsistema intervención	Subsistema de integración
Nivel de lenguaje	Elementos para estructurar la comunidad: Actor Plantilla de grupo Usuario Grupo	Elementos para organizar el trabajo de los estudiantes: Plantilla de proyecto Plantilla de actividad Espacio de trabajo Proyecto Actividad Referencia	Elementos para definir estrategias de intervención adaptativa: Política Regla política Script de adaptación	Elementos para integrar herramientas dentro del entorno: Herramienta Servicio Servicios Web Rol Propiedades de rol
Nivel de modelo	**Modelo de sociedad** - Actores - estudiante - profesor - monitor - Plantillas de grupo - colegio - clase - grupo - par	**Modelo de colaboración** - Plantilla proyecto - Electricidad... - Plantillas actividad 1. Ley de Ohm 2. Leyes Kirchoff	**Modelo de intervención** - Reglas políticas - Secuenciamiento - 1 round robin - 2 debate abierto - automatizar - evaluación - monitorización	**Modelo de integración** - Servicios 1. chat y repositorio 2. foro y votación - Roles 1. worker, observer 2. lider, worker
Nivel de despliegue	**Comunidad** - Grupo A - Luis (estudiante) - María (estudiante) - Juan (estudiante) - mmillan (profesor) - Grupo B - Oscar (estudiante) - Pedro (estudiante) - Isaac (estudiante) - mmillan (profesor)	**Despliegue** - Workspace A - Electricidad started 1. Passed 2. In development - Worksspace B - Electricidad started 1. In development 2. Init	**Aplicación** - Reglas A - Control de turnos - Avisos al monitor - Reglas B - Cambio de roles - Avisos al profesor	**Interacción** - Workspace A - Luis (lider) - María (worker) - Juan (worker) - mmillan (teacher) - Workspace B - Oscar (worker) - Pedro (observer) - Isaac (worker) - mmillan (teacher)

Fase de diseño: administrador, diseñador

Fase de desarrollo: profesor, estudiantes, monitor

Tabla 3.5. Aportación de cada subsistema a la especificación de un escenario.

4 Desarrollo de pruebas y resultados

La plataforma Pelican que hemos presentado en el capítulo anterior proporciona una colección de funcionalidades que permiten especificar las estructura social de una comunidad virtual y mantenerla a lo largo del tiempo, organizar el trabajo colaborativo de los estudiantes en términos de proyectos y actividades y proporcionar la infraestructura tecnológica necesaria para su desarrollo, integrar herramientas externas de soporte a la interacción colaborativa y ofrecer los mecanismos necesarios para alcanzar un nivel de interoperabilidad adecuado entre las mismas y aplicar automáticamente scripts de adaptación para poder transformar de forma dinámica el entorno de aprendizaje a tenor de los requerimientos de cambio del flujo de trabajo instruccional.

La pregunta que cabe hacerse ahora es si con estas prestaciones hemos dado alcance a los objetivos identificados en el capítulo primero de nuestro trabajo. Esto es, dar completo soporte tanto al diseño como al desarrollo de experiencias de aprendizaje colaborativo. Para responder a esta pregunta hemos desarrollado una serie de pruebas cuyos resultados discutimos a lo largo de este capítulo. Presentaremos, en primer lugar, el ecosistema tecnológico utilizado para comprobar la viabilidad de la plataforma. Abordaremos la descripción de una librería de scripts con soluciones para problemas recurrentes en este tipo de experiencias formativas. Discutiremos cómo las prestaciones de Pelican permiten dar soporte a la especificación de cinco escenarios pedagógicos de ejemplo complejos y detallaremos las experiencias que se han desarrollado en situaciones reales de educación formal.

4.1. Introducción

Para poner a prueba la viabilidad de nuestro trabajo de tesis se han llevado a cabo una serie de pruebas conceptuales y en escenarios reales que se encuentran inscritas dentro del proyecto [ENLACE], cuyos objetivos generales fueron descritos en la sección 1.4. Asociado al mismo ha germinado, durante los 5 años de duración del proyecto, un ecosistema tecnológico donde destacan, de un lado, la plataforma de integración Pelican y, de otro, una familia de herramientas específicas para dar soporte a diversas tareas y protocolos de interacción típicamente relacionadas con el desarrollo de experiencias de aprendizaje colaborativo. En la siguiente sección presentaremos en detalle todas estas herramientas y el nivel de integración que alcanzan con Pelican, lo cual resulta fundamental para poner en contexto el resto de experiencias realizadas. A partir de estos esfuerzos de desarrollo ha sido posible realizar tres familias de pruebas que serán descritas en las secciones subsiguientes de este capítulo. A continuación resumimos brevemente cada una de ellas:

- **Librería de scripts de adaptación P#.** Uno de los esfuerzos que han sido realizados para llevar a cabo el desarrollo de las pruebas ha sido la construcción de una librería de scripts de adaptación que vinieran a resolver eficientemente una colección de problemas que aparecen de forma recurrente durante el diseño de escenarios de aprendizaje colaborativo. En efecto, dado que la implantación de los aspectos dinámicos de un escenario tiene un grado de complejidad comparativamente superior con respecto a aquél que presentan el resto de los subsistemas se vio necesario y conveniente proporcionar a los diseñadores esta librería. Su uso es tecnológicamente posible gracias a la existencia de la tarea *callScript* que permite invocar scripts de adaptación desde otros scripts y pasarles unos argumentos actuales en la llamada (consulte la tabla 3.3 para conocer la colección de tareas de P#).

- **Experimentación en escenarios reales de educación formal.** A lo largo del tiempo de desarrollo, el proyecto ENLACE ha supuesto un marco de evaluación excepcional para probar nuestra plataforma en escenarios reales de educación formal con estudiantes de enseñanza secundaria. Se trataba, en este sentido, de poner a prueba la viabilidad en el uso de la plataforma inmersa en un curriculum específico y constreñido a las limitaciones temporales necesarias para llevar a cabo la experiencia tanto en el aula como en laboratorios y en excursiones a distintos ecosistemas naturales.

- **Cinco escenarios colaborativos complejos.** Para verificar que las prestaciones proporcionadas por los cuatro subsistemas de Pelican, y en espacial por el subsistema de intervención, dan soporte completo al diseño y desarrollo de experiencias de aprendizaje colaborativo se han implantando en Pelican cinco escenarios complejos inspirados en diferentes métodos pedagógicos descritos en la literatura [Barkley et al., 2005]. El subsistema de intervención de Pelican proporciona, a este respecto, una colección de facilidades que hace posible capturar en tales especificaciones los aspectos dinámicos inherentes a los mismos.

4.2. Ecosistema tecnológico de ENLACE

Una de las aportaciones principales del proyecto ENLACE es el ecosistema tecnológico que se ha desarrollado a lo largo de sus cinco años de duración [Verdejo et al., 2009]. Dicho ecosistema está constituido por una colección de herramientas de soporte a la interacción colaborativa que han permitido llevar a cabo diversas pruebas de evaluación en condiciones controladas (apartado 4.3 y 4.5) así como múltiples experimentos en escenarios reales de educación formal (apartado 4.4).

Antes de pasar a describir la familia de herramientas es preciso abordar la estrategia de integración que ha sido utilizada para obtener este ecosistema. En concreto, ésta hace uso de tres de los cuatro mecanismos que fueron descritos en el capítulo anterior. A continuación precisaremos cómo son particularmente explotados estos mecanismos para proporcionar un alto nivel de interoperabilidad y cohesión:

- **Integración Ligera (Nivel A).** Como es preceptivo, todas las herramientas del ecosistema de ENLACE se incorporan a los espacios de trabajo y actividades de la plataforma mediante la declaración de servicios. Cada uno de estos servicios constituye un uso contextualizado de una herramienta que permite a los miembros de un grupo acceder a uno de los servicios de interacción ofrecidos por la misma. Las herramientas desarrolladas dentro del proyecto ENLACE ofrecen, en este sentido, diversas prestaciones y tienen en consideración ciertos elementos del contexto socio–colaborativo donde éstas se utilizan, tales como el usuario que las accede o el grupo al que pertenece.

- **Integración dirigida por la plataforma (Nivel B).** Con la integración dirigida por la plataforma Pelican puede llevar a cabo protocolos de integración que permiten transformar adaptativamente los modelos propios de las herramientas externas según avanza el flujo de trabajo instruccional. Dentro del ecosistema tecnológico de ENLACE esto es llevado a cabo a través del uso de tres artefactos:

  – *Uso de servicios Web.* Los servicios Web ofrecidos por las herramientas externas proporcionan un punto de acceso programático a las mismas. En el ecosistema de ENLACE, algunas de las herramientas implementan servicios para la administración y gestión de características propias de las mismas que ofrecen la posibilidad a la plataforma de adaptar puntualmente sus modelos a tenor de los requerimientos del flujo de trabajo instruccional.

  – *Uso de reglas políticas.* Los protocolos de orquestación consisten, esencialmente, en la aplicación puntual de transformaciones adaptativas sobre los modelos internos de la plataforma y sobre los de las herramientas externas integradas en el entorno mediante la invocación de las operaciones que, a tal efecto, ofrecen los servicios Web de las mismas. Para especificar estos protocolos, los diseñadores instruccionales incluyen las reglas pertinentes en el modelo de intervención del escenario, lo que permite que tales transformaciones tenga efecto en respuesta a los acontecimientos ocurridos en el entorno – tanto la plataforma como las herramientas – durante el desarrollo de la expe-

riencia. En el caso de ENLACE, estas facilidades del subsistema de intervención se han utilizado para la implantación en Pelican de ciertas políticas que serán descritas en mayor detalle en la sección 4.5.

– *Uso de objetos de aprendizaje.* Los objetos de aprendizaje son una representación digital de los recursos pedagógicos que intervienen, de una u otra forma, a lo largo de un proceso instructivo [Wiley, 2000] [IMS–CP]. En escenarios no colaborativos, estos objetos son construidos por los diseñadores instruccionales para proporcionar material didáctico a los estudiantes. Sin embargo, en escenarios de colaboración los objetos de aprendizaje son productos generados dinámicamente por las herramientas como consecuencia del uso que los estudiantes hacen de las mismas a lo largo del desarrollo de la experiencia [Hoppe et al., 2005]. De esta manera, cada objeto constituye típicamente una representación parcial y tangible del trabajo colaborativo de los estudiantes y del estado en que se mantuvo el desarrollo de la experiencia durante cierto momento del tiempo. Sin embargo, en el ecosistema tecnológico de ENLACE, los objetos de aprendizaje juegan también un papel preponderante en la integración. En efecto, la mayoría de las herramientas del ecosistemas son productoras y/o consumidoras de estos recursos. Pero adicionalmente, estos objetos son aquí artefactos computables con una estructura interna bien definida [Vélez & Celorrio, 2005], lo que permite a Pelican interpretar su contenido y aplicar, en función de éste, transformaciones adaptativas en el entorno. En la sección 4.4 pueden encontrarse numerosos ejemplos de situaciones donde se realiza una adaptación basada en el uso de objetos de aprendizaje que reflejan típicas decisiones colaborativas alcanzadas por consenso. En este sentido, los objetos de aprendizaje pueden ser considerados como un elemento vehicular que permite poner en contacto unas herramientas con otras y desarrollar esquemas de integración tanto verticales como horizontales.

• **Integración dirigida por la herramienta (Nivel C).** En este mecanismo de integración las herramientas externas utilizan los servicios Web de la plataforma para consultar el estado de la misma y adaptan sus modelos internos para sincronizarlos con dicho estado. Para cada herramienta en concreto, esta sincronización adaptativa suele centrarse en algunos elementos específicos del contexto socio–colaborativo de los usuarios (consulte la tabla 3.2 para obtener detalles sobre el contexto). Sin embargo, dentro del ecosistema tecnológico de ENLACE podemos distinguir, en este sentido, dos tipos de adaptaciones a las que se adscriben las herramientas. A continuación describimos brevemente las mismas:

– *Adaptación basada en la estructura social.* El caso más frecuente en ENLACE de integración dirigida por la herramienta basa su adaptación en la información social que la plataforma mantiene del usuario que accede a la herramienta. En concreto se utilizan las credenciales del usuario (login y password) para identificar a éste como un miembro de la comunidad virtual con permisos de acceso a la misma y, adicionalmente en otros casos, se utiliza el grupo con el que dicho usuario se encuentra trabajando en el momento actual.

– *Adaptación basada en la asignación de roles.* Tal y como aconsejan los principios del subsistema de integración, otro de los aspectos del contexto de usuario que puede ser utilizado por las herramientas para adaptar su comportamiento con la plataforma es el rol colaborativo que desempaña en cada momento el usuario dentro de la actividad, y más específicamente las propiedades que lo definen. Dado que tanto la asignación de roles como la definición de estas propiedades puede ser alterada dinámicamente durante el transcurso de la experiencia, la herramienta debe asegurarse de mantener sincronizado su estado con la configuración de estos aspectos.

Una vez analizadas las estrategias de integración utilizadas para incorporar las diversas herramientas al entorno tecnológico del proyecto ENLACE estamos en disposición de pasar y presentar las mismas. Para cada una de ella haremos una breve descripción de sus características y funcionalidades principales y discutiremos cómo y a qué niveles se produce la integración con Pelican:

- **LOR.** La herramienta LOR (del inglés, Learning Object Repository) es un repositorio de recursos que permite almacenar, gestionar y recuperar objetos de aprendizaje encapsulados de acuerdo a un modelo de empaquetamiento inspirado en [IMS–CP]. La ventaja principal con respecto a otras propuestas [Merlot] [Ariadne] radica en que la herramienta LOR permite a los administradores definir diferentes tipos de objetos y asociarles un esquema de meta–documentación propio lo cual aumenta las capacidades de búsqueda. En efecto, cada tipo de objeto de aprendizaje identifica un tipo de producto diferente tal como un documento, una imagen, un sonido, una discusión, una votación, etc. Por tanto su caracterización para la búsqueda requerirá un conjunto de metadatos cuando menos parcialmente diferente. Esto supone una mejora con respecto a la aproximación generalista de [IMS–LOM] que ofrece un único esquema de metadatos en general desajustado a cada tipo de objeto. La otra ventaja significativa de LOR es que ofrece facilidades para el control de las versiones de los objetos de aprendizaje almacenados en el repositorio. De cada versión pueden generarse nuevas versiones con posibles modificaciones con respecto al anterior bien en su contenido o en sus metadatos. Además, cada versión queda vinculada a su anterior por una relación de parentesco lo que permite capturar el proceso de desarrollo de una experiencia formativa como una cadena arborescente de versiones de objetos, representando cada una de ellas un estado significativo de dicho proceso. Para permitir el uso de esta herramienta dentro del contexto tecnológico de ENLACE se utilizan los tres mecanismos de integración que describimos con anterioridad:

  – *Integración Ligera.* En el ecosistema de ENLACE se han utilizado cuatro tipos de servicios proporcionados por LOR: 1) almacenar un objeto en el repositorio, 2) navegar por los objetos almacenados en el mismo, 3) descargar uno de ellos en local y 4) ver en línea su contenido. Todos estos servicios requieren que se pase como parámetro las credenciales de usuario. Adicionalmente es necesario para el primero, el identificador del repositorio, para el segundo y el tercero, el identificador del objeto y para el cuarto, el repositorio y el tipo de objeto.

  - *Integración dirigida por la plataforma.* La implantación de muchos escenarios colaborativos, como por ejemplo aquéllos descritos en la sección 4.5, requiere de la inclusión de reglas en el modelo de intervención para extraer e interpretar objetos almacenados previamente en el repositorio por otras herramientas o sesiones de trabajo. Los servicios Web de LOR permiten llevar a cabo estas tareas. Sin embargo, existen otras labores de carácter administrativo relacionadas con el uso de LOR que también pueden ser automatizadas mediante reglas y scripts de adaptación. En concreto nos estamos refiriendo a la creación de repositorios y a la asignación de permisos de acceso para alinear éstos con los espacios de trabajo de la plataforma. Este alineamiento es un principio general de diseño de todos los escenarios de ENLACE.

  - *Integración dirigida por la herramienta.* LOR utiliza los servicios Web proporcionados por la plataforma para comprobar la validez de las credenciales de usuario que son proporcionadas en los parámetros de invocación. Cuando LOR es accedido desde los espacios de trabajo de la plataforma, esta comprobación resulta redundante ya que es precisamente la plataforma quien proporciona el valor de las credenciales. Sin embargo, no debemos olvidar que LOR puede ser también utilizado como una herramienta autónoma en cuyo caso esta comprobación resulta imprescindible. Además LOR también envía eventos externos a la plataforma para que sean tratados puntualmente por esta.

- **CARDS.** La herramienta CARDS permite crear fichas para registrar datos producidos durante el desarrollo de las experiencias colaborativas. Para cada tipo de ficha ha de definirse una plantilla de ficha que constituye un modelo de información para ésta. Este modelo indica la colección de secciones, campos de datos y tipos de datos que contienen las fichas asociadas al mismo. Generalmente la responsabilidad de diseñar las plantillas de fichas recae sobre los diseñadores instruccionales mientras que los estudiantes son los encargados de crear y editar las fichas para registrar las observaciones relacionadas con el desarrollo de alguna actividad. Sin embargo, en las últimas versiones, la herramienta CARDS permite también que sean los propios estudiantes los que diseñen sus propias plantillas, lo cual ofrece mayores posibilidades a la hora de articular experiencias colaborativas. Tanto las fichas como las plantillas de ficha son persistidas por la herramienta LOR como objetos de aprendizaje. En el primer caso éstas se almacenan típicamente en el repositorio del grupo que las generó y a veces en el de clase. En el segundo caso, las plantillas de ficha son almacenadas en un repositorio especial que contiene exclusivamente a todas ellas para que puedan ser consultadas por la herramienta CARDS cuando sea necesario.

  - *Integración Ligera.* En las experiencia de ENLACE se han definido tres tipos de servicios de la herramienta CARDS: 1) crear una nueva ficha, 2) editar una ficha ya existente y 3) diseñar una nueva plantilla de ficha. Todos estos servicios requieren que se pasen como parámetro las credenciales de usuario. En los dos primeros también es necesario el repositorio donde reside o residirá la ficha y en el segundo además requiere el identificador de ésta dentro del repositorio.

  - *Integración dirigida por la plataforma.* La herramienta CARDS no ofrece servicios Web para permitir a la plataforma interoperar con la misma de manera que en CARDS este tipo de integración no se produce.

  - *Integración dirigida por la herramienta.* CARDS ofrece dos modos de acceso a la herramienta que ofrecen diferentes funcionalidades: una para la creación y edición de fichas y otra para el diseño de plantillas de ficha. Cada una de ellas se accede con un role diferente. En concreto cuando el usuario que accede a la herramienta desde Pelican lo hace desempeñando un rol que contiene la propiedad *cards.mode* con valor *edit* se accede al modo de edición y cuando tiene el valor *design* se accede al modo de diseño

- **AGORA.** AGORA es una herramienta de soporte a la toma de decisiones colaborativas. Con ella, los miembros de un grupo pueden realizar distintos tipos de votaciones: simples, con ronda de desempate, secretas, descubiertas, etc. El tipo de opciones candidatas de una votación también es un aspecto configurable. En concreto, en las votaciones de AGORA se puede elegir entre votar textos, imágenes, objetos de aprendizaje almacenados en la herramienta LOR e incluso miembros del grupo que está implicado en la votación. En ellas existen tres tipos de participantes: el moderador, encargado de realizar las labores de control sobre flujo de trabajo de la votación; los candidatos, cuya participación en la misma puede ser restringida o permitida y los votantes que eligen su opción preferida. Al finalizar una sesión de votación, la herramienta genera una ficha CARDS y la almacena en el repositorio del grupo implicado. Esta ficha contiene un acta sobre el desarrollo de la misma indicando el texto de la votación, las opciones candidatas, los usuarios participantes y la traza con las votaciones que realizó cada uno de ellos. La herramienta AGORA ofrece a los usuarios tres vistas diferentes: la del ordenador del moderador, la que se proyecta en el aula, que es común a todos los estudiantes y la de el ordenador de cada estudiante.

  - *Integración Ligera.* El acceso a la herramienta AGORA a través de los espacios de trabajo de la plataforma requiere la definición de un servicio que envíe a ésta como parámetros las credenciales del usuario y el grupo al que pertenece. Una vez autenticado, éste pueden acceder a la sesión en curso y seguir las instrucciones para realizan la votación.

  - *Integración dirigida por la plataforma.* AGORA ofrece una colección de servicios Web que permiten llevar a cabo tareas de administración relacionadas con las sesiones de votación mantenidas por la herramienta. Haciendo uso de estas prestaciones y de las facilidades de integración de Pelican, la plataforma puede crear, configurar, arrancar, parar y destruir sesiones.

  - *Integración dirigida por la herramienta.* Este tipo de integración es utilizado por AGORA para comprobar la validez de las credenciales de usuario proporcionadas en la invocación de la herramienta así como para extraer información del modelo social cuando la votación implica a los miembros del grupo como candidatos. Además AGORA también envía eventos a Pelican para informar de

los acontecimientos relevantes que ocurren durante el transcurso de la sesión de votación. Estos eventos pueden ser asociados a reglas políticas para ejecutar scripts de adaptación que ofrezcan una respuesta reactiva a los mismos.

- **CHAT.** CHAT es una herramienta de mensajería instantánea que permite poner en contacto a todos los miembros de un grupo que participan en el desarrollo de una actividad. Esta herramienta distingue tres modos de funcionamiento: debate abierto, debate por turnos y debate por solicitud de turno. En el debate abierto todos los participantes pueden intervenir en cualquier momento en la discusión. En el debate por turnos la herramienta va asignando consecutivamente turnos de intervención a cada miembro del grupo. Durante cada turno de intervención sólo un usuario puede intervenir mientras los otros permanecen a la escucha. Finalmente, en el debate por solicitud de turnos, para intervenir los usuarios tienen que solicitar a la herramienta un turno de palabra. Ésta recoge la solicitud y, como en un debate, le concederá el turno de palabra sólo cuando el resto de usuarios que pidió intervenir previamente hayan consumido su uso de palabra.

  – *Integración Ligera.* Para incorporar una sesión de CHAT en los espacios de trabajo de la plataforma es necesario definir un servicio asociado a la herramienta. Este servicio debe proporcionar información sobre las credenciales de usuario así como un nombre para identificar unívocamente la sala de discusión donde van a reunirse los participantes. Lo mas común es asociar un salón para cada grupo que desarrolla una experiencia por tanto el valor de este parámetro suele coincidir con el nombre del grupo o con una composición de este y la actividad que está desempeñando. Adicionalmente, el servicio también debe proporcionar información acerca del modo de interacción que desea utilizarse (abierta, por turnos o por solicitud de turnos).

  – *Integración dirigida por la plataforma.* Este tipo de integración no es utilizada para integrar CHAT dentro de la plataforma ya que no existen servicios Web externos proporcionados por la herramienta.

  – *Integración dirigida por la herramienta.* Los modos de interacción basados en turnos (debate por turnos y debate por solicitud de turno) delegan la gestión de la concesión de turnos en la plataforma a través del uso de roles. En concreto los roles que se espera desempeñen los usuarios deben tener definidas las propiedades booleanas: *chat.read*, para indicar si el usuario puede leer mensajes y *chat.write*, para indicar si puede escribirlos. Para consultar estos valores se hace uso del servicio de colaboración de la plataforma. De esta manera, la política de concesión de turnos puede ser llevada a cabo por Pelican para ser alineada con los requerimientos pedagógicos del escenario. El otro aspecto que fomenta la integración dirigida por la plataforma es que CHAT también envía a ésta la colección de eventos externos ocurridos durante su uso en las sesiones de debate. En concreto, cada vez que un usuario entra o sale de un salón de discusión, cada vez que envía un mensaje y cada vez que solicita un turno de intervención la plataforma recibe un evento de la herramienta.

- **COMPO.** La herramienta COMPO permite a los diseñadores instruccionales definir una representación virtual de un itinerario en la naturaleza, que es almacenado en el repositorio en forma de ficha para que los estudiantes posicionen sobre él las observaciones que encuentren a lo largo de su recorrido. Estas observaciones son a su vez también recogidas como fichas de CARDS que contienen, además de los datos sobre la observación, información relativa a las posiciones geográficas donde se realizó la observación.
  - *Integración Ligera.* Para posibilitar el uso de COMPO en Pelican es necesario crear un servicio que proporcione a la herramienta información de las credenciales de usuario y el repositorio donde se almacenan las fichas del itinerario.
  - *Integración dirigida por la plataforma.* COMPO no ofrece servicios Web que permitan a la plataforma establecer una comunicación con la herramienta. Por tanto este tipo de integración no es utilizado en este caso.
  - *Integración dirigida por la herramienta.* La herramienta COMPO explora la información del subsistema social de la plataforma a través de los servicios Web proporcionado por ésta para validar la autenticidad de las credenciales de usuario proporcionadas en la invocación. Además extrae información relativa a los miembros constituyentes del grupo en curso en que se encuentra el usuario para poder gestionar un Chat propio que esta herramienta tiene incorporado.
- **COMET.** COMET es una extensión colaborativa y basada en objetos de aprendizaje del editor de mapas conceptuales CMED [Larrañaga et al., 2002]. Los elementos de estos mapas están constituidos típicamente por conceptos y relaciones entre los mismos. Sin embargo COMET permite adicionalmente insertar objetos almacenados en los repositorios gestionados por LOR tales como imágenes, o fichas de CARDS. Además, cada mapa conceptual de COMET es guardado también en forma de ficha en el repositorio del grupo que desarrolla el mismo.
  - *Integración Ligera.* El acceso de COMET a la herramienta requiere, como en los casos anteriores, de la creación de un servicio que proporcione las credenciales del usuario. Sin embargo, tecnológicamente el proceso de apertura de sesión es un poco más complejo que en los casos anteriores ya que COMET no es una herramienta Web sino una aplicación de escritorio arrancada haciendo uso del *Web Start* de [Java]. Esto ralentiza el proceso de arranque del editor pero resulta transparente al usuario.
  - *Integración dirigida por la plataforma.* COMET no se integra a este nivel con la plataforma puesto que no dispone de servicios Web para controlar la herramienta de forma remota.
  - *Integración dirigida por la herramienta.* La herramienta COMET hace uso del servicio social de Pelican para validar las credenciales de usuario que le son proporcionadas durante la invocación.

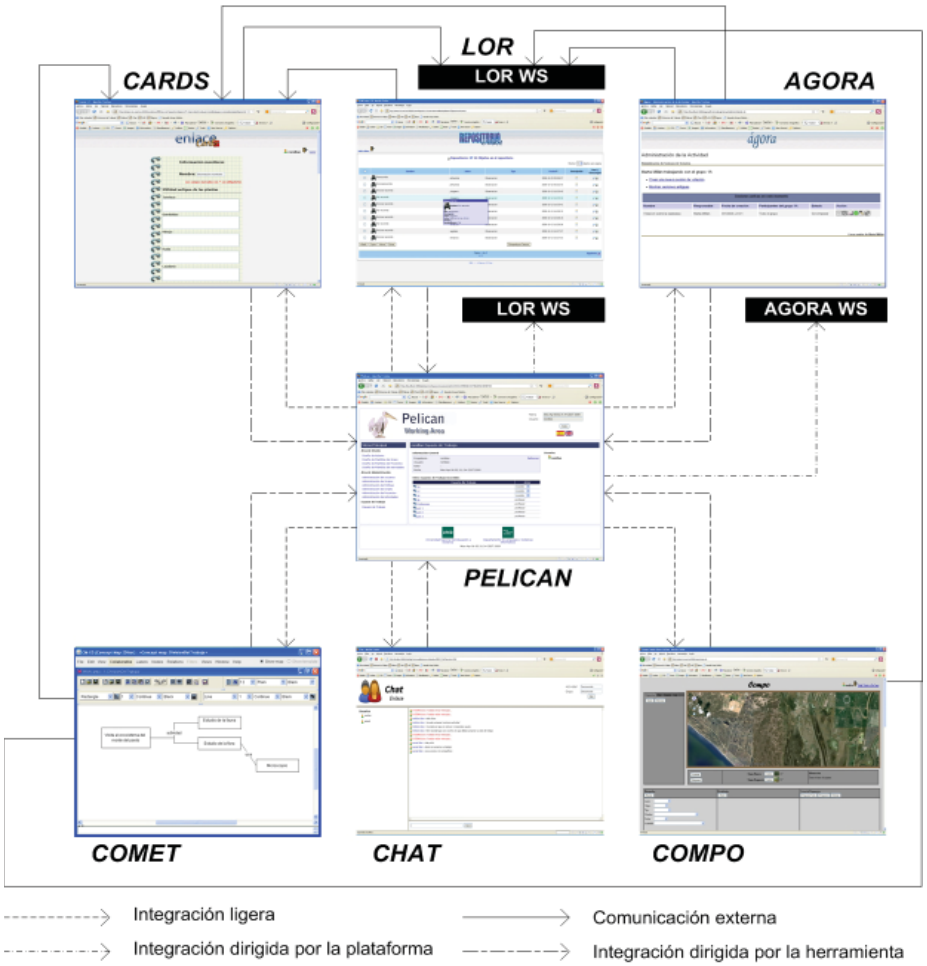

Figura 4.1. Ecosistema tecnológico del proyecto ENLACE.

En la figura 4.1 se muestra una representación esquemática de todas las herramientas que constituyen el ecosistema tecnológico del proyecto ENLACE, cómo éstas se integran en el entorno y cómo se comunican unas con otras. Como puede apreciarse, todas las herramientas son incorporados a los espacios de trabajo de la plataforma haciendo uso de integración ligera. Esto es, mediante la definición de servicios que proporcionen a las mismas un valor para los parámetros de invocación necesarios adaptados al contexto de uso. La integración de nivel B (dirigida por la plataforma) sólo puede ser llevada a cabo con las herramientas LOR y AGORA puesto que estas son las únicas que disponen de servicios Web para que otras herramientas, y en particular Pelican, las utilicen (consulte apéndice A). Por su parte, la integración de nivel C (dirigida por la herramienta) es alcanzada, en mayor o menor medida por todas las herramientas. En efecto, todas ellas, excepto CHAT, consultan el servicio social de Pelican para comprobar las credenciales de usuarios y otras como CARDS, AGORA y CHAT acceden al resto de información social y colaborativa para adaptar apropiadamente su comportamiento. Ademas se producen comunicaciones ajenas a Pelican de la herramienta CARDS, AGORA, COMET y COMPO con LOR a través de sus servicios Web y de LOR y AGORA con CARDS.

4.3. Una Librería de scripts de adaptación

Como discutimos en el capítulo anterior, los scripts de adaptación constituyen una potente herramienta para capturar los aspectos dinámicos que aparecen comúnmente vinculados a muchos escenarios de aprendizaje colaborativo. Sin embargo, a lo largo del desarrollo de nuestras experiencias de evaluación hemos descubierto que su especificación puede resultar algo compleja para diseñadores y administradores ya que se requieren profundos conocimientos de la arquitectura de la plataforma.

Precisamente para hacer frente a este problema, uno de nuestros esfuerzos de desarrollo en las pruebas ha sido la creación de una librería de scripts que proporcionase soluciones probadas para problemas de diseño que aparecen recurrentemente en la especificación de escenarios colaborativos. Los objetivos perseguidos con este desarrollo son los siguientes:

- **La librería como catálogo de scripts reutilizables.** Esta librería pretende poner a disposición de los diseñadores una familia de scripts para que puedan ser invocados directamente desde otros scripts haciendo uso de la tarea *callScript* (consulte la tabla 3.3). De esta manera, desde una perspectiva de caja negra, los scripts de la librería pueden ser considerados operaciones no atómicas de un mayor nivel de abstracción que las tareas pero dirigidos igualmente a tratar problemas frecuentes.

- **La librería como manual de referencia técnico.** La librería puede ser utilizada como un manual de referencia para permitir a los diseñadores conocer las técnicas y estrategias más apropiadas para la implantación de ciertos requerimientos de adaptación. En este sentido, los diseñadores pueden consultar el código fuente de estos scripts para inspirarse y tomar ideas acerca de cómo crear sus propios scripts.

- **La librería como repositorio abierto de scripts.** El proceso de construcción de scripts de adaptación puede ser considerado una labor incremental en la que unos scripts se construyen a partir de la invocación de otros mas sencillos. Desde este enfoque, las especificaciones de un escenario previo no deben ser eliminadas de la plataforma sino mantenidas para que otros diseñadores las aprovechen después. En este sentido, la librería de scripts que proporcionamos debe ser considerada un punto de partida para realizar scripts de adaptación de más alto nivel.

La librería de scripts que presentamos en esta sección se encuentra conceptualmente organizada en seis categorías cada una de las cuales incluye una familia de scripts de frecuente utilidad relacionados funcionalmente entre sí. A continuación describiremos cada una de estas categorías y, para cada una de ellas, comentaremos los scripts mas relevantes:

- **Scripts de adaptación social.** Los scripts de adaptación social llevan a cabo transformaciones sobre el modelo mantenido por el subsistema social de Pelican. Estos scripts se utilizan para gestionar grupos y admisiones de usuarios. En concreto,

podemos distinguir, dentro de esta familia, seis scripts de adaptación. A continuación los describimos brevemente:

- *Absorb Groups.* Este script toma como entrada una colección de grupos y un grupo base e incorpora los usuarios de los primeros en el segundo, respetando el actor que encarna cada uno.
- *Create Sequencial Groups.* Este script se utiliza para crear grupos de trabajo de forma automática. A partir de una colección de usuarios y un número, que indica cuántos grupos deben crearse, el algoritmo distribuye a cada usuario en un grupo de acuerdo a una estrategia secuencial que mantiene el tamaño de los mismos lo más homogéneo posible.
- *Create Random Groups.* Este scripts es similar al anterior con la diferencia de que aquí, la estrategia de asignación de usuarios a grupos es aleatoria. Se numeran los grupos y, para cada usuario, se sortea un número que indica el grupo al que éste será asignado. El sorteo es llevado a cabo procurando la homogeneidad en el tamaño de todos los grupos.
- *Fussion Groups.* El script de fusión es similar al de absorción. Sin embargo en este caso, todos los grupos responden a un mismo de plantilla de grupo. En el primer script puede decirse que una colección de grupos son absorbidos por otro grupo mientras que aquí varios grupos se fusionan para hacer uno mayor.
- *Jigsaw Groups.* Este script recibe una colección de n grupos formados por k miembros y crea otra colección de k grupos formado por n miembros. Para ello selecciona un miembro de cada grupo inicial y lo inserta en uno de los nuevos grupos creados procediendo iterativamente hasta que todos los usuarios hayan sido procesados, tal y como prescribe el método de jigsaw.
- *Swap Users.* Este script recibe dos grupos y sendos miembros de los mismos y los intercambia enrolando al usuario del primer grupo en el segundo y el del segundo en el primero.

- **Scripts de control del flujo de trabajo.** Una labor típica de administración relacionada con la fase de desarrollo del trabajo colaborativo de un escenario es la gestión del secuenciamiento en que deben ser presentadas las actividades en los espacios de trabajo para su realización por parte de los estudiantes. En este sentido, los scripts de esta categoría ofrece facilidades para llevar esto a cabo:
  - *Create Workflow.* El script de creación de flujos de trabajo recibe una especificación declarativa en XML que prescribe cómo deben ser secuenciadas temporalmente las actividades de un proyecto y las condiciones que deben satisfacerse para que las transiciones se hagan efectivas (véase por ejemplo subsección 4.5.2 y 4.5.4). Esta especificación distingue entre ejecución de actividades secuencial, alternativa, excluyente, paralela, condicional, iterativa y temporal. El script crea un objeto de clase *Workflow* que es almacenable en los contextos de usuario y permite realizar el secuenciamiento.

  - *Move Workflow.* Este sencillo script recibe como argumentos un motor de workflow y una actividad cuyo desarrollo acaba de ser finalizado. El algoritmo realiza las transiciones pertinentes sobre el motor lo que provoca que las actividades subsiguientes de la especificación interna del motor se activen y que la actividad recibida como parámetro se desactive. De esta manera, el motor de workflow pasa a un nuevo estado caracterizado por la colección de todas las actividades que ya estaban activas más aquéllas que se han activado en esta transición.

  - *Read Workflow.* Las especificaciones de secuenciamiento para los motores de workflow son un contenido candidato a ser almacenado en los repositorios de LOR como un objeto de aprendizaje. No en vano, estas especificaciones suelen ser frecuentemente reutilizadas. Este script recibe un identificador de objeto de LOR con una especificación de secuenciamiento ,y construye un motor de workflow invocando para ello al script *Create Workflow*.

- **Scripts de gestión de roles.** Otro de los aspectos importantes que es necesario gestionar durante el desarrollo de actividades de aprendizaje es la asignación de roles a los usuarios. En esta categoría existen cuatro scripts de adaptación definidos:

  - *Fixed Role Allocation.* Este script recibe como parámetros un rol, una actividad y una colección de usuarios y asigna automáticamente a todos los usuarios el dicho rol dentro de la actividad. Este script puede ser de utilidad para llevar a cabo la asignación inicial de roles.

  - *Random Role Allocation.* Este script recibe como parámetros una colección de usuarios y una actividad y realiza una asignación aleatoria de los roles a los usuarios. Esta es otra estrategia de asignación inicial frecuentemente utilizada.

  - *Role Swapping.* Este script recibe como parámetros una actividad y dos usuarios e intercambia sus roles asignando el rol del primero al segundo y recíprocamente.

  - *Round – Robin Step Role Allocation.* Este script recibe como parámetro una actividad y una lista de usuarios y altera la distribución de roles asignando a cada usuario el rol de su predecesor en la lista y al primero el rol del último.

- **Scripts de integración.** Los scripts de esta categoría tienen por objeto llevar a cabo ciertas operaciones sobre algunas de las diferentes herramientas del ecosistema tecnológico de ENLACE. Dada la gran cantidad de scripts que contiene esta categoría aquí nos limitaremos a describir los grandes tipos en los que éstos pueden agruparte indicando sus propósitos generales:

  - *Scripts de AGORA.* Este subconjunto de scripts de integración está formado por cuatro scripts que tienen por objeto crear y borrar sesiones de votación en AGORA y obtener el resultado y traza de los votos una vez finalizada las mismas mediante la interpretación de la ficha de acta. Los dos primeros scripts son fre-

cuentemente invocados en la especificación el modelo de intervención de un escenario para crear automáticamente sesiones de votación cuyas opciones candidatas sean un producto generado dinámicamente durante el transcurso de la experiencia. Los dos últimos scripts son utilizados para realizar transformaciones adaptativas, tales como la creaciones de nuevos grupos y actividades o la asignación de roles, en función de los resultados de las votaciones.

– *Scripts de CARDS.* En este subconjunto se incluyen una colección de scripts que han sido utilizados en los escenarios de ejemplo que presentaremos en la sección 4.5. Estos scripts se encargan de interpretar una familia de fichas generadas durante el desarrollo de la experiencia por los estudiantes y que contienen instrucciones precisas acerca de cómo se realizará la división del trabajo, la asignación de grupos o el reparto de responsabilidades colaborativas. Cada script procesa el contenido de una de estas fichas y adapta la plataforma para permitir el desarrollo de la experiencia.

– *Scripts de LOR.* Los scripts de adaptación de LOR permiten llevar a cabo tareas de carácter administrativo relacionadas con esta herramienta. En concreto, pueden destacarse scripts para la creación y borrado de repositorios, para el copiado, borrado y transferencia de objetos entre repositorios, para la descarga y salvaguarda de objetos y para la asignación de permisos de acceso a los mismos para los usuarios y grupos de la plataforma.

• **Scripts de monitorización.** Esta categoría contiene scripts de adaptación que permiten automatizar algunas tareas de control típicamente relacionadas con los procesos de monitorización de las experiencias colaborativas. Aunque aquí podrían incluirse una gran cantidad de algoritmos dado las numerosas estrategias de seguimiento que es posible aplicar, en este caso hemos querido proporcionar tan sólo un compendio de scripts de ejemplos prototípicos:

– *Reference Enabling.* Este script permite habilitar y deshabilitar el acceso a las referencias de una actividad colaborativa. Su uso resulta de utilidad como mecanismo de control ya que permite ir proporcionando progresivamente información según los estudiantes la van necesitando.

– *Role Fussion.* Una de las posibles causas por la que algunos miembros de un grupo se muestran poco participativos en una actividad colaborativa es que la división de responsabilidades en roles es demasiado especializada. Para esos casos una medida de reestructuración que puede llevarse a cabo es la fusión varios roles en uno sólo. Este script realiza esta labor de fusión.

– *Role Split.* La situación opuesta a la anterior se da cuando algunos usuarios de un grupo se ven saturados de trabajo mientras otros permanecen ociosos. El problema en este caso es que se ha realizado una división del trabajo desigual. La solución consiste en dividir el rol de los usuarios sobrecargados en varios roles y asignar cada nuevo rol a otros miembros con menos trabajo. Para llevar a cabo esta división de roles se utiliza este script.

- *Send Mail Advice.* Cuando un monitor – o el propio sistema a través de mecanismos de detección automáticos implantados a través del subsistema de intervención – observa que el desarrollo de una actividad requiere proporcionar algunas directrices de orientación a los estudiantes puede utilizar este script para mandarle un correo a todos o algunos de los miembros del grupo.

- *Send Message Advice.* Este script es similar al anterior en tanto que también es utilizado para orientar a los estudiantes, con la diferencia de que, en este caso, dicha orientación se comunica en forma de mensajes instantáneos que saltan en la pantalla del navegador del usuario.

- *Service Enabling.* De forma asimilar al primero de esta categoría, este script se utiliza para activar y desactivar selectivamente cada uno de los servicios que aparecen desplegados en las actividades colaborativas. Esto permite a los monitores, o al propio sistema, controlar la cantidad y tipo de interacciones colaborativas que se permiten entre los estudiantes.

• **Scripts de evaluación.** Los scripts de esta categoría permiten llevar a cabo ciertas tareas que aparecieron de forma recurrente a lo largo del desarrollo de las experiencias de aprendizaje. A continuación describimos brevemente la familia de scripts que entran dentro de esta categoría:

- *Approval Report.* Cuando el profesor evalúa una actividad, típicamente a través de la revisión de los productos generados y del informe de colaboración del monitor, y la aprueba, éste puede enviar un mensaje a todos los miembros del grupo informando de que pueden continuar con el desarrollo de la actividad. El propósito de este script es precisamente eso. Su ejecución puede automatizarse declarando una regla asociada al evento de cambio de estado del ciclo de vida de la actividad donde el estado final sea el de *aprobada*.

- *Rejection Report.* De manera similar al anterior, este script sirve para comunicar a los miembros del grupo que no han aprobado la actividad y que deben seguir trabajando en ella. Este script se vincula al evento de transición hacia el estado de *suspendida*.

- *Request For Approval.* Cuando los estudiantes terminan una actividad deben comunicarlo al profesor para que la corrija enviándole un mensaje. Este script hace esto de forma automática de manera que puede vincularse a la transición de la actividad *pendiente de revisión*.

- *Social Upgrading.* En cada nuevo año académico las generaciones de estudiantes promocionan al curso siguiente. Dado que en la comunidad de ENLACE el nombre de los grupos coincide con el de las clases (1A, 2B, etc.), es necesario realizar una tarea de promoción social que mueva a los estudiantes a la clase que le corresponde. Este script se encarga de realizar este trabajo.

- *Product Preservation Upgrading.* Este script invoca al script anterior y además salvaguarda los productos generados el curso anterior al repositorio general.

4.4. Experiencias en escenarios reales

El proyecto ENLACE ha constituido un marco idóneo para probar la plataforma en contextos pedagógicos reales. En este sentido, los esfuerzos de evaluación de esta parte han ido dirigidos a comprobar el comportamiento de Pelican – y de las herramientas integradas en ella – sometido a situaciones concurrentes con múltiples usuarios y en entornos tecnológico formados por dispositivos inalámbricos tales como teléfonos móviles de última generación, PDA, ordenadores portátiles y de sobremesa o pizarras táctiles inteligentes, todos ellos conectados entre sí a través una red Wi–fi.

En concreto, el proyecto nos ha permitido desarrollar una colección de escenarios reales que se han llevado a cabo a lo largo de sus cinco años de duración dentro de un contexto de educación formal con estudiantes de primer y segundo curso de bachillerato de Enseñanza Secundaria Obligatoria pertenecientes al instituto Diego Velázquez de Torrelodones (Madrid). El dominio de conocimiento de todos los escenarios es común y está condicionado por la finalidad de dicho proyecto. Esto es, explorar el diseño y desarrollo de entornos educativos innovadores que ofrezcan soporte inteligente para realizar un amplio abanico de actividades de aprendizaje en dominios relacionados con las ciencias de la naturaleza [ENLACE]. En concreto, este proyecto conjuga dos espacios para situar el aprendizaje: el aula y la naturaleza. Las experiencias en el aula han permitido incorporar el uso de la tecnología para dar soporte al desarrollo colaborativo de experimentos. Las experiencias en la naturaleza, por su parte, han consistido en visitas regulares a distintos ecosistemas naturales tales como el parque nacional de Doñana o El Monte de El Pardo, coordinadas por la organización [Seo–BirdLife]. En este caso, los típicos cuadernos de campo se han sustituido por el uso de dispositivos PDA para registrar in situ las observaciones en forma de objetos de aprendizaje que luego pudieran ser reutilizadas en subsiguientes actividades en el aula. A continuación describiremos brevemente todos estos escenarios detallando las actividades que lo conforman y sus objetivos generales, entre los que destaca, como denominador común, introducir a los estudiantes en la aplicación del método científico como procedimiento para abordar la resolución de cualquier tipo de problema [Sharan & Sharan, 1994]:

- **Toma de contacto con la tecnología.** El escenario de toma de contacto con la tecnología es desarrollado a principio de curso por cada nuevo grupo de estudiantes que comienzan con el desarrollo de este tipo de experiencias formativas y tiene por objeto servir de primera toma de contacto con las herramientas y dispositivos que deberán utilizarán a lo largo del curso. Consta de dos actividades:

  – *Cuestionario del alumno.* En esta actividad, cada alumno debe rellenar, individualmente, tres cuestionarios acerca del instituto, sus hábitos de estudio, sus preferencias musicales, deportivas, etc. Para ello, se hace uso de servicios de CARDS y LOR.

  – *Elección de delegado.* Esta actividad consiste en elegir el delegado y el subdelegado de la clase utilizando las PDA y la pizarra electrónica. El profesor

controla, a través de AGORA, la sesión de votación que permite a los alumnos presentar sus candidaturas y votar a sus candidatos desde su PDA, mientras los resultados son visualizados en la pizarra común.

- **El mundo de la física.** Este escenario recoge una colección de actividades colaborativas cuyo objetivo es utilizar el procedimiento de enseñanza por descubrimiento para distintas leyes de la física, según el cual los estudiantes aprenden a partir de la experimentación en situaciones reales y simuladas. En concreto, el escenario incluye las siguientes actividades:

  – *Construcción de un modelo sobre la densidad.* En esta actividad se pretende que los estudiantes redescubran las leyes que rigen el calculo de la densidad de un compuesto líquido. Para ello, realizan una serie de experimentos en el laboratorio, rellenan un test, y elaboran una hipótesis de la relación existente entre las medidas de masa y volumen a partir de sus observaciones. Todo esto queda soportado por el uso de la herramienta CARDS.

  – *Construcción de un modelo sobre flujo y caudal.* En esta actividad, los estudiantes deben elaborar un modelo sobre el flujo y caudal de un río a partir de una serie de medidas tomadas en la naturaleza. La recogida de datos es realizada con la herramienta CARDS usando dispositivos PDA.

  – *Construcción de un modelo sobre la ley de Ohm.* En esta actividad los estudiantes construyen en el laboratorio diferentes circuitos de corriente continua con resistencias, baterías y bombillas montadas en serie y en paralelo. Con el uso de un multímetro, miden las tensiones e intensidades de corrientes y las registran en fichas de CARDS con el uso de PDA. Después, a partir de sus observaciones deben inferir la relación existente entre ambas magnitudes.

  – *Construcción de un modelo para el cálculo de la eclíptica del sol.* En esta actividad los estudiantes realizan medidas de la longitud de la sombra proyectada por un árbol a diferentes horas del día. A partir de estos datos deben elaborar un modelo de la eclíptica solar que les permita conocer la altura del árbol. Para el registro de datos se utiliza la herramienta CARDS desde los dispositivos PDA entregados a los estudiantes.

- **Visita al monte de El Pardo.** Este escenario contiene una colección de actividades relacionadas con las visitas a los ecosistemas de El Monte de El Pardo. En concreto, dentro de este entorno natural se distinguen dos ecosistemas: la ribera y el encinar. El desarrollo de la experiencia consiste en dividir a la clase en varios grupos y asignar a cada uno de ellos un ecosistema. Cada grupo realiza un itinerario distinto donde recogen datos sobre observaciones realizadas a lo largo del mismo. Todo esto se articula a partir de las siguientes actividades:

  – *Identificación del canto de las aves.* En esta primera actividad, realizada en el aula, el profesor presenta la aves que están presenten en los ecosistemas de El Pardo y se hace una audición de su trino para aprender a identificarlos.

Los alumnos escuchan las audiciones y discuten sobre el ave que la emite. Se utiliza la herramienta AGORA para alcanzar un consenso en la identificación.

– *Visita a El Pardo.* En esta actividad cada grupo realiza un itinerario y va recogiendo datos sobre diferentes elementos encontrados a lo largo del mismo tales como información sobre el hábitat, cantos de aves en las estaciones de escucha, observaciones sobre la fauna y la flora, huellas y rastros, etc. Todo ello es soportado por la herramienta CARDS a través del uso de PDA. Cada ficha incluye información del posicionamiento geográfico para poder ubicar sobre el itinerario cada registro de datos.

– *Construcción de modelos comunes.* De nuevo en el aula, los estudiantes deben construir modelos que describan diferentes aspectos el ecosistema que han visitado: modelo de flora y fauna, modelo topográfico, modelo de la cadena trófica, modelo de claves dicotómicas, etc.

– *Construcción de modelos específicos.* En esta actividad los miembros de cada grupo eligen entre uno de los siguiente modelos para construir colaborativamente: modelo de plantas aromáticas y medicinales, modelo de simbiosis y parasitismo, modelo de paisaje, etc.

– *Construcción de un modelo general.* Haciendo uso de la herramienta COMPO y de los recursos almacenados en LOR en las actividades anteriores, los estudiantes deben componer una representación digital del itinerario, ubicando en cada punto del mismo la ficha que fue construida. Los datos de posicionamiento geográfico de las fichas asisten en esta labor.

Elaboración de un modelo para la visita a el Pardo. En las actividades anteriores del escenario los estudiantes fueron dirigidos por el diseño instruccional en la realización de una serie de actividades tales como la recolección de datos de las observaciones encontradas en el itinerario, la construcción de modelos para esos ecosistemas, la ejercitación de la identificación del trino de las aves en el aula para poder reconocerlos en las distintas estaciones de escucha preparadas, etc. En esta actividad por el contrario se pretende que sean los propios estudiantes los que elaboren su propio plan de visita, determinando colaborativamente qué actividades deberían realizarse y que datos deberían tomarse. Para ello diseñan en CARDS las plantillas de fichas necesarias indicando la estructura interna de los campos de datos de que estará formada cada una. Para articular a esta actividad, los estudiantes disponen de la herramienta AGORA para dar soporte a la toma de decisiones colaborativa, la herramienta CHAT para poder discutir, la herramienta COMET para definir formalmente el plan de la visita y el repositorio de objetos de aprendizaje LOR para poder almacenar las plantillas de fichas diseñadas.

- **Visitas a museos.** El escenario de visitas a museos incluye todas las actividades relacionadas a distintos museos y actividades extraescolares. En concreto aquí se incluyen las dos siguientes actividades:

– *Visita al museo Cosmoxaixa.* En esta actividad los estudiantes utilizan sus dispositivos PDA para rellenar una colección de fichas de CARDS relacionadas con cada uno de los puntos de interés del itinerario del museo: electricidad, luz, mecánica, presión, etc. Después con la herramienta COMET elaboran un mapa conceptual para relacionar los conceptos que han aprendido.

– *Participación en concurso de investigación.* Esta actividad fue planificada a petición de un grupo de estudiantes que querían presentarse al concurso de investigación *Es de Libro* [EsDeLibro]. En ella se abordan temas relacionados con la radiación solar y sus efectos en el ser humano. Para dar soporte a su desarrolló se proporcionaron diversas fichas de CARDS para que rellenaran los alumnos así como servicios de AGORA, LOR y COMET.

4.5. Cinco escenarios colaborativos de ejemplo

A lo largo de esta sección presentaremos cinco ejemplos de escenarios colaborativos complejos que han sido desarrollados para poner a prueba las capacidades de la plataforma Pelican. Para cada uno de ellos proporcionaremos una breve descripción, el contexto donde se aplica, su dominio de conocimiento y las fases de desarrollo que lo constituyen. Además discutiremos cómo puede ser llevada a cabo su implantación en Pelican deteniéndonos en el diseño de cada uno de los sus cuatro modelos constituyentes (de sociedad, de colaboración, de integración y de intervención).

4.5.1. Procedimiento Jigsaw

Las capacidades de la plataforma, y en particular las del sistema de intervención, permiten transformar progresivamente la organización social de una comunidad virtual de aprendizaje durante el desarrollo de las experiencias. Para ilustrar esto, hemos escogido el conocido ejemplo de Jigsaw que constituye un método pedagógico estándar dentro del aprendizaje colaborativo [Slavin, 1991]. A continuación pasamos a describir la versión que hemos hecho del mismo:

- **Descripción.** En el escenario de Jigsaw, el profesor comienza la experiencia presentando a los estudiantes la materia en estudio. Una vez establecida una primera toma de contacto con ella, éstos deberán proponer una colección de temas en los que puede ser descompuesta la misma. Una vez hecho esto, cada estudiante escoge aquel tema que le resulte de mayor interés personal. Respetando esta elección en la medida de lo posible, se creará un grupo de expertos para cada tema con el fin de que sus miembros profundicen en su estudio. Después, se forman equipos de trabajo constituidos por un miembro de cada grupo experto y éstos pasarán a realizar una actividad colaborativa. La especialización de conocimiento adquirida por este procedimiento garantiza la colaboración entre los estudiantes en el desarrollo de esta actividad.

- **Contexto.** La puesta en práctica de este escenario se realiza sobre una clase de k x n estudiantes. El primer proceso de estratificación divide a la clase en k grupos expertos formados por n miembros cada uno. El segundo, redistribuye a los

estudiantes de estos grupos de acuerdo al procedimiento Jigsaw para obtener un total de n equipos de trabajo formados por k miembros cada uno. En el ejemplo que proporcionamos en este escenario las actividades están pensadas para ser desarrolladas por estudiantes de educación formal de enseñanza secundaria de edades comprendidas entre los 15 y 16 años implicados en una asignatura de introducción a la historia del mundo.

- **Dominio.** En el escenario que presentamos en este apartado, hemos escogido como dominio de conocimiento el nazismo y la segunda guerra mundial. Por tanto, todas nuestras explicaciones y ejemplos quedarán circunscritos a temas relacionados con este episodio de la historia. No obstante, todo lo aquí expuesto es fácilmente extrapolable a otros dominios diferentes.

- **Fases de desarrollo.** El desarrollo de este escenario puede ser dividido en una serie o fases que se desarrollan de forma secuencial. A continuación describimos brevemente cada una de estas fases:

  – *Propuesta de temas expertos.* Cada estudiante propone uno o varios temas para ponernos en consideración con el resto de compañeros de la clase de k x n estudiantes.

  – *Elección de temas expertos.* Una vez recibidas todas las propuestas de temas expertos los estudiantes votan su tema favorito. Los k temas más votados serán seleccionados como temas expertos.

  – *Formación de grupos expertos.* Los estudiantes vuelven a votar para expresar su interés por uno de los temas seleccionados en la fase anterior. El resultado de esta votación es utilizado para formar k grupos expertos constituidos por n estudiantes cada uno.

  – *Desarrollo de la actividad experta.* Cada grupo experto desarrolla una actividad relacionada con el tema cuyo estudio se le ha asignado. El propósito de la misma es conseguir que los estudiantes adquieran un conocimiento profundo de los aspectos relacionados con dicho tema.

  – *Formación de equipos de trabajo.* Tras el desarrollo de la actividad experta se forman n equipos de trabajo constituidos por k miembros cada uno. La técnica de estratificación para llevar esto a cabo consiste en seleccionar a un miembro de cada grupo experto para formar cada equipo, de manera que se garantice que cada uno de ellos estará constituido por un experto en cada tema.

  – *Desarrollo de la actividad de Jigsaw.* Cada equipo de trabajo aborda el desarrollo de una misma actividad colaborativa que llamaremos actividad de Jigsaw. Los conocimientos adquiridos por cada miembro durante el desarrollo de las actividades expertas capacitan a los estudiantes para llevarla a cabo.

La figura 4.2 ilustra las fases de desarrollo del escenario de Jigsaw que estamos describiendo en esta sección. Como puede verse, en las dos primeras cada estudiante

envía propuestas de temas y vota sus favoritas. En la fase III y IV se escogen las 4 más votadas para crear sendos grupos expertos formados por 3 miembros cada uno – en total hay 12 estudiantes en el aula – y se llevan a cabo las actividades expertas. En las dos últimas fases se reorganiza a los estudiantes para formar 3 equipos de trabajo de 4 miembros cada uno y se realizan las actividades de Jigsaw. En la figura se ha numerado a cada estudiante para poderlos identificar en cada fase. Comparando los colectivos de las fases III y V puede apreciarse que el procedimiento de estratificación es aquél prescrito por este método pedagógico.

Una vez presentados los objetivos y una breve descripción del escenario de Jigsaw, estamos en disposición de discutir cómo éste puede ser implantado en Pelican. En los siguientes apartados de esta subsección describiremos en detalle el diseño de los modelos de sociedad, colaboración, integración e intervención para que den soporte al mismo.

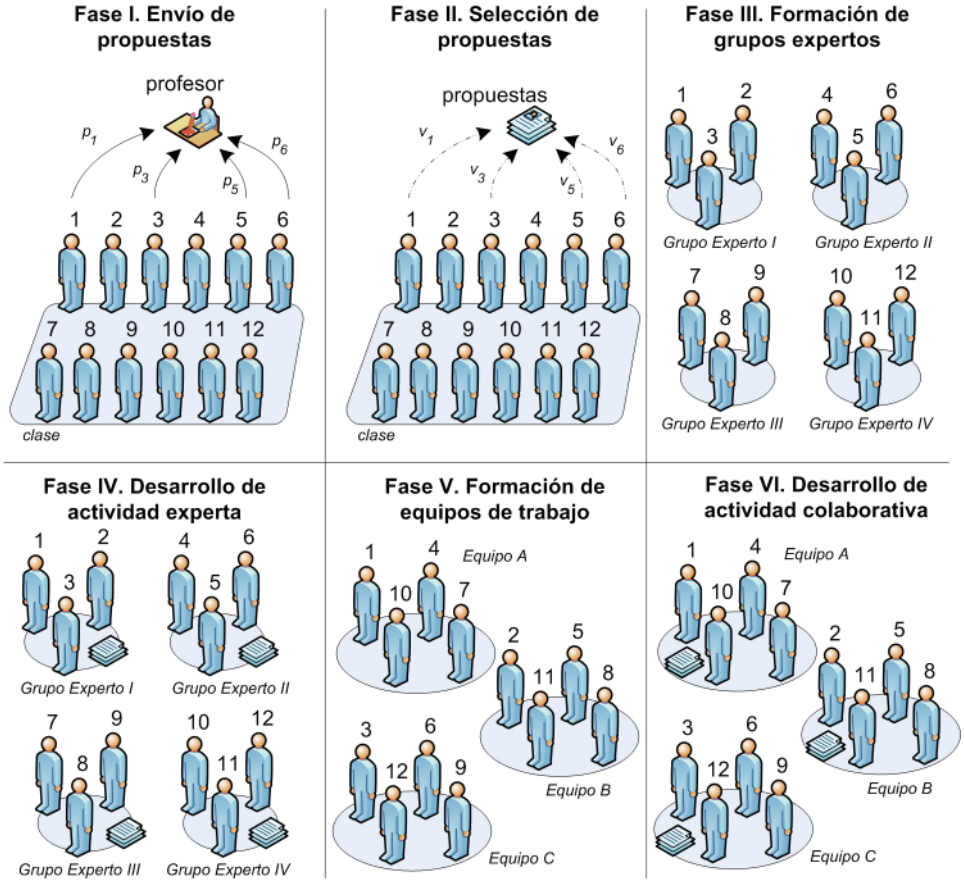

Figura 4.2. Fases del escenario de Jigsaw.

Diseño del modelo de sociedad

El modelo de sociedad que es necesario construir para poner en práctica este escenario colaborativo incluye una serie de actores y plantillas de grupo que pasamos a describir brevemente a continuación:

- **Actores.** Sin perdida de generalidad y en aras a la simplicidad podemos afirmar que en este escenario tan sólo se requiere contar con la participación de dos actores principales. La aparición de otros actores, como la de aquellos descritos en el capítulo 2 de esta tesis, resulta tangencial y su participación no se ha incluido en esta descripción:
  - *Profesor.* El profesor es el encargado de presentar la experiencia de aprendizaje colaborativo a los estudiantes y en este caso hará labores de administración llevando a cabo tareas de despliegue de actividades, control del ciclo de vida de las mismas y gestión del flujo de trabajo. Bien es cierto, que muchas de estas tareas pueden automatizarse en Pelican pero aquí pretendemos centrar nuestra discusión en la dimensión social de la experiencia. Es decir, en ilustrar cómo el entramado de grupos expertos y equipos de trabajo se va modificando dinámicamente según avanza el flujo de trabajo instruccional.
  - *Estudiante.* El estudiante es el encargado de llevar a cabo las actividades de aprendizaje definidas durante la experiencia. Estas no solamente incluyen las actividades colaborativas descritas en las fases IV y VI sino, adicionalmente todas aquellas necesarias para llevar acabo la formación de grupos expertos y equipos de trabajo (propuesta y votación de temas y selección del grupo experto al que desea pertenecer cada estudiante).
- **Plantillas de grupo.** En el desarrollo de este escenario entran en juego tres plantillas de grupo que se alinean con sendos niveles de estratificación social. A continuación describimos brevemente el objetivo y estructura interna de las mismas:
  - *Clase.* Dado que estamos en un contexto de educación formal, supondremos que esta experiencia se desarrolla con los estudiantes de un determinado aula. La clase representa este colectivo y está constituido por un profesor y k x n estudiantes. Además la clase incluye dos plantillas de grupo hijas: el grupo experto, con una cardinalidad mínima y máxima de k y el equipo de trabajo con una cardinalidad mínima y máxima de n.
  - *Grupo experto.* El grupo experto representa al colectivo de estudiantes que profundiza en el estudio de uno de los temas seleccionados. Dado que existen k equipos, el tamaño de estos grupos es de n miembros. Aquí es posible incluir al profesor aunque su participación no es estrictamente necesaria.
  - *Equipo de trabajo.* Cada equipo de trabajo está formado exactamente por k estudiantes que son los encargados de llevar a cabo la actividad de Jigsaw descrita en la fase VI. Dentro de estos grupos también sería posible incluir al profesor pero por simplicidad nosotros no lo hemos hecho.

Una vez definidos estos cinco elementos, el modelo de sociedad ya está preparado para dar soporte al escenario. Al comienzo de la fase de desarrollo, los administradores deben seleccionar la clase sobre la que van a desplegar la experiencia. Sin embargo, la formación de los grupos expertos y los equipos de trabajo no puede ser

llevada a cabo en este momento ya que aún no se conocen los miembros que los constituyen. Por eso, esta es una tarea que será realizada dinámicamente haciendo uso de las prestaciones del subsistema de intervención, tal y como discutiremos más adelante. En efecto, hasta que no se alcance la fase III, donde los estudiantes votan en qué tema desean especializarse, no es posible crear los grupos expertos y, como los equipos son formados aleatoriamente a partir de dichos grupos, éstos tampoco pueden ser constituidos.

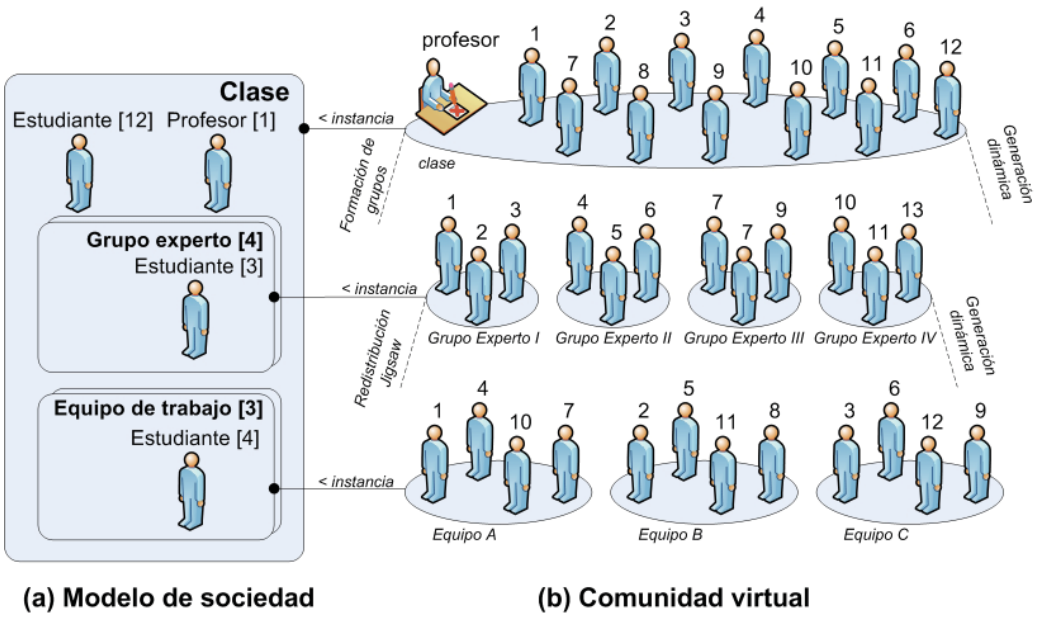

Figura 4.3. Modelo de sociedad y comunidad del escenario de Jigsaw.

La figura 4.3 (a) muestra el modelo de sociedad necesario para este escenario. En este ejemplo, la clase está formada por un profesor y doce estudiantes así como por cuatro grupos expertos de tres estudiantes cada uno y tres equipos de trabajo de cuatro estudiantes cada uno. En la parte (b) de la figura puede apreciarse cómo cada plantilla representa un nivel social diferente. La clase se descompone en los grupos expertos I (estudiantes 1 a 3), II (estudiantes 4 a 6), III (estudiantes 7 a 9), y IV (estudiantes 10 a 12). Después, a partir del proceso de redistribución prescrito por este método, se forman los equipos de trabajo A (estudiantes 1,4, 7 y 10), B (estudiantes 2, 5, 8 y 11) y C (estudiantes 2,6, 9 y 12) encargados de llevar acabo la actividad de Jigsaw.

Diseño del modelo de colaboración

El diseño colaborativo que debe ser llevado a cabo para la especificación del escenario de Jigsaw requiere únicamente de la definición de una plantilla de proyecto que incluya un total de cinco plantillas de actividad:

- **Presentación de la experiencia.** Esta plantilla tiene como propósito presentar a los estudiantes el tema de la segunda guerra mundial. Para ello, incluye una colección de referencia a recursos en la Web sobre diferentes temas relacionados con el nazismo, la vida de Hitler, la persecución antisemita, el contexto geopolítico etc.

- **Elección de temas expertos.** En esta actividad los estudiantes proponen temas expertos relacionados con la segunda guerra mundial y determinan, a través de una votación, cuáles serán aquéllos objeto de estudio durante la experiencia.
- **Formación de grupos expertos.** En esta actividad cada estudiante expresa su interés por participar en un grupo experto a través de una votación. Una vez finalizada la misma, el sistema interpretará los resultados y formará los grupos expertos respetando, en la medida de lo posible estas elecciones.
- **Actividad experta.** Las actividades expertas son aquellas que desarrollan los grupos expertos para adquirir un conocimiento profundo sobre el tema de estudio. Existen dos aproximaciones para dar soporte a esta fase del escenario. Es posible definir manualmente una plantilla de actividad especifica para cada tema experto una vez conocidos cuáles son éstos o, alternativamente, crear una plantilla genérica común para todos los grupos expertos que incluya referencias y servicios de utilidad tales como motores de búsqueda, enciclopedias y diccionarios en línea. Nosotros asumiremos esta segunda aproximación más sencilla que resulta suficiente para nuestras pretensiones ilustrativas.
- **Actividad de Jigsaw.** La plantilla de Jigsaw es aquélla que se realiza en cada equipo de trabajo e implica a un experto de cada tema. Finalizada esta actividad la experiencia ha terminado y el proyecto se puede considerar cerrado.

Durante la fase de desarrollo, todas estas actividades serán llevadas a cabo de manera secuencial tal y como se muestra en la figura 4.4. La responsabilidad de mantener este orden recae, en este escenario, sobre el profesor que las activará y desactivará manualmente. Tal y como dijimos anteriormente esta gestión no se ha automatizado con reglas y scripts de adaptación para mantener sencillo el escenario. Además, la realización de cada una de las actividades compete a un nivel social distinto, lo cual ha de ser tenido en cuenta durante el despliegue, ya que cada una debe aparecer sobre el espacio de trabajo apropiado. En efecto, como se ve en la figura, las tres primeras actividades son realizadas a niveles de clase, la cuarta a nivel de grupo experto y la quinta a nivel de equipo de trabajo.

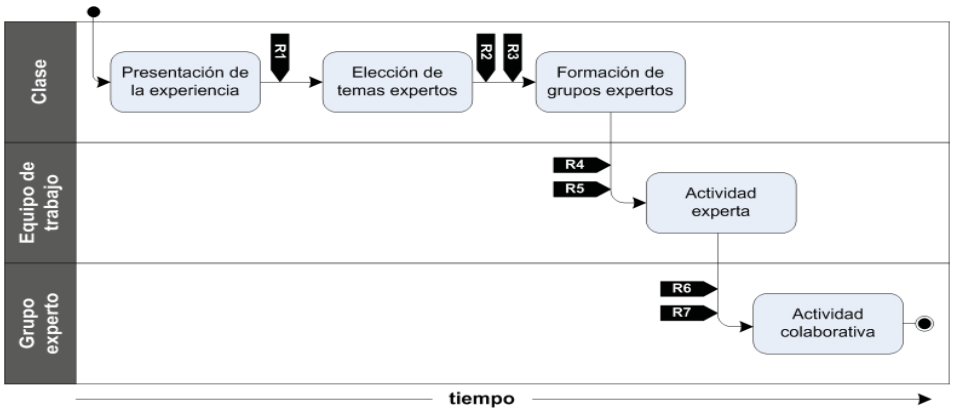

Figura 4.4. Flujo de trabajo instruccional del escenario de Jigsaw.

Diseño del modelo de integración

En este apartado presentamos todos los servicios proporcionados por las herramientas del ecosistema tecnológico de ENLACE que son necesarios para llevar acabo el desarrollo de esta experiencia. Para cada herramienta, describiremos en qué actividades se incluyen servicios de la misma y cual es su propósito en relación a ella. La discusión sobre los aspectos dinámicos de la integración se postergará al siguiente apartado donde abordaremos el modelo de intervención de este escenario:

- **CHAT.** Para dar soporte a la interacción entre los miembros de la clase, de cada equipo y grupo se proporcionan servicios de CHAT en todas las actividades.
- **AGORA.** En la actividad elección de temas expertos y formación de grupos expertos se incluyen sendos servicios de AGORA para permitir a los estudiantes votar los temas expertos e indicar al sistema a qué grupo experto desean pertenecer.
- **LOR.** Para proporcionar recursos a los estudiantes en la actividad de presentación se proporciona un servicio que da acceso al repositorio de clase. Asimismo la actividad experta y la de Jigsaw contienen ambas sendos servicios para acceder al repositorio de grupo experto y equipo de trabajo, respectivamente.
- **CARDS.** En la actividad final de Jigsaw el trabajo consiste en la construcción de unas fichas de CARDS cuyas plantillas cubren diversos aspectos relacionados con la segunda guerra mundial y el nazismo.

Diseño del modelo de intervención

El modelo de intervención de este escenario se centra en dos aspectos fundamentales. En primer lugar, proporcionar las reglas necesarias para construir dinámicamente los grupos expertos y los equipos de trabajo una vez recogidos los resultados de la votación de formación de grupos. En segundo lugar, articular los mecanismos de integración oportunos con AGORA para crear las sesiones de votación que permitan recoger las decisiones de los estudiantes. Muchas de las reglas que se ilustran en este ejemplo y en los de las subsecciones siguientes han sido expresados por simplicidad con valores literales en su código para hacer referencia al nombre de actores, grupos, actividades, etc. Debe recordarse, no obstante, que Pelican permite parametrizar las reglas y scripts de adaptación para aumentar el nivel de reutilización de la especificación. A continuación presentamos en detalle todas las reglas en el orden que se van disparando a lo largo del flujo de trabajo instruccional (ver figura 4.4):

- **R1 Creación de la sesión de votación de temas expertos.** Cuando la actividad de presentación ha finalizado el profesor transita a la actividad de elección de temas expertos que, como dijimos en el apartado anterior incluye una sesión de votación para permitir a los estudiantes proponer temas candidatos y votarlos. Esta regla crea esa sesión de votación. Como se ve en el listado 4.1, la regla está asociada al evento de cambio de estado y el disparador indica que la ejecución sólo se producirá si se transita al estado de *pendiente*. El script invoca a otro script de la librería para crear la sesión de AGORA.

Listado 4.1. Regla de creación de la sesión de votación de temas expertos.

```
Evento: pelican.workspace.activity.updateState
Disparador: event.newActivity.activityTemplate.name=='Presentacion' &&
            event.newActivity.state=='PENDING'
Script:
 def group = event.newActivity.getProject().getWorkspace().getOwner()
 def teacherBinding = group.getActorBinding('profesor')
 def teacher = teacherBinding.getUsers() [0]
 common.callScript ('INT.AG.Create Session',
                    ['name'       : 'Eleccion de temas Jigsaw',
                     'moderator'  : teacher,
                     'group'      : group,
                     'type'       : 'TEXT',
                     'candidates' : [],'mode': 'YES'])
```

- **R2 Creación de grupos expertos.** Al final de la actividad de elección de temas expertos, la votación ha concluido y la ficha CARDS del acta de la misma se ha almacenado en el repositorio. El contenido de esta ficha, que indica cuáles han sido los k temas más votados es interpretado por esta segunda regla para crear un grupo experto por cada tema seleccionado al que le asigna su nombre. La regla que es mostrada en el listado 4.2, se asocia al cambio de estado y es ejecutada cuando el estado pasa a ser *pendiente*. El script recoge, en primer lugar, el voto de cada estudiante mediante un script de nuestra librería (❶). Después contabiliza los mismos (❷) y los ordena decrecientemente en una lista (❸). Finalmente toma esta lista y recorre sus k primeros elementos, creando para cada uno de ellos un grupo. La construcción de estos grupos utiliza la plantilla de grupo experto. Para llevar esto a cabo el script llama a la tarea *createGroup* (❹).

Listado 4.2. Regla de creación de grupos expertos.

```
Evento: pelican.workspace.activity.updateState
Disparador: event.newActivity.activityTemplate.name=='Eleccion de temas' &&
            event.newActivity.state=='PENDING'
Prioridad: 1
Script:
 def voteMap = common.callScript ('INT.AG.Get Session Votes', ❶
                                  ['name' : 'Eleccion de temas Jigsaw'])
 def votes = [:] ❷
 def users = voteMap.keySet ()
 for (aUser in users) {
   def aVote = voteMap [aUser].text
   if (votes.keySet().contains(aVote)) votes[aVote]++
   else votes[aVote] = 1
 }
 def options = [] ❸
 options.addAll (votes.keySet ())
 for (def i = 0; i < options.size (); i++)
```

```
  for (def j = 0; j < options.size (); j++)
    if (votes[options[i]] < votes[options[j]]) {
      def aux = options[i]
      options[i] = options[j]
      options[j] = aux
    }
def expertGroups = [] ❹
def groupTemplate = social.getGroupTemplate ('grupo experto')
def parentGroup = event.newAactivity.getProject().getWorkspace().getOwner()
for (def i = 0; i < nExpertGroups; i++) {
  def aGroup = social.createGroup (options[i], options[i],
                                   groupTemplate,
                                   parentGroup)
  expertGroups.add (aGroup)
}
```

- **R3 Creación de la sesión de votación de grupos expertos.** Antes de empezar la actividad de formación de grupos, es necesario crear una nueva sesión de votación en AGORA para ella, mediante la que cada estudiante pueda indicar a cuál de los k grupos desea pertenecer. Esta regla coge el nombre de los grupos generados con R2 y los pone como opciones candidatas de la votación. La regla (listado 4.3) es asociada igualmente al evento de cambio de estado de la actividad y las condiciones de disparo son las mismas que en R2. Sin embargo, la prioridad de esta regla indica que ésta debe ejecutarse después de R2 tal y como se muestra en la figura 4.4. El script asociado comienza recolectando el nombre de los grupos expertos (❶). Después invoca a un script de la librería para crear una nueva sesión de votación donde las opciones son estos nombres (❷).

Listado 4.3. Regla de creación de la sesión de votación de grupos expertos.

```
Evento: pelican.workspace.activity.updateState
Disparador: event.newActivity.activityTemplate.name=='Eleccion de temas' &&
            event.newActivity.state=='PENDING'
Prioridad: 2
Script:
 def group = event.newActivity.getProject().getWorkspace().getOwner()
 def teacherBinding = group.getActorBinding('profesor')
 def teacher = teacherBinding.getUsers () [0]
 def expertGroupTemplate = social.getGroupTemplate ('grupo experto')
 def childGroups = group.getSubgroups (expertGroupTemplate)
 def options = childGroups.collect() {it -> it.name} ❶
 common.callScript ('INT.AG.Create Session',
                    ['name'      : 'Eleccion de grupo experto',
                     'moderator' : teacher,
                     'group'     : group,
                     'type'      : 'TEXT',
                     'candidates' : options, ❷
                     'mode'      : 'NO'])
```

- **R4 Enrolado de estudiantes en grupos expertos.** Una vez finalizada la votación de formación de grupos, el sistema recoge, a través de esta regla, el acta de la misma e enrola a cada estudiante en el grupo experto que ha elegido. Este proceso se realiza respetando en la medida de lo posible la voluntad de los estudiantes pero garantizando ante todo que cada grupo experto esté formado por n estudiantes. La regla que lleva a cabo este proceso es ilustrada en el listado 4.4. Como puede verse, el evento y las condiciones de disparo prescriben que se ejecutará cuando el estado de la actividad de formación de grupos pase a *pendiente*. El script comienza recuperando el voto de cada estudiante (❶) y lo procesa para agrupar a los estudiantes en función del grupo experto que hayan elegido(❷). Después, aplica una corrección para garantizar que cada colectivo esté formado exactamente por n estudiantes (❸). A partir de este agrupamiento, que es temporalmente almacenado en la variable *enrrollment*, el script incluye a cada estudiante en el grupo experto que ha seleccionado (❹).

Listado 4.4. Regla de enrolado de estudiantes en grupos expertos.

```
Evento: pelican.workspace.activity.updateState
Disparador: event.newActivity.activityTemplate.name=='Eleccion grupos' &&
            event.newActivity.state=='PENDING'
Prioridad: 1
Script:
 def votes = common.callScript ('INT.AG.Get Session Votes', ❶
                                ['name' : 'Eleccion de grupo experto'])
 def users = votes.keySet () ❷
 def groups = votes.values ()
 def enrollment = [:]
 groups.collect () {enrollment[it.text] = []}
 for (user in users) { ❸
   def group = votes [user].text
   if (enrollment[group].size() >= maxSize)
     for (aGroup in enrollment.keySet ())
       if (enrollment[aGroup].size() < maxSize) {
         group = aGroup
         break
       }
   enrollment[group].add (user)
 }
 def expertGroup = event.newActivity.getProject().getWorkspace().getOwner()
 def teacherBinding = expertGroup.getActorBinding('profesor')
 def teacher = teacherBinding.getUsers () [0]
 def student = social.getActor ('estudiante')
 for (aGroup in enrollment.keySet()) { ❹
   def realGroup = social.getGroup (aGroup)
   for (aUser in enrollment [aGroup]) {
     def realUser = social.getUser (aUser)
```

```
      social.bindUserToActor (realUser, student, realGroup)
    }
    social.bindUserToActor (teacher, teacherBinding.actor, realGroup)
  }
```

- **R5 Despliegue de actividades expertas.** Tras la formación de los grupos expertos la plantilla de actividad experta es desplegada en sus espacios de trabajo. La regla que se encarga de llevar a cabo esta labor se muestra en el listado 4.5. En ella, puede apreciarse como tanto el evento al que está asociado la misma como las condiciones de disparo coinciden con los de la regla R4. Por tanto la ejecución de ambas reglas coincide en el tiempo. Sin embargo la prioridad de cada una de ellas prescribe que R5 sigue a R4. El script asociado a la regla, consulta al subsistema social para obtener la colección de grupos expertos que existen definidos en la comunidad (❶). Después consulta al subsistema de colaboración para obtener la plantilla de proyecto y actividad de la actividad experta (❷). Y finalmente, crea un proyecto y actividad para cada grupo experto obtenido (❸).

Listado 4.5. Regla de despliegue de actividades expertas.

```
Evento: pelican.workspace.activity.updateState
Disparador: event.newActivity.activityTemplate.name=='Eleccion grupos' &&
            event.newActivity.state=='PENDING'
Prioridad: 2
Script:
 def parentGroup = event.newActivity.getProject().getWorkspace().getOwner() ❶
 def expertGroupTemplate = social.getGroupTemplate ('grupo experto')
 def expertGroups = parentGroup.getSubgroups (expertGroupTemplate)
 def expertPTemplate = collaboration.getProjectTemplate ('Jigsaw') ❷
 def expertATemplate = collaboration.getActivityTemplate ('Actividad experta')
 for (expertGroup in expertGroups) { ❸
   def expertProject = collaboration.createProject (expertPTemplate, expertGroup)
    def expertActivity = collaboration.createActivity (expertATemplate)
    collaboration.addActivity (expertActivity, expertProject)
 }
```

- **R6 Creación de equipos de trabajo.** Finalizada la actividad experta en todos los grupos se deben crear los equipos de trabajo. Para automatizar esta labor se utiliza la regla R6 que aparece representada en el listado 4.6. El evento al que está vinculado esta regla es el de cambio de estado de la actividad. De acuerdo a lo expresado en el disparador, para ejecutarla, el profesor debe cambiar el estado de la actividad de elección de grupos de *pendiente* a *aprobada*. El script de adaptación de esta regla extrae del subsistema social la plantilla de equipo de trabajo y de grupo experto así como la colección de grupos que se ajustan a esta última plantilla (❶). Esta información es utilizada como argumento en la llamada al script de librería *Jigsaw Groups* (❷). Como describimos en la sección 4.2 anterior este script lleva a cabo la formación de grupos siguiendo el procedimiento del método de este escenario.

Listado 4.6. Regla de creación de equipos de trabajo.

```
Evento: pelican.workspace.activity.updateState
Disparador: event.newActivity.activityTemplate.name=='Eleccion grupos' &&
            event.oldActivity.state=='PENDING' &&
            event.newActivity.state=='PASSED'
Prioridad: 1
Script:
 def teamTemplate = social.getGroupTemplate ('equipo de trabajo') ❶
 def expertGroupTemplate = social.getGroupTemplate ('grupo experto')
 def expertGroups = parentGroup.getSubgroups (expertGroupTemplate)
 def student = social.getActor ('estudiante')
 common.callScript ('SOC.Jigsaw Groups', ❷
                   ['groups'        : expertGroups,
                    'groupTemplate' : teamTemplate,
                    'parentGroup'   : parentGroup,
                    'baseName'      : 'equipo',
                    'actor'         : student])
```

- **R7 Despliegue de actividades Jigsaw.** Una vez creados los equipos de trabajo, la plantilla de Jigsaw debe ser desplegada sobre sus espacios de trabajo para permitir a los estudiantes terminar la experiencia. La regla R7, que se muestra en el listado 4.7, lleva a cabo este proceso. Como puede apreciarse tanto el evento como el disparador de esta regla coincide con el de la regla R6. Por tanto ambas se ejecutarán a la vez. Nuevamente las prioridades de cada uno aseguran que esta regla se ejecutará después de la anterior. El script de adaptación es muy similar al de la regla R5. En primer lugar, recupera la colección de los equipos de trabajo (❶). Después recupera las plantillas de proyecto y actividad de Jigsaw (❷). Y finalmente, recorre todos los grupos para crear un proyecto y una actividad de Jigsaw en el espacio de trabajo de cada equipo (❸).

Listado 4.7. Regla de despliegue de actividades Jigsaw.

```
Evento: pelican.workspace.activity.updateState
Disparador: event.newActivity.activityTemplate.name=='Eleccion grupos' &&
            event.oldActivity.state=='PENDING' &&
            event.newActivity.state=='PASSED'
Prioridad: 2
Script:
 def parentGroup = event.newActivity.getProject().getWorkspace().getOwner() ❶
 def groupTemplate = social.getGroupTemplate ('equipo de trabajo')
 def teams = parentGroup.getSubgroups (groupTemplate)
 def projectTemplate = collaboration.getProjectTemplate ('Jigsaw') ❷
 def activityTemplate = collaboration.getActivityTemplate ('Actividad de Jigsaw')
 for (team in teams) { ❸
   def project = collaboration.createProject (projectTemplate, team)
   def activity = collaboration.createActivity (activityTemplate)
   collaboration.addActivity (activity, project) }
```

4.5.2. División colaborativa del trabajo

En el escenario anterior se ha puesto de manifiesto cómo pueden utilizarse las capacidades del subsistema de intervención de Pelican para realizar transformaciones sobre la comunidad virtual de aprendizaje donde se desarrolló el escenario. No obstante, el control del flujo de trabajo de las actividades – esto es, su secuenciamiento a lo largo del tiempo – es, en ese caso, llevado a cabo manualmente por el profesor.

En el escenario que presentamos en esta sección mostraremos cómo pueden automatizarse estas labores con Pelican. Muchos entornos de e–learning ya disponen de un mecanismo para el secuenciamiento de actividades similar al descrito en [IMS–SS]. Sin embargo, en todos estos casos, una vez establecida una lógica concreta de secuenciamiento ésta no puede ser alterada dinámicamente durante el desarrollo de la experiencia. Como veremos aquí, las capacidades declarativas del subsistema de intervención permiten realizar estos cambios y hacerlos efectivos sin necesidad de paralizar el desarrollo del escenario. Es más, las transformaciones realizadas sobre el flujo de trabajo instruccional pueden hacerse depender de las decisiones colaborativas tomadas por los estudiantes en la experiencia. Para ilustrar esta flexibilidad presentamos un escenario donde la división del trabajo en actividades colaborativas es realizada por los estudiantes:

- **Descripción.** El primer paso en el desarrollo de este escenario comienza presentando a los miembros del grupo un problema complejo para que reflexionen sobre él. Después, los alumnos deben negociar cuál es la estrategia más apropiada de descomposición del trabajo para resolver el problema. Como fruto de esta descomposición debe obtenerse una colección de subproyectos formados por actividades colaborativas que serán realizadas secuencialmente, así como una asignación de responsabilidades que divide a los miembros del grupo en subgrupos y les asigna un subproyecto. Cuando todos los subproyectos han finalizado – esto es, cuando ha finalizado la última actividad de cada uno de ellos – se procede a realizar una última actividad conjunta, a nivel de grupo, para integrar los resultados de cada subproyecto.

- **Contexto.** El escenario de división colaborativa del trabajo es desarrollado por los estudiantes de una clase organizados en grupos. Es decir, la unidad social de trabajo es el grupo y no la clase. Supondremos, de cara a los ejemplos, que el desarrollo de este escenario se enmarca dentro de un contexto pedagógico de educación formal con alumnos de secundaria de edades entre los 15 y 16 años.

- **Dominio.** En la evaluación que hemos hecho de este escenario hemos tomado como ejemplo la planificación de una excursión a El monte de El Pardo. En ella, los estudiantes deben diseñar un itinerario y realizar un plan de trabajo basado en la realización de observaciones de la flora y fauna presentes en dicho ecosistema. El lector puede suponer, sin embargo, que este escenario puede aplicarse a cualquier situación pedagógica que implique la resolución de un problema complejo y abierto. Es decir, donde varias posibles soluciones sean igualmente aceptables.

- **Fases de desarrollo.** El desarrollo del escenario de división colaborativa del trabajo está formado por un total de cuatro fases que son desarrolladas secuencialmente. A continuación describimos brevemente cada una de ellas:
  - *Análisis del problema.* En esta fase se entrega a los estudiantes un problema para que realicen una primera toma de contacto y analicen su complejidad.
  - *División colaborativa del trabajo.* En esta fase los estudiantes plantean una descomposición del trabajo en subproyectos y determinan quién se encargara de su desarrollo.
  - *Realización de subproyectos.* Durante esta fase se lleva a cabo la realización en paralelo de todos los subproyectos que han sido definidos en la fase anterior. Esta fase no termina hasta que cada subgrupo haya finalizado su subproyecto.
  - *Integración de resultado.* En la última fase, los estudiantes se reúnen para componer todos los productos parciales generados en cada subproyecto e integrarlos en un único informe final de la experiencia.

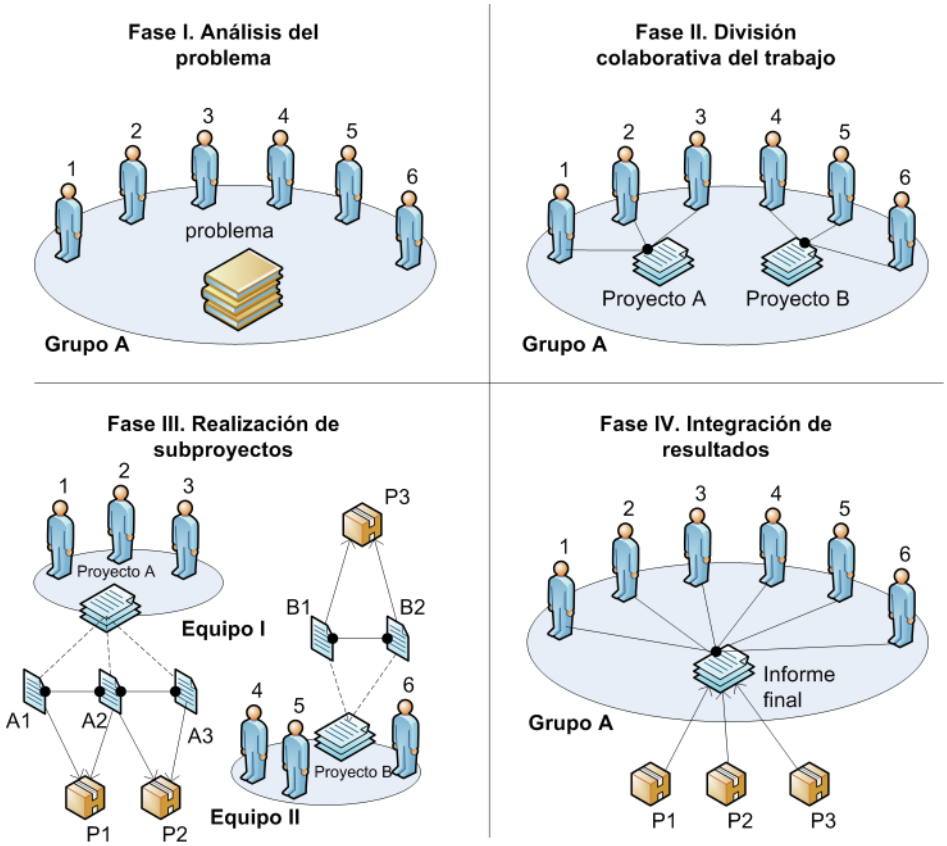

Figura 4.5. Fases del escenario de división colaborativa del trabajo.

El diagrama de la figura 4.5 representa cada una de las cuatro fases de este escenario. Como puede apreciarse, durante el análisis del problema los estudiantes analizan el problema que se les ha entregado. Después éstos deciden, de forma consensuada,

que la solución requiere realizar los proyectos A y B de cuyo desarrollo se encargarán los estudiantes 1, 2 y 3 para el A y 4, 5 y 6 para el B. En la fase III se realizan las actividades colaborativas asociadas a sendos proyectos obteniéndose tres productos parciales. Y finalmente, en la fase IV, éstos son integrados en un informe final donde participan todos los miembros del grupo.

Una vez descrita la experiencia pasaremos a describir el diseño que debe ser llevado a cabo en cada uno de los cuatro modelos que constituyen la especificación del escenario.

Diseño del modelo de sociedad

El modelo de sociedad que utilizaremos para implantar este escenario en Pelican esta formado por los mismos actores que el escenario de Jigsaw por tanto para no resultar reiterativos centraremos en este apartado nuestra discusión en presentar qué plantillas de proyecto es necesario definir y cuál es su composición interna. En concreto podemos distinguir en el modelo de sociedad dos plantillas de proyecto:

- **Grupo de trabajo.** El grupo de trabajo representa a cada uno de los colectivos de estudiantes donde se desarrolla la experiencia. En el ejemplo de la figura 4.5, los seis estudiantes implicados en la misma constituirían un grupo de trabajo. Cada grupo de trabajo incluye al profesor y a los estudiantes así como un número, a priori desconocido, de equipos colaborativos.
- **Equipo colaborativo.** El equipo colaborativo se define para dar soporte a cada uno de los colectivos que se formarán en torno al desarrollo de los subproyectos fruto de la descomposición del trabajo. Esta plantilla es formada por el profesor y un conjunto de estudiantes, que por construcción será un subconjunto de los miembros del grupo de trabajo.

Cuando la experiencia pasa a la fase de desarrollo hemos de suponer que los grupos de trabajo están ya formados. Es decir, consideraremos que la formación de grupos de trabajo no forma parte de las fases de este escenario. La creación de los equipos colaborativos especificos, por el contrario, será realizada dinámicamente una vez que se conozca el plan de descomposición del trabajo y en particular el número de ellos y los estudiantes concretos que los constituyen. Como se trata de un aspecto dinámico postergaremos su discusión más adelante al discutir el modelo de intervención.

Diseño del modelo de colaboración

Para dar soporte a las fases de desarrollo de este escenario es necesario definir una plantilla de proyecto constituida por tres plantillas de actividad. A continuación pasamos a describir brevemente cada una de ellas:

- **Análisis del problema.** En esta plantilla se incluyen todos los recursos y referencias Web necesarios para proporcionar a los miembros del grupo de trabajo una completa descripción del problema a resolver. Su desarrollo se alinea con la primera fase de este escenario.

- **División colaborativa del trabajo.** La plantilla de división colaborativa del trabajo incluye todas las herramientas necesarias para dar soporte a la especificación por parte de los estudiantes del plan de descomposición del trabajo y de asignación de responsabilidades descrito en la fase II (véase el modelo de integración en el siguiente apartado).
- **Integración de resultados.** La plantilla de integración de resultados se corresponde con la fase IV de este escenario. En ella se proporciona a los estudiantes las herramientas necesarias para general el informe final.

El desarrollo de esta experiencia requiere desplegar, sobre el espacio de trabajo del grupo, estas plantillas para obtener sendas actividades. Sin embargo, uno de los requerimientos del escenario es automatizar el secuenciamiento de las mismas para evitar que sean los profesores y administradores lo que tengan que activarlas y desactivarlas manualmente. Esta es una labor que será realizada por el subsistema de intervención y como tal su especificación será abordada más adelante al presentar dicho modelo. No obstante, el diseñador instruccional puede definir en este punto el documento XML que será utilizado por el motor de workflow para realizar el secuenciamiento de las actividades y almacenarlo en el repositorio de grupo de LOR para que, desde allí, pueda ser accedido por la plataforma (véanse scripts de control de flujo de trabajo en la sección 4.3). El listado 4.8 muestra el contenido de este documento:

Listado 4.8. Especificación inicial de la lógica de secuenciamiento

```
<workflow>
 <sequence>
  <activity name='DoL 01. Presentacion'/>
  <activity name='DoL 02. Division colaborativa del trabajo'/> ❷
  <activity name='DoL 03. Integracion de resultados'/>
 </sequence>
</workflow>
```

Esta especificación prescribe que las tres actividades anteriores deben ser llevadas a cabo de forma secuencial. Sin embargo, la fase III del escenario – esto es, la realización de los subproyectos – no está incluida en la misma. Por ello, tras la obtención de un plan de descomposición, al finalizar la segunda actividad, es necesario llevar a cabo de manera automática varias transformaciones sobre el escenario:

- **Crear plantillas de proyecto y actividad para los subproyectos.** Para cada subproyecto definido en el plan de descomposición de la segunda actividad debe crearse una plantilla de proyecto que incluya plantillas de actividad nuevas para cada una de las actividades descritas en dicho plan.
- **Desplegar los nuevos proyectos sobre los equipos colaborativos.** Obtenidas las plantillas de proyecto y actividad que describen el plan de descomposición determinado por los estudiantes, deben desplegarse dichas plantillas sobre los espacios de trabajo correspondientes atendiendo a la asignación de responsabilidades descrita en el plan.

- **Incluir en el flujo de trabajo el secuenciamiento de los subproyectos.** La descripción de la lógica de secuenciamiento inicial, mostrada en el listado 4.8 anterior, debe ser alterada dinámicamente para incorporar, tras la segunda actividad (❷), las nuevas actividades definidas como fruto de la decisión colaborativa de descomposición del trabajo. Éstas deben ser incluidas de manera que su desarrollo se realice en paralelo para cada subproyecto y secuencialmente para cada una de las actividades que los constituyen. Una vez hecho esto, debe actualizarse el motor de workflow para que incorpore esta nueva especificación. En el listado 4.9 se muestra cómo quedaría el listado 4.8 tras esta transformación para el ejemplo concreto presentado en la figura 4.5.

Listado 4.9. Especificación transformada de la lógica de secuenciamiento.

```
<workflow>
 <sequence>
   <activity name='DoL 01. Presentacion'/>
   <activity name='DoL 02. Division colaborativa del trabajo'/>
     <split>
       <sequence>
         <activity name='A1'/>
         <activity name='A2'/>
         <activity name='A3'/>
       </sequence>
       <sequence>
         <activity name='B1'/>
         <activity name='B2'/>
       </sequence>
     </split>
   <activity name='DoL 03. Integracion de resultados'/>
 </sequence>
</workflow>
```

Como puede apreciarse, este listado incluye, tras la actividad 2, un elemento *split* que indica al motor de workflow que todos sus elementos integrantes serán desarrollados en paralelo. En concreto, en él se incluyen dos secuencias: las correspondientes al proyecto A, formado por las actividades A1, A2, y A3 y las del proyecto B, formado por las actividades B1 y B2. El resultado de todo esto en tiempo de ejecución puede verse representado gráficamente en la figura 4.6 donde se muestra el flujo de trabajo instruccional desarrollado a nivel de equipo y grupos. En efecto, las dos primeras actividades son realizadas a nivel de grupo. Tras la actividad 2 la plataforma pasará a activar las actividades A1 y B1 desplegadas en sendos espacios de trabajo. Igualmente hasta que A3 y B2 no estén finalizas la plataforma no transitará a la actividad 3.

Es necesario advertir que en la plataforma no existe ningún artefacto específico para llevar a cabo este secuenciamiento. Todo esto es conseguido a través del uso de los scripts de control de flujo de trabajo que fueron descritos en la sección 4.3 y de las reglas que discutiremos más adelante en el modelo de intervención.

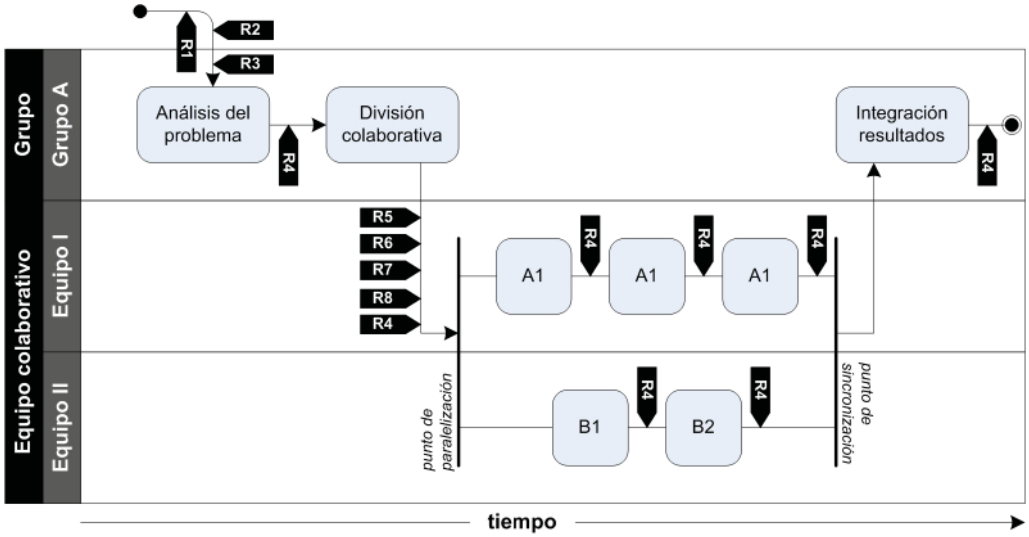

Figura 4.6. Flujo de trabajo instruccional del escenario de división del trabajo.

Diseño del modelo de integración

En las tres plantillas de actividad iniciales que fueron descritas en el apartado anterior se da acceso a tres de las herramientas del ecosistema tecnológico de ENLACE. A continuación describimos donde se definen sus servicios y que uso se hace de los mismos:

- **CHAT.** En la actividad de división colaborativa del trabajo y en la de integración de resultados se proporciona un servicio de CHAT a los miembros del grupo para que puedan discutir cómo realizarán el plan de descomposición del trabajo y cómo elaborarán el informe final.

- **LOR.** Estas mismas actividades también ofrecen acceso al repositorio de grupo mantenido por LOR. Concretamente en la actividad de división del trabajo, el repositorio es utilizado para almacenar los objetos que describen la descomposición ideada por los estudiantes. En la actividad de integración, por su parte, el acceso a este repositorio permite a los estudiantes recuperar los productos parciales que se generan durante el desarrollo de los subproyectos.

- **CARDS.** Para describir el plan de trabajo, la segunda actividad incluye tres servicios de CARDS para crear fichar asociadas a tres plantillas de ficha definidas por el diseñador. A continuación describimos el propósito de cada una de ellas:

  – *Plantilla para la definición de subproyectos.* Esta plantilla permite a los estudiantes definir los proyectos que deben ser creados dinámicamente para dar soporte al desarrollo de los subproyectos. Con este servicio de CARDS los estudiantes deben crear una ficha para cada uno de ellos indicando las actividades que los constituyen y los servicios (preconfigurados) que deben incluir a elegir entre charla de equipo con CHAT, votación de equipo con AGORA y repositorio de grupo o equipo con LOR.

– *Plantilla para la formación de equipos.* La plantilla de equipos permite indicar los estudiantes que constituirán cada uno de los equipos colaborativos que deben ser creados dinámicamente para desarrollar los subproyectos.

– *Plantilla para la asignación de responsabilidades.* Esta plantilla permite describir qué subproyecto es asignado a cada equipo colaborativo y, por tanto, Es utilizada por la plataforma para desplegar apropiadamente los subproyectos en los espacios de trabajo correspondientes.

Según lo expuesto, los servicios de las actividades construidas dinámicamente son creadas por la plataforma bajo demanda de los usuario. En el siguiente apartado describimos todas las reglas que conforman el modelo de intervención del escenario.

Diseño del modelo de intervención

El modelo de intervención de este escenario está formado por un total de ocho reglas políticas. La aplicación de las mismas a lo largo del desarrollo del flujo de trabajo instruccional aparece representada en la figura 4.6. Como se ve las tres primeras reglas (R1 a R3) se aplican sobre el arranque ya que se encargan de configurar diversos aspectos del escenario. Las reglas R5 a R9 se aplican tras la actividad de división colaborativa del trabajo ya que su misión es preparar el entorno para permitir la realización en paralelo de todos los subproyectos. La regla R4 se aplica en cada transición ya que es la responsable de realizar las transiciones sobre el motor de workflow. A continuación pasamos a comentar cada una de estas reglas:

- **R1 Despliegue de actividades.** Para comenzar el escenario, el profesor despliega el proyecto sobre el espacio de trabajo de los grupos que vayan a desarrollar la experiencia. En este caso en concreto, todas las plantillas de actividad del proyecto deben ser asimismo desplegadas. Esta regla, ilustrada en el listado 4.10, se encarga de automatizar esta tarea. Como se ve, la regla está asociada al evento de despliegue de un proyecto. Las condiciones de disparo indican que ésta debe ser ejecutada cuando el proyecto que se está desplegando se corresponde con el de este escenario. El script se encarga de obtener todas las plantillas de actividad del proyecto (❶) y las despliega sobre el espacio de trabajo del grupo (❷).

Listado 4.10. Regla para el despliegue de las actividades.

```
Evento: pelican.workspace.management.project.new
Disparador: event.project.projectTemplate.name=='Division del trabajo'
Prioridad: 1
Script:
 def group = event.group
 def project = event.project
 def projectTemplate = project.projectTemplate
 def activityTemplates = projectTemplate.activityTemplates ❶
 for (activityTemplate in activityTemplates) {
   def activity = collaboration.createActivity (activityTemplate)
   collaboration.addActivity (activity, project) ❷ }
```

- **R2 Creación del motor de workflow.** Para poder realizar el secuenciamiento de las tres actividades desplegadas con la regla anterior es necesario crear un motor de workflow de acuerdo a las especificaciones que se mostraron en el listado 4.8 y que fueron almacenadas en el repositorio de grupo. Esta regla, se ejecuta justo después de la regla R1 puesto que tiene idéntico disparador pero su prioridad es 2. Como puede verse en la el listado 4.11, el script de adaptación comienza recuperando esta especificación mediante la invocación del script *Read Workflow* de nuestra librería (❶). Después construye el motor invocando a *Create Workflow* (❷). Y por último, selecciona a un miembro del grupo para almacenar en su contexto este artefacto (❸).

Listado 4.11. Regla de creación del motor de workflow.

```
Evento: pelican.workspace.management.project.new
Disparador: event.project.projectTemplate.name=='Division del trabajo'
Prioridad: 2
Script:
 def project = event.project
 def group = event.group
 def xmlWorkflow = common.callScript ('WKF.Read Workflow', ❶
                                      ['repository' : group.name,
                                       'learningObject' : 'Workflow']).
 def doLWorkflow = common.callScript ('WKF.Create Workflow', ❷
                                      ['xmlWorkflow' : xmlWorkflow,
                                       'project' : project,
                                       'group' : group])
 def aStudent = group.getActorBinding('estudiante').users[0] ❷
 common.saveInContext (aStudent, 'doLWorkflow', doLWorkflow)
```

- **R3 Arranque del motor de workflow.** En este punto es necesario recuperar del contexto de usuario el motor de workflow previamente creado y arrancarlo. La regla del listado 4.12 tiene el mismo evento y disparador que las dos anteriores pero con prioridad 3 para que se ejecute en este orden. El script se limita a recuperar del contexto el motor (❶), invocar el método de arranque (❷) y volver a salvaguardar el mismo (❸). Esta sencilla instrucción no se ha incluido al final del script de R2 para mantener la independencia de las reglas y fomentar su reutilización.

Listado 4.12. Regla de arranque del motor de workflow.

```
Evento: pelican.workspace.management.project.new
Disparador: event.project.projectTemplate.name=='Division del trabajo'
Prioridad: 3
Script:
 def group = event.group
 def aStudent = group.getActorBinding('estudiante').users[0] ❶
 def wf = common.loadFromContext (aStudent, 'doLWorkflow')
 wf.open() ❷
 common.saveInContext (aStudent, 'doLWorkflow', wf) ❸
```

- **R4 Transición de actividad.** La regla de transición de actividad debe ser lanzada a ejecución cada vez que finaliza una actividad del flujo de trabajo. Esto es, cuando algún miembro del grupo pone el estado de la misma a pendiente de evaluar. En efecto, como se puede ver en el listado 4.13, el evento al que está asociado esta regla es el de cambio de la actividad y las condiciones de disparo indican que sólo deberá lanzarse a ejecución cuando el proyecto de la misma sea el de división colaborativa del trabajo. La prioridad se ha puesto en 10 para garantizar que esta regla siempre se ejecute en último lugar. El script de adaptación de R4 comienza recuperando el motor de workflow del contexto de usuario donde se encuentra almacenado (❶). Después se invoca sobre él al método *next* que toma como argumento la actividad asociada al evento. Este método informa al motor de que dicha actividad ha finalizado lo que actualiza el estado interno del mismo y habilita las actividades sobre los espacios de trabajo para que sean accesibles por los estudiantes (❷). Finalmente, se salvaguarda el motor de workflow, y se consulta al mismo cuál es la familia de actividades actualmente abiertas (❸) para mandar un mensaje a todos los miembros del grupo invitándoles a que procedan a su desarrollo. Si no hay más actividades pendientes el proyecto habrá finalizado (❹).

Listado 4.13. Regla de transición de actividad

```
Evento: pelican.workspace.activity.updateState
Disparador: event.newActivity.project.projectTemplate.name=='Division...' &&
            event.newActivity.state=='PENDING'
Prioridad: 10
Script:
 def activity = event.newActivity
 def group = activity.project.workspace.owner
 def aStudent = group.getActorBinding('estudiante').users[0] ❶
 def wf = common.loadFromContext (aStudent, 'doLWorkflow')
 wf.next (activity) ❷
 common.saveInContext (teacher, 'doLWorkflow', wf)
 def nextActivities = wf.state() ❸
 for (user in group.users) { ❹
   if (nextActivities.isEmpty())
     message = 'Enhorabuena, has finalizado el proyecto.'
   else
     message = 'Por favor, desarrollar las actividades $nextActivities'
   common.sendMessage (user, message)
 }
```

- **R5 Creación de subproyectos.** Finalizada la actividad de división del trabajo, es necesario recuperar del repositorio de grupo todas las fichas que describen algún subproyecto fruto de la descomposición del trabajo. La regla R5 que se muestra en el listado 4.14 lleva a cabo esta tarea. El evento, el disparador y la prioridad de la misma indican que será ejecutada en primer lugar al poner el estado de la actividad segunda en estado *pendiente*. El script comienza recuperando el identificador interno en LOR para el repositorio del que tienen que recuperarse las

ficha de proyecto (❶). Después se pide al LOR, mediante una consulta (❷), que devuelva todas las fichas de división del trabajo en dicho repositorio. Finalmente, se recuperan los identificadores en el LOR de las fichas y se invoca al script de división del trabajo de la librería estándar para que proceda con la creación de los proyectos (❸). Este script se encarga de recuperar del LOR la ficha indicada por el identificador pasado como parámetro y de ir procesando su estructura interna en XML. Creando una plantilla de proyecto, y añadiéndole las plantillas de actividad con los servicios allí definidos.

Listado 4.14. Regla de creación de subproyectos

```
Evento: pelican.workspace.activity.updateState
Disparador: event.newActivity.activityTemplate.name=='Division colaborativa' &&
            event.newActivity.state=='PENDING'
Prioridad: 1
Script:
 def rep = event.newActivity.project.workspace.owner.name
 def xmlResponse = tools.callService (endPoint,'getRepositoryByName', [rep])
 def response = tools.readXml (xmlResponse)
 def repId = response.repositories.repository.@id.text () ❶
 def xmlQuery = ''' <query> ❷
                       <entity>
                          <slot name='lor.general.repository'>
                             <value>$repId</value>
                          </slot>
                          <slot name='enlace.ficha.plantilla'>
                             <value>DivisionDelTrabajo</value>
                          </slot>
                       </entity>
                  </query> '''
 def xmlSearchResponse = tools.callService (clientEndPoint,'search',[xmlQuery])
 def searchResponse = tools.readXml (xmlSearchResponse)
 def learningObjectIds = searchResponse.query_result.entity.@id
 for (learningObjectId in learningObjectIds) {
   def aLearningObjectId = learningObjectId.text()
   common.callScript ('INT.CRD.Division Of Labour (by Id)',
                     ['learningObjectId' : aLearningObjectId]) ❸
 }
```

- **R6 Formación de equipos colaborativos.** El único cambio en la especificación de las condiciones de disparo R6 con respecto a R5 es su prioridad paga garantizar una aplicación secuencial. El propósito de esta regla (listado 4.15) es llevar a cabo la formación de equipos colaborativos una vez obtenida la ficha de formación de los mismos tras la finalización de la segunda actividad. Para ello, el script de adaptación de la regla invoca al script de librería *Group Allocation*, pasándole como parámetro el nombre del repositorio y el de la ficha donde se encuentra descrita la composición de los equipos así como el grupo padre donde deben crearse los

mismos y su plantilla de grupo (❶). Este script de la librería estándar recupera del repositorio indicado la ficha de formación de equipos y la procesa para conocer el nombre de los mismos y la colección de estudiantes que los constituyen. Con esta información y a partir de la plantilla de grupo, y el grupo padre proporcionado, crea los nuevos equipos colaborativos. Una vez finalizada su ejecución el control retorna al script de la regla para terminar de ejecutar las siguientes instrucciones. En concreto, éstas se encargan de introducir al profesor en todos dos equipos (❷).

Listado 4.15. Regla de formación de equipos colaborativos.

```
Evento: pelican.workspace.activity.updateState
Disparador: event.newActivity.activityTemplate.name=='Division colaborativa' &&
            event.newActivity.state=='PENDING'
Prioridad: 2
Script:
 def group = event.newActivity.project.workspace.owner
 def groupTemplate = social.getGroupTemplate ('grupo')
 common.callScript ('INT.CRD.Group Allocation', ❶
                   ['repository' : group.name,
                    'learningObject' : 'DivisionEnGrupo',
                    'parentGroup' : group,
                    'groupTemplate' : groupTemplate])
 def teacherBinding = group.getActorBinding ('profesor') ❷
 def teacher = teacherBinding.users[0]
 def subgroups = group.subgroups
 for (subgroup in subgroups)
   social.bindUserToActor (teacher, teacherBinding.actor, subgroup)
```

- **R7 Despliegue de subproyectos.** Una vez construidos las plantillas de proyecto y actividad para cada subproyecto y formados los equipos de colaboración, es necesario desplegarlas en los espacios de trabajo de los equipos correspondientes. De ello se encarga la regla R7 cuyo código se muestra en el listado 4.16. Como puede apreciarse las condiciones de disparo que esta regla solo se ejecutará tras la regla R6. El script delega todo su trabajo en el script de librería Project Allocation (❶), que procesa la ficha correspondiente y realiza el despliegue de acuerdo a su contenido.

Listado 4.16. Regla de formación de equipos colaborativos.

```
Evento: pelican.workspace.activity.updateState
Disparador: event.newActivity.activityTemplate.name=='Division colaborativa' &&
            event.newActivity.state=='PENDING'
Prioridad: 3
Script:
 def group = event['activity.new'].project.workspace.owner
 common.callScript ('INT.CRD.Project Allocation', ❶
                    ['repository' : group.name,
                     'learningObject' : 'RepartoTrabajo'])
```

- **R8 Transformación del flujo de trabajo.** Por ultimo es necesario modificar la especificación del flujo de trabajo para incluir las nuevas actividades según el listado 4.9. La regla R8 (listado 4.17) se ejecuta después de R7 y lleva a cabo esta labor. Su script comienza construyendo un nuevo motor de workflow basado en una especificación que comienza por la actividad de división colaborativa del trabajo y continúa con un desarrollo en paralelo de cada secuencia de actividades. La construcción de este XML se hace iterando las actividades de los subproyectos desplegados en cada equipo colaborativo (❶). La construcción del nuevo motor se lleva a cabo invocando el script de librería *Create Workflow* (❷). Después se arranca el motor (❸). Ahora ambos motores – el nuevo y el que está en uso – tienen el mismo estado puesto que la actividad en curso en ambos es la segunda. Recuperado el motor en uso, se invoca sobre éste el método *changeComponent* que sustituye en él la actividad actual por el nuevo motor integrándolo como parte de su especificación (❹). Con esto se consigue alterar dinámicamente la lógica de secuenciamiento sin necesidad de parar la experiencia.

Listado 4.17. Regla de transformación del flujo de trabajo.

```
Evento: pelican.workspace.activity.updateState
Disparador: event.newActivity.activityTemplate.name=='Division colaborativa' &&
            event.newActivity.state=='PENDING'
Prioridad: 4
Script:
 def act = event.newActivity
 def prj = act.project
 def grp = prj.workspace.owner
 def subgroups = group.subgroups
 def xmlSubWf = '''<workflow> ❶
                    <sequence>
                      <activity name='$act' project= '$prj' group='$grp'/>
                      <split>'''
 for (subgrp in subgroups) {
  def subprojects = subgroup.workspace.projects
  for (aPrj in subprojects){
    xmlSubWf += '<sequence>'
    def activities = aPrj.activities
    for (anAct in activities)
      xmlSubWf += "<activity name='$anAct' project='$aPrj' group='$subgrp'/>"
    xmlSubWf += '</sequence>'  }  }
 xmlSubWf += '</split></sequence></workflow>'
 def subWf = common.callScript ('WKF.Create Workflow', ['xmlWorkflow' : xmlSubWf,
                                'project' : project, 'group' : group])
 subWf.open() ❸
 def aStudent = group.getActorBinding('estudiante').users[0]
 def wf = common.loadFromContext (aStudent, 'doLWorkflow')
 wf.changeComponent (wf.start.components[1], subWf.start) ❹
 common.saveInContext (aStudent, 'doLWorkflow', wf)
```

4.5.3. Debate colaborativo

En las dos subsecciones anteriores presentamos sendos escenarios que ponían de manifiesto cómo pueden ser empleadas las capacidades de intervención dinámica de la plataforma para transformar bajo demanda el modelo de sociedad y de colaboración. En este escenario, centraremos nuestra atención en la adaptación de herramientas externas y veremos cómo el mecanismo de integración dirigido por las herramientas ofrece grandes posibilidades para articular experiencias colaborativas con un elevado nivel de cohesión e interoperabilidad. En concreto, usaremos la herramienta CHAT para mostrar cómo el principio de integración orientado a roles nos permite delegar en Pelican el control de la interacción y vincularlo al flujo de trabajo instruccional. Para ilustrar todo esto hemos escogido como escenario de ejemplo la realización de un debate colaborativo inspirado en el método de pecera descrito en [Barkley et al., 2005]:

- **Descripción.** En la realización de un debate colaborativo, el profesor comienza presentando a los estudiantes un tema de fuerte controversia y pide a éstos que se pronuncien con respecto al mismo. Después, divide a los estudiantes en dos grupos de manera que, en cada uno de ellos, haya un 50% de votantes a favor y otro 50% de votantes en contra. Los grupos son organizados en clase formando dos circulos concéntricos. A los estudiantes del grupo interior se le asigna el rol de participantes mientras que a los del grupo exterior se les da el rol de anotadores. En una primera fase, el grupo de participantes realizará un debate sobre el tema mientras que los anotadores observarán en silencio tomando notas sobre el mismo. Transcurrido un periodo de tiempo determinado por el profesor, el debate concluye ofreciendo a cada participante un ultimo turno de intervención para que exponga sus alegatos finales. Después, los anotadores comentarán por turnos, en voz alta, sus impresiones generales sobre la experiencia. Y finalmente, se hará una nueva votación para comprobar si el debate ha provocado cambios en las opiniones de los estudiantes.

- **Contexto.** Para llevar a cabo adecuadamente este escenario, es importante asegurarse que los dos colectivos de estudiantes sean del mismo tamaño y que la distribución de opiniones sea equilibrada en ambos. Supondremos en este ejemplo, que la clase está formada por un total de 12 estudiante de enseñanza secundaria y que la sesión de trabajo es de 50 minutos donde el tiempo de debate son 10 minutos y cada turno de intervención dura 5 minutos, aunque estos valores pueden ajustarse discrecionalmente en función del número de alumnos.

- **Dominio.** En los ejemplos que ilustran la descripción de este escenario hemos supuesto que el tema de debate es sobre la conveniencia de legalizar el aborto libre en nuestro país y, en caso de ser así, bajo qué condiciones. Aunque este método pedagógico puede ser aplicado a otros dominios siempre ha de tenerse en cuenta que el tema que se presente a los estudiantes debe ser lo suficientemente controvertido y de respuesta dicotómica para poder estratificar fácilmente a los estudiantes en dos colectivos (votantes a favor frente a votantes en contra).

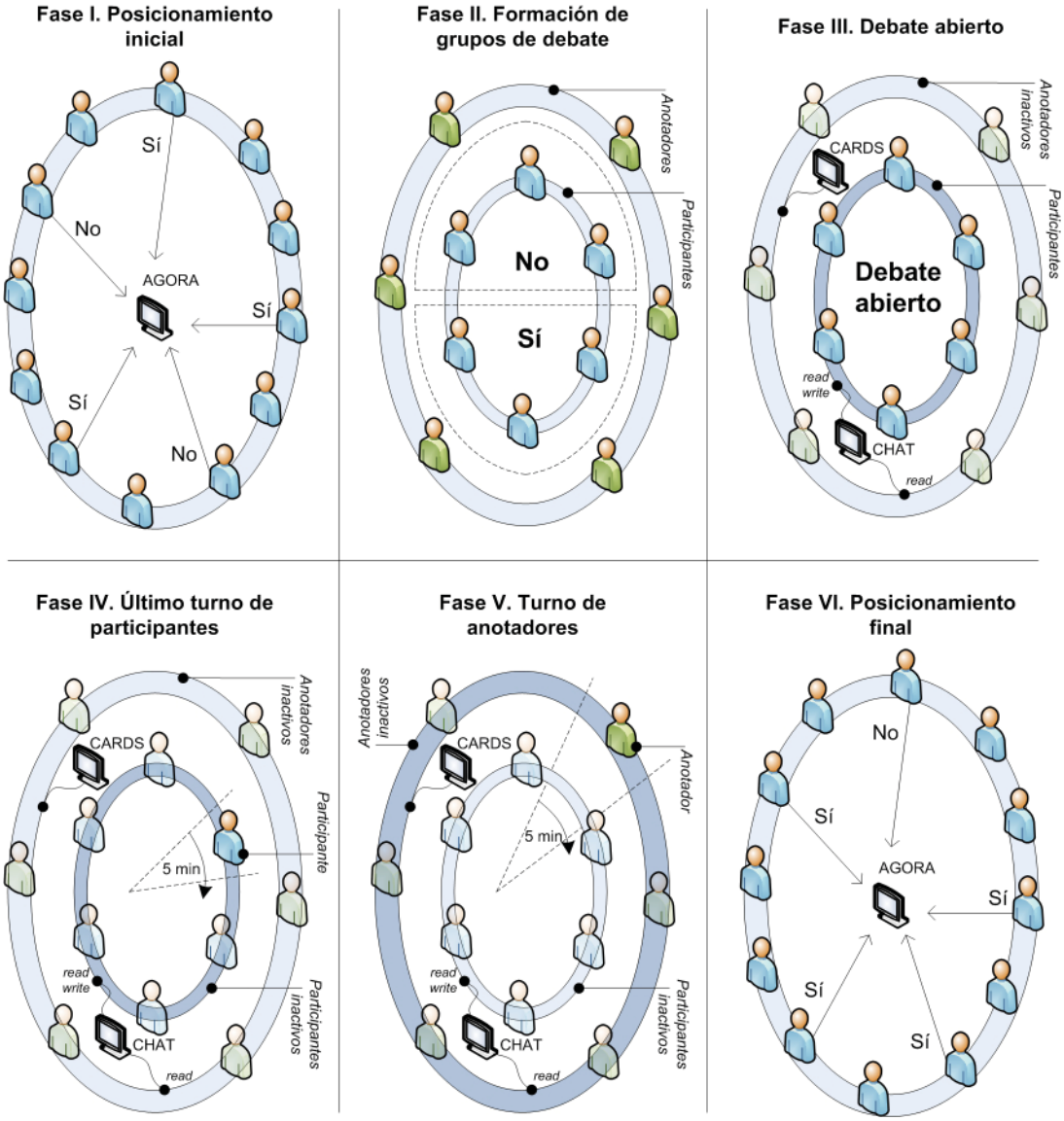

Figura 4.7. Fases del escenario de debate colaborativo.

- **Fases de desarrollo.** El desarrollo de este escenario implica un total de seis fases que aparecen representadas en la figura 4.7 y que pasamos a describimos brevemente a continuación:

  – *Posicionamiento inicial.* En la primera fase de este escenario el profesor presenta a los estudiantes el tema de debate ofreciéndoles argumentos a favor y en contra y solicita a éstos que voten cuál es su posición al respecto.

  – *Organización de grupos de debate.* El profesor divide a los estudiantes en dos colectivos, votantes a favor y votantes en contra. A partir de aquí, crea dos grupos formados al 50% por estudiantes de cada uno de estos colectivos. Este reparto equitativo garantiza la controversia dentro de cada grupo. A uno de ellos les asigna el rol de participantes en el debate y al otro el de anotadores.

– *Debate abierto.* Los participantes realizan un debate abierto donde todos ellos pueden intervenir libremente cuando lo deseen. Mientras tanto, los anotadores observan en silencio y registran los acontecimientos significativos ocurridos durante el debate.

– *Último turno de palabra para participantes.* El debate finaliza concediendo un último turno de intervención a cada participante para que exponga sus alegatos finales a favor o en contra.

– *Turno de palabra para anotadores.* En esta fase, los participantes se mantienen en silencio y cada anotador dispone de un turno de intervención para compartir con toda la clase sus impresiones sobre la experiencia.

– *Posicionamiento final.* Finalmente, toda la clase vuelve a votar para comprobar si ha habido cambio de opiniones entre los estudiantes.

Diseño del modelo de sociedad

Como se desprende de la descripción anterior, este escenario implica, en su dimensión social, dos tipos de usuarios: profesor y estudiante, que serán representados por los actores de su mismo nombre descritos en el capítulo 2. Por otro lado, también se requiere dar soporte a tres tipos diferentes de agrupaciones: la clase, el grupo de participantes y el grupo de anotadores. Para el primer tipo utilizaremos la plantilla de proyecto *clase* que como ya ha sido descrita anteriormente estará formada por los estudiantes y el profesor. Sin embargo, para los dos últimos tipos, no definiremos plantillas de grupo ya que el soporte a esta estratificación será mantenida a través del uso de roles, tal y como discutiremos más adelante en el modelo de integración.

Diseño del modelo de colaboración

Dado que la clase es el único nivel social existente en este escenario, todas las actividades colaborativas del mismo se desarrollan a ese nivel. En concreto, las seis fases descritas anteriormente y representadas en la figura 4.7, se articulan en tres plantillas de actividad. A continuación describimos la responsabilidad de cada una de ellas:

- **Posicionamiento inicial.** Esta plantilla permite desarrollar la primera fase del escenario donde se pone a disposición de los estudiantes diferentes referencia Web relacionadas con el aborto. Desde aquí también los estudiantes emitirán un voto para expresar su posicionamiento, a favor o en contra, de este tema.

- **Debate colaborativo.** El debate colaborativo resulta la actividad central de este escenario ya que se alinea con las fases II a V del mismo. Así, dentro de ella, los participantes comienzan realizando el debate abierto y lo terminan con el último turno palabra para dar paso a los anotadores que cierran la actividad exponiendo, por turnos, las conclusiones extraídas.

- **Posicionamiento final.** Finalmente, se realiza la actividad de posicionamiento final, que consiste en la repetición de una votación de posicionamiento para registrar los cambios de opinión.

Pese a que, por simplicidad, en este escenario no hemos incluido lógica declarativa de secuenciamiento ha de suponerse que el profesor se encarga de garantizar que el desarrollo de las actividades se hará en el orden en que acabamos de describirlas. Las facilidades de las interfaces del subsistema de colaboración (ver figura 3.15) permiten activarlas y desactivarlas manualmente para darles acceso desde los espacios de trabajo. La figura 4.8 muestra el flujo de trabajo del escenario, donde se han incluido el momento exacto en que las reglas políticas son lanzadas a ejecución como se verá en el modelo de intervención.

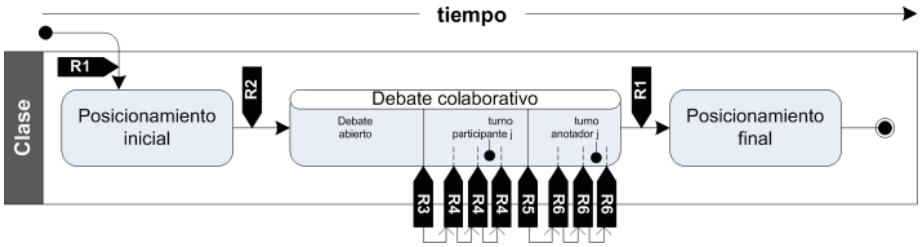

Figura 4.8. Flujo de trabajo instruccional del escenario de debate colaborativo.

Diseño del modelo de integración

Para dar soporte al desarrollo de este escenario se han utilizado tres herramientas del ecosistema tecnológico de ENLACE cuyos servicios de interacción aparecen distribuidos en las plantillas de actividad descritas anteriormente. A continuación describimos qué responsabilidades tienen los mismos y cómo han sido configurados:

- **AGORA.** Para dar soporte a las sesiones de votación inicial y final de las fases I y VI del escenario se han incluido en las actividades 1 y 3 del modelo de colaboración sendos servicios AGORA. Como veremos en el siguiente apartado, la creación de estas sesiones en la herramienta será realizado dinámicamente de forma automática por la plataforma a través de las capacidades del subsistema de intervención.
- **CHAT.** La actividad de debate colaborativo proporciona un servicio de CHAT accesible para todos los miembros de la clase. No obstante, su uso queda regulado a través de la definición de cuatro roles en la plantilla de actividad que incluyen dos propiedades para caracterizar el nivel de acceso a la herramienta: *chat.read* para indicar si el usuario puede o no leer los mensajes de los demás usuarios y *chat.write* para concederle o no permisos de envío de mensajes. A continuación describimos los roles que se han definido:
  - *Participante.* Este rol será asignado a la mitad de los estudiantes de acuerdo a los criterios de estratificación que se describieron en la introducción. Los participantes pueden leer y escribir en el CHAT.
  - *Participante inactivo.* Este rol representa a los participantes que no gozan del turno de palabra durante las fases IV y V del escenario. Como tal, está caracterizado por conceder permiso de lectura de los mensajes del CHAT pero no permiso de escritura. En el ejemplo de la figura 4.7, en la fase IV los participantes están inactivos todo el rato menos en su turno de palabra.

  - *Anotador.* Este rol representa a los estudiantes anotadores que participan por turnos en el debate durante la fase V del escenario. Las propiedades que lo definen semánticamente prescriben que los anotadores pueden enviar y recibir mensajes en el CHAT.

  - *Anotador inactivo.* Los anotadores inactivos representan a los estudiantes anotadores que permanecen en silencio durante las fases III, IV y V del escenario. En las fases IV y V todos los anotadores son inactivos puesto que no está permitida su participación en el debate. En la fase V, sólo el estudiante que tenga temporalmente concedido el turno de intervención es anotador, los demás mantendrán el rol de participante inactivo.

- **CARDS.** La actividad de debate colaborativo incluye además un servicio de CARDS que permite a los estudiantes rellenar fichas de anotación definidas ex–profeso para esta actividad. Con respecto a esta herramienta, los cuatro roles anteriores contienen la propiedad pelican.services.locked configurada de tal forma que sólo los anotadores y los anotadores inactivos tendrán acceso a la misma desde los espacios de trabajo.

La integración orientada a roles que se ha presentado en este apartado permite a la plataforma tomar control de las interacción que se darán con las herramientas de soporte CHAT y CARDS. Aunque la asignación de roles puede ser gestionada manualmente por el profesor, lo interesante en este escenario es permitir que sea el modelo de intervención de la plataforma quien la gestione. En efecto, durante la actividad de debate colaborativo deben medirse los tiempos del debate abierto y de cada turno de palabra en las fases IV y V. Las reglas políticas puedes asociarse a eventos temporales para que provoquen la redistribución de roles según corresponda. En el siguiente apartado pasamos a describir todas estas tareas en profundidad.

Diseño del modelo de intervención

El modelo de intervención de este escenario está formado por un total de seis reglas, cuyo momento exacto se muestra en la figura 4.8. Las similitudes funcionales entre algunas de ellas nos permiten agruparlas en tres categorías diferentes. Las regla R1 se encarga de preparar la sesión de votación en AGORA para la primera y tercera actividad. La regla R2 prepara la configuración inicial de roles para arrancar la sesión de debate abierto. Y finalmente, las reglas R3 y R5 arrancan las fases IV y V del escenario mientras que las reglas R4 y R6 gestionan la concesión de turnos a lo largo de las mismas. Para todas estas reglas hemos definido los parámetros VOTO_SI y VOTO_NO para hacer aplicable el diseño a cualquier dominio. Para este ejemplo debería asignarse, mediante el interfaz Web de Pelican, 'Aborto Sí' al primer parámetro y 'Aborto No' al segundo.

- **R1 Creación de las sesiones de votación inicial y final.** Esta regla se aplica antes del comienzo de las actividades 1 y 3 de nuestro escenario de ejemplo y tienen como misión crear una votación para los miembros de la clase con dos opciones como candidatos: a favor y en contra. En el listado 4.18 se muestra el código de

la misma. En él puede verse cómo el evento al que es asociado la regla es el cambio de estado de la actividad y el disparador indica que esta regla será lanzada a ejecución cuando comiencen alguna de estas dos actividades. Es decir, cuando su estado sea cambiado de *inicio* a *en desarrollo*. Como puede apreciarse, el script asociado se limita a hacer uso del script de librería *Create Session* que tiene como misión la creación de una nueva sesión de votación en AGORA (❶).

Listado 4.18. Regla de creación de votación inicial y final.

```
Evento: pelican.workspace.activity.updateState
Disparador: (event.newActivity.activityTemplate.name=='Posicion inicial' ||
             event.newActivity.activityTemplate.name=='Posicion final') &&
             event.newActivity.state=='DEVELOPMENT'
Script:
 def activity = event.newActivity
 def group =  activity.project.workspace.owner
 def moderator = group.getActorBinding('profesor').users[0]
 common.callScript ('INT.AG.Create Session', ❶
                   ['name' : 'Votacion inicial de debate',
                    'moderator' : moderator,
                    'group' : group,
                    'type' : 'TEXT',
                    'candidates' : [VOTO_SI, VOTO_NO],
                    'mode' : 'NO'])
```

- **R2 Formación de colectivos de debate.** Esta regla se encarga de realizar la formación de los dos colectivos de estudiantes de este escenario (participantes y anotadores) mediante la asignación de sendos roles a los miembros de la clase y a partir de los resultados de la votación de posicionamiento inicial. La regla se muestra en el listado 4.19. En ella se puede ver que el evento al que está asociada la misma es el cambio de estado de la actividad. El disparador prescribe que la regla será lanzada a ejecución cuando la actividad de debate colaborativo alcance el estado *en desarrollo*. El script de la regla comienza recuperando el resultado de la votación de posicionamiento inicial (❶) y a partir de esta información clasifica a los estudiantes en dos tipos: los que votan sí y los que votan no (❷). Una vez hecho esto, combina a los estudiantes de sendos tipos para formar dos colectivos: el de estudiantes participantes y el de estudiantes anotadores compuestos a partes iguales de votantes a favor y en contra en el tema de debate (❸). Organizados los miembros de la clase de esta manera se pasa a asignar el rol *participante* a los participantes y el rol *anotador inactivo* a los anotadores, según corresponde a la fase III (❹). Finalmente, se introducen en el contexto las variables que sirven para contar el número de intervenciones de los participantes (*pRoundRobinCounter*) y los anotadores (*oRoundRobinCounter*), que serán usadas en las fases subsiguientes, se arranca un cronómetro que disparará un evento cuando el tiempo para la sesión de debate abierto haya finalizado y se manda un mensaje a los participantes para indicarles que pueden comenzar a debatir (❺).

Listado 4.19. Regla de formación de colectivos de debate.

```
Evento: pelican.workspace.activity.updateState
Disparador: event.newActivity.activityTemplate.name=='Debate colaborativo' &&
            event.newActivity.state=='DEVELOPMENT'
Script:
 def activity = event.newActivity
 def project = activity.project
 def result = common.callScript ('INT.AG.Get Session Votes', ❶
                                 ['name' : 'Votacion inicial de debate'])
 def votanSi = [] ❷
 def votanNo = []
 for (user in result.keySet()) {
   def aVote = result[user]
   if (aVote.text == VOTO_SI) votanSi += social.getUser(user)
   else votanNo += social.getUser(user)
 }
 def observadores = [] ❸
 def participantes = []
 def max = votanSi.size
 def med = max / 2
 for (index in [0..med]) {
   participantes += votanSi[index]
   participantes += votanNo[index]
 }
 for (index in [med+1..max]) {
   observadores += votanSi[index]
   observadores += votanNo[index]
 }
 def pRole = activity.activityTemplate.getRole ('participante') ❹
 def oiRole = activity.activityTemplate.getRole ('anotador inactivo')
 for (participante in participantes)
   collaboration.bindUserToRole (participante, pRole, activity)
 for (observador in observadores)
   collaboration.bindUserToRole (observador, oiRole, activity)
 common.saveInContext (teacher, 'pRoundRobinCounter', 0)
 common.saveInContext (teacher, 'oRoundRobinCounter', 0)
 common.startTimer (teacher,  ❺
                    'participantsTimer',
                    ['group' : activity.project.workspace.owner,
                     'project' : project,
                     'activity' : activity,
                     'participants': participantes,
                     'observers' : observadores],
                      openDebateTime * 1000, ❻
                      lastParticipantTurnTime * 1000) ❼
 for (user in group.users)
   common.sendMessage (user, 'Comienza el debate...')
```

- **R3 Arranque del último turno de palabra para los participantes.** Esta regla se encarga de preparar la asignación de roles para comenzar la sesión de concesión de turnos para los participantes. En concreto, asigna a todos los participantes, menos al primero, el rol de *participante inactivo* para que no puedan intervenir en el CHAT. Como se vio en la regla anterior, el script de adaptación ejecuta una tarea *startTimer* que arranca un cronómetro configurado con el tiempo que dura la sesión de debate abierto y lo almacena en el contexto de usuario del profesor como una variable de nombre *participantsTimer*. Los cronómetros, al llegar a 0, lanzan un evento de tipo *pelican.core.timeOver*. Precisamente este es el tipo de evento al que esta vinculada la regla R3 cuyo código aparece mostrado en el listado 4.20. Como puede verse, las condiciones de disparo, indican que la regla sólo se ejecuta cuando la actividad del evento sea la del debate colaborativo, y el nombre del cronometro que lanzo el evento sea *participantsTimer.* Esto permite caracterizan univocamente la situación de turno de participantes. Pero ademas el disparador indica que la variable *pRoundRobinCounter* debe estar a 0. Esto es para garantizar que esta regla sólo se lanzara al inicio del primer turno de intervención. El script de adaptación asociado comienza recuperando del subsistema de colaboración los roles *participante* y *participante inactivo* de la actividad en curso (❶) y después recorre la lista de participantes (❷) para asignarles el rol de *participante inactivo*, e informarles de que disponen de 5 minutos para su alegato final. La condición en (❸) sirve para garantizar que el primer participante no será inactivo. Finalmente, se incrementa en 1 el valor de *pRoundRobinCounter* ya que el turno del primer participante ha comenzado (❹).

Listado 4.20. Regla de arranque el último turno de palabra para los participantes.

```
Evento: pelican.core.timeOver
Disparador: event.newActivity.activityTemplate.name=='Debate colaborativo' &&
            eventName=='participantsTimer' && pRoundRobinCounter==0
Script:
 def activity = event.activity
 def activityTemplate = activity.activityTemplate
 def group = event.group
 int t = lastParticipantTurnTime / 60
 def pRole = activityTemplate.getRole ('participante') ❶
 def piRole = activityTemplate.getRole ('participante inactivo')
 def participants = activity.getRoleBinding ('participante').users
 def step = 0
 for (participant in participants) { ❷
   if (step > 0) { ❸
     collaboration.bindUserToRole (participant, piRole, activity)
     common.sendMessage (participant, 'Tienes $t minutos para intervenir')
   }
   step = step + 1
 }
 def teacher = group.getActorBinding('profesor').users[0]
 common.saveInContext (teacher, 'pRoundRobinCounter', 1) ❹
```

- **R4 Cambio de turno de palabra al siguiente participante.** Como se representa en la figura 4.8, esta regla se ejecuta cada 5 minutos para conceder el turno de palabra a un nuevo participante. En realidad el cronómetro que se arrancó en la regla R2 lanza un evento por primera vez transcurrido el tiempo inicial de debate abierto (❻) y después continua lanzando eventos a intervalos regulares de tiempo *lastParticipantTurnTime*, 5 minutos en nuestro ejemplo (❼). Estos nuevos eventos son atendidos por la regla R4 para su procesamiento. Como puede verse en el listado 4.21, el disparador de la regla incluye condiciones sobre la variable *pRoundRobinCounter*, para asegurar que el script será lanzado a ejecución sólo para cada uno de los turnos de intervención de participantes que quedan por concederse. Como en R3, el script comienza recuperando los usuarios con roles de *participante* y *participante inactivo* (❶) y delega en un script de librería el cambio de roles de forma circular (❷). Hecho esto, se manda un mensaje al participante invitándole a intervenir (❸) y se actualiza en el contexto de profesor la variable *pRoundRobinCounter* (❹). Cuando se ha asignado el último turno de palabra (❺), ha de pararse el cronometro *participantsTimer* y arrancarse otro cronometro iterativo para controlar los turnos de intervención de los anotadores en la siguiente fase.

Listado 4.21. Regla de cambio de turno de palabra al siguiente participante.

```
Evento: pelican.core.timeOver
Disparador: event.newActivity.activityTemplate.name=='Debate colaborativo' &&
            eventName=='participantsTimer' && pRoundRobinCounter<pSize &&
            pRoundRobinCounter>0
Script:
 def pSize = event.participants.size
 def observadores = event.observers
 int t = lastParticipantTurnTime / 60
 def participants = event.activity.getRoleBinding('participante').users ❶
 def inactives = event.activity.getRoleBinding('participante inactivo').users
 def users = (participants + inactives).sort()
 common.callScript ('FLR.Round-robin Step Role reallocation',
                    ['activity': event.activity,'users': users]) ❷
 def participant = event.activity.getRoleBinding ('participante').users[0]
 common.sendMessage (participant, 'Tienes $t minutos para intervenir') ❸
 def teacher = event.group.getActorBinding('profesor').users[0]
 pRoundRobinCounter = pRoundRobinCounter + 1
 common.saveInContext (teacher, 'pRoundRobinCounter', pRoundRobinCounter) ❹
 if (pRoundRobinCounter == pSize) { ❺
  common.stopTimer ('participantsTimer')
  common.startTimer (teacher, 'observersTimer',
                     ['group' : group,
                      'project' : event.project,
                      'activity' : event.activity,
                      'observers' : observadores],
                       0, lastRecorderTurnTime * 1000)  }
```

- **R5 Arranque del turno de palabra de los anotadores.** Esta regla es muy similar a la regla R3 y su misión es preparar la asignación de roles para llevar a cabo la fase V del escenario. El listado 4.22 muestra que el evento de la regla es *pelican.core.timeOver* y el disparador indica que ésta sólo será ejecutada en el primer turno de intervención. El script comienza asignando a todos los participantes el rol *participante inactivo* (❶). Después asigna a todos los anotadores, menos al primero, el rol de *anotador inactivo* y a éste le envía un mensaje para invitarle a hacer su exposición (❷). Finalmente actualiza en el contexto de usuario la variable contadora de turnos de palabra para anotadores, *oRoundRobinCounter* (❸).

Listado 4.22. Regla de arranque del turno de palabra de los anotadores.

```
Evento: pelican.core.timeOver
Disparador: event.newActivity.activityTemplate.name=='Debate colaborativo' &&
            eventName=='observersTimer' && oRoundRobinCounter==0
Script:
 def activity = event.activity
 def activityTemplate = activity.activityTemplate
 def group = event.group
 int t = lastRecorderTurnTime / 60
 def piRole = activityTemplate.getRole ('participante inactivo') ❶
 def participants = activity.getRoleBinding ('participante').users
 for (participant in participants)
   collaboration.bindUserToRole (participant, piRole, activity)
 def oRole = activityTemplate.getRole ('anotador') ❷
 def ioRole = activityTemplate.getRole ('anotador inactivo')
 def observers = activity.getRoleBinding ('anotador inactivo').users
 def step = 0
 for (observer in observers) {
   if (step > 0)
     collaboration.bindUserToRole (observer, ioRole, activity)
   else {
     collaboration.bindUserToRole (observer, oRole, activity)
     common.sendMessage (observer, 'Tienes $t minutos para hacer tu exposición')
   }
   step = step + 1
 }
 def teacher = group.getActorBinding('profesor').users[0] ❸
 common.saveInContext (teacher, 'oRoundRobinCounter', 1)
```

- **R6 Cambio de turno de palabra al siguiente anotador.** Esta regla es disparada cada nueva vez que el cronometro de anotadores lanza un nuevo evento, tras la primera intervención de anotador. El script asociado (listado 4.23) delega en el script usado en R4 para hacer el cambio circular de roles (❶). Después se manda un mensaje al nuevo anotador para invitarle a participar (❷) y se actualiza, en el contexto de usuario del profesor, el contador de turnos para anotadores. Cuando todos los anotadores han participado se para el cronómetro de anotadores (❸).

Listado 4.23. Regla de arranque del turno de palabra de los anotadores.

```
Evento: pelican.core.timeOver
Disparador: event.newActivity.activityTemplate.name=='Debate colaborativo' &&
            eventName=='observersTimer' && oRoundRobinCounter<oSize &&
            oRoundRobinCounter>0
Script:
 def group = event.group
 def activity = event.activity
 def oSize = event.observers.size
 int t = lastRecorderTurnTime / 60
 def observers = activity.getRoleBinding('anotador').users ❶
 def inactives = activity.getRoleBinding('anotador inactivo').users
 def users = (observers + inactives).sort()
 common.callScript ('FLR.Round-robin Step Role reallocation',
                  ['activity': activity,'users': users])
 def observer = activity.getRoleBinding ('anotador').users[0]
 common.sendMessage (observer, 'Tienes $t minutos para hacer tu exposicion')
 def teacher = group.getActorBinding('profesor').users[0]
 common.saveInContext (teacher, 'oRoundRobinCounter', oRoundRobinCounter++) ❷
 if (oRoundRobinCounter == oSize) ❸
   common.stopTimer ('observersTimer')
```

4.5.4. Elección de delegados

Los escenarios anteriores se han centrado, principalmente, en demostrar las capacidades adaptativas de la plataforma con respecto a los procesos de estratificación social y de colaboración y cooperación. Este escenario es un ejemplo de cómo pueden utilizarse estas mismas capacidades para dar soporte a la monitorización y control de las actividades colaborativas. En concreto, el escenario de elección de delegados que presentamos en esta sección, pone de manifiesto cómo definir en Pelican un modelo de intervención para articular dicha monitorización internamente. Es decir, cómo las reglas políticas permiten dar soporte tanto al cálculo dinámico de los valores de ciertos indicadores de análisis de la colaboración como a la aplicación de transformaciones adaptativas en respuesta de la variación que sufran los mismos con respecto a ciertos valores umbrales establecidos en tiempo de diseño. Los ejemplos de indicadores y las estrategias de control son aquí pretendidamente simplistas pero servirán para ilustrar lo que hemos dicho:

- **Descripción.** En la elección de delegados, algunos de los miembros de la clase comienzan presentado a sus compañeros sus candidaturas. Después, éstos se enfrentan a un debate donde los estudiantes pueden preguntarles acerca de sus programas de actuación y finalmente, se procede a votar para elegir el delegado de clase. Durante el debate, el profesor debe moderar las intervenciones tanto de los candidatos como de los estudiantes. Si un alumno hace un uso abusivo del tiempo de intervención, el profesor podrá penalizarle durante un tiempo sin intervenir. Además si la votación acabara en empate habría que comenzar otro debate.

- **Contexto.** Este escenario se puede realizar durante la hora semanal de tutorías a principio de curso e implica a todos los miembros de la clase. La duración aproximada de la sesión es de unos 50 minutos.
- **Dominio.** Dada su naturaleza, en este escenario no existe ningún dominio de conocimiento para la aplicación del mismo. Durante el desarrollo de esta experiencia formativa, los estudiantes aprenden a seguir las normas protocolarias de los debates moderados y el valor de las decisiones democráticas.
- **Fases de desarrollo.** El desarrollo del escenario de elección de delegados se articula en tres fases consecutivas que aparecen esquemáticamente representadas en la figura 4.9. A continuación describimos brevemente las mismas:
  - *Presentación de candidaturas.* Durante la fase inicial, algunos de los miembros de la clase presentan su candidatura para ser elegidos delegados de clase.
  - *Debate electoral.* En esta fase, todos los estudiantes que presentaron su candidatura en la fase anterior son invitados a participar en un debate dónde serán sometidos a las preguntas de sus compañeros. El debate está moderado por el profesor que concederá discrecionalmente el turno de palabra.
  - *Elección de delegado.* En la fase de elección de delegado, cada estudiante vota a su candidato favorito. En caso de empate seria necesario realizar un nuevo debate y volver a votar.

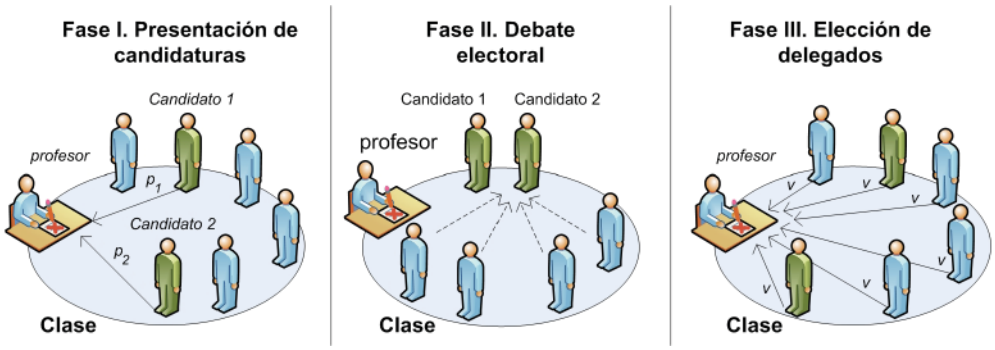

Figura 4.9. Fases del escenario de elección de delegados.

Diseño del modelo de sociedad

El desarrollo de este escenario implica a todos los miembros de una clase. Por tanto, utilizaremos la plantilla de grupo *clase* que como ya vimos en otros escenarios esta formado por los siguientes actores:

- **Estudiantes.** Los estudiantes son los actores principales involucrados en este escenarios puesto que son éstos los que presentan las candidaturas, participan en el debate y votan a su representante.
- **Profesor.** El profesor se incluye para permitirle observar el desarrollo de la experiencia. Nótese que nuestro objetivo es automatizar las tareas de monitorización.

Diseño del modelo de colaboración

Para dar soporte al desarrollo de este escenario, definiremos tres plantillas de actividad que se alinean con cada una de las fases de desarrollo del escenario descritas anteriormente:

- **Presentación de candidaturas.** En la presentación de candidaturas, algunos o todos los estudiantes de la clase generan una candidatura y la comparte con el resto de compañeros de la clase. Esto es soportado por herramientas cuyo uso será discutido en detalle en el siguiente apartado.
- **Debate electoral.** En el debate electoral todos los estudiantes participan en un debate abierto similar al del escenario anterior para permitir a los estudiantes conocer los programas de cada candidato.
- **Elección de delegado.** En esta última fase, todos los estudiantes, incluido los candidatos, emiten su voto para escoger a su candidato favorito.

La realización de estas actividades debe desarrollarse en secuencia. Además, si en la última fase existe empate en la votación, el flujo de trabajo debe retornar a la segunda actividad para repetir el debate y realizar una nueva votación. Para gestionar ambos aspectos utilizaremos un motor de workflow similar al que se definió en el escenario de división colaborativa del trabajo. El listado 4.24 muestra el contenido de especificación de la lógica de secuenciamiento para el mismo expresada en XML:

Listado 4.24. Especificación de la lógica de secuenciamiento.

```
<workflow>
  <sequence>
   <while condition='nCandidatures @lt $minCandidatures'> ❶
    <activity name='MON.01. Presentacion de candidaturas'/>
   </while>
   <while condition='!validPoll'> ❷
    <sequence>
     <activity name='MON.02. Debate electoral'/>
     <activity name='MON.03. Eleccion de delegado'/>
    </sequence>
   </while>
  </sequence>
 </workflow>
```

Como puede apreciarse, el escenario comienza por una actividad de la que no se puede salir hasta que el número de candidaturas (*nCandidatures*) supere el mínimo de candidaturas admisible (*minCandidatures*, típicamente 2) (❶). Después, las actividades 2 y 3 son ejecutadas en secuencia y de forma iterativa hasta que la votación sea válida. Esto es, no acabe en empate (❷). Las variables *nCandidatures* y *validPoll* son dos de los indicadores que se mantendrán actualizados por las reglas del modelo de intervención. La figura 4.10 ilustra todo esto de forma gráfica.

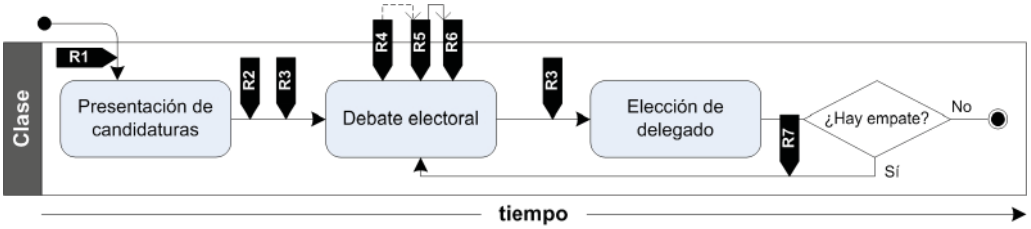

Figura 4.10. Flujo de trabajo instruccional del escenario de elección de delegados.

Diseño del modelo de integración

En las tres plantillas de actividad que hemos descrito en el apartado anterior se han definido varios servicios de las herramientas ENLACE cuyo uso describimos aquí:

- **CARDS.** Para dar soporte a la presentación de candidaturas se ha incluido en la primera actividad de este escenario un servicio de CARDS que permite crear fichas de tipo Candidatura. Cada candidato debe rellenar una ficha indicando su nombre y breve descripción del programa electoral.

- **LOR.** Los estudiantes pueden consultar las candidaturas creadas por los candidatos accediendo al repositorio de clase donde han sido almacenadas por éstos últimos. Para ello, la primera actividad también incluye un servicio de LOR que muestra el contenido de dicho repositorio.

- **CHAT.** La herramienta CHAT se incorpora en la segunda actividad del escenario para permitir a los estudiantes realizar el debate electoral. Cada vez que los estudiantes envían mensajes, esta herramienta emite un evento externo de tipo *chat.send.message* para avisar a la plataforma. Como veremos esto será aprovechado en el modelo de intervención. Para este servicio se han definido en esta actividad dos roles que permiten a la plataforma moderar el debate penalizando temporalmente sin permiso de palabra a aquellos estudiantes que consuman demasiado tiempo en sus intervenciones:

  – *Participante.* El rol participante concede al usuario que lo desempeña permiso de escritura y lectura de mensajes en el CHAT. Al inicio todos los estudiantes son participantes.

  – *Observador.* El rol de observador representa a aquellos usuarios que se encuentran penalizados y no pueden intervenir en el CHAT. Un observador puede leer mensajes pero no puede escribirlos. Cuando Pelican detecta que un participante habla demasiado le asigna temporalmente el rol de observador y después le vuelve asignar el rol participante, como explicaremos después.

- **AGORA.** La herramienta AGORA se utiliza en la tercera actividad para realizar la votación de elección de delegados. Esta será creada dinámicamente por la plataforma a partir de la especificación del modelo de intervención. Para detectar las situaciones de empate, AGORA emite el evento externo *agora.session.exists.draw*, que es atendido asimismo por las reglas de dicho modelo.

Diseño del modelo de intervención

El modelo de intervención para este escenario está formado por siete reglas, de las cuales dos (R1 y R3) se utilizan para gestionar el secuenciamiento de las actividades, el resto están relacionadas con las tareas de monitorización y control. A continuación describimos brevemente las mismas:

- **R1 Creación del motor de workflow.** Esta es la primera regla que es lanzada a ejecución cuando el profesor despliega el proyecto sobre el espacio de trabajo de clase. Su misión es crear el motor de workflow que controlara el secuenciamiento de las actividades. Como se ve en el listado 4.25, la regla está asociada al evento de despliegue de un proyecto y las condiciones restringen la ejecución al caso en que dicho proyecto sea el de elección de delegados. El script, muy similar al de la regla R2 del escenario de división colaborativa del trabajo, comienza cargando la especificación XML del repositorio (❶) y construyendo un motor (❷) para arrancarlo (❸) y almacenarlo en el contexto del profesor (❹).

Listado 4.25. Regla de creación del motor de workflow.

```
Evento: pelican.management.project.new
Disparador: event.project.projectTemplate.name=='Eleccion de delegados'
Script:
 def xmlWorkflow = common.callScript ('WKF.Read Workflow', ❶
                                      ['repository' : group.name,
                                       'learningObject' : 'Workflow']).
 def workflow = common.callScript ('WKF.Create Workflow', ❷
                                   ['xmlWorkflow' : xmlWorkflow,
                                    'project' : event.project,
                                    'group' : event.group])
 workflow.open () ❸
 def teacher = event.group.getActorBinding ('profesor').users[0]
 common.saveInContext (teacher, 'workflow', workflow) ❹
```

- **R2 Cálculo del indicador nCandidaturas.** La lógica de secuenciamiento, cuyo XML aparece representado en el listado 4.24, prescribe que la primera actividad no se debe abandonar hasta que al menos existan almacenadas en el repositorio de clase cierto número de candidaturas. En este escenario, el número de candidaturas es un indicador que será calculado cada vez que se intente transitar a la siguiente actividad para permitir al motor de workflow comprobar que tal condición de transición se satisface. En efecto, la regla R2, cuyo código se muestra en el listado 4.26, sirve a este propósito. Como puede apreciarse, la regla está vinculada al evento de transición de estado de la actividad y su disparador especifica que sólo será lanzada a ejecución cuando el estado de la misma pase a *pendiente*, indicando así su finalización. Para calcular este indicador, el script de adaptación recupera del repositorio la colección de identificadores de objetos de aprendizaje de tipo Candidatura invocando a un script de librería (❶). Luego cuenta el número de resultados obtenidos (❷) y lo almacena en el contexto del profesor (❸).

Listado 4.26. Regla para el cálculo de indicador nCandidaturas.

```
Evento: pelican.workspace.activity.updateState
Disparador: event.newActivity.activityTemplate.name=='Presentacion de ca...' &&
            event.newActivity.state=='PENDING'
Script:
 def activity = event.newActivity
 def group = activity.project.workspace.owner
 def loIds = common.callScript('INT.LOR.Get Learning Object Ids', ❶
                               ['repository' : group.name,
                                'metadata': ['lor.general.name': 'Candidatura']])
 def teacher = group.getActorBinding ('profesor').users[0]
 def nCandidatures = loIds.size () ❷
 common.saveInContext (teacher, 'nCandidatures', nCandidatures) ❸
```

- **R3 Transición de actividad.** La regla de transición de este escenario encierra un doble propósito. Por un lado se encarga de realizar las transiciones sobre el motor de workflow para secuenciar las actividades según lo establecido en la especificación mostrada en el listado 4.24. Por otro lado, una vez obtenida una nueva actividad, lleva a cabo una serie de tareas que dependen de cuál sea ésta para preparar su desarrollo por parte de los estudiantes. Como puede apreciarse en el listado 4.27, esta regla se encuentra asociada al evento de cambio de estado de la transición y su disparador indica que se lanzará a ejecución siempre que la actividad involucrada en el evento se trate de una de las asociadas al proyecto de este escenario. El script de adaptación de la regla comienza recuperando el motor de workflow del contexto de usuario del profesor, donde fue almacenado por la regla R1 (❶). Después realiza la transición hacia la siguiente actividad invocando al método next y pasándole como argumento la actividad en curso (❷). Hecho esto se vuelve a almacenar el motor de workflow en el contexto para salvaguardar su estado (❸). Entonces, se recupera el estado de la nueva actividad en curso (❹) y se entra en un bloque switch. Si la actividad nueva es la presentación de candidaturas quiere decir que las condiciones de transición asociadas a la primera actividad en la especificación de secuenciamiento no se han satisfecho. En ese caso, el tratamiento que se hace es advertir al usuario que forzó la transición de que no existen suficientes candidaturas en el repositorio como para proceder con la fase de debate electoral y restaurar el estado de la actividad a *en desarrollo* (❺). Si la nueva actividad es el debate electoral significa que se superó la primera actividad o que la votación acabó en empare. Como se ve en el listado 4.24 esto es controlado por el indicador *validPoll*. En la primera iteración la condición se satisface ya que al no existir esta variable en el contexto la expresión se hace falsa y por tanto su negación cierta. En cualquier caso, en esta situación se informa a los usuarios de que pueden empezar el debate y se guarda en el contexto el valor 0 para los indicadores *nMessages* en los contextos de usuario, cuyo uso explicaremos en la regla R4 (❻). Finalmente, si la actividad en curso es la votación, se crea una nueva sesión de votación en AGORA (❼) y se pone a cierto el indicador *validPoll*. La regla R7 ya se encargará de falsificarlo si se ha producido un empate en la misma.

Listado 4.27. Regla de transición de actividad.

```
Evento: pelican.workspace.activity.updateState
Disparador: event.newActivity.proyect.projectTemplate.name=='Eleccion de...' &&
            event.newActivity.state=='PENDING'
Script:
 def activity = event.newActivity
 def group = activity.project.workspace.owner
 def teacher = group.getActorBinding ('profesor').users[0]
 def workflow = common.loadFromContext (teacher, 'workflow') ❶
 workflow.next(activity) ❷
 common.saveInContext (teacher, 'workflow', workflow) ❸
 def newActivity = (workflow.state()[0]).name ❹
 switch (newActivity) {
   case 'Presentacion de candidaturas': ❺
     common.sendMessage (currentUser, 'No hay suficientes candidaturas')
     common.update (Activity, activity.id, 'state', new State ('DEVELOPMENT'))
     break;
   case 'Debate electoral': ❻
     for (user in group.users) {
       common.sendMessage (user, 'Ahora puedes empezar a debatir')
       common.saveInContext (user, 'nMessages', 0)
     }
     break;
   case 'Eleccion de delegado': ❼
     for (user in group.users)
       common.sendMessage (user, 'Ahora puedes votar a tu candidato')
       def candidates = common.callScript ('INT.LOR.Get Learning Object Ids',
                              ['repository' : group.name,
                               'metadata': ['lor.general.name': 'Candidatura']])
       common.callScript ('INT.AG.Create Session',
                          ['name' : 'Eleccion de delegados',
                           'moderator' : teacher,
                           'group' : group,
                           'type' : 'CARDID',
                           'candidates' : candidates,
                           'mode' : 'NO'])
      common.saveInContext (teacher, 'validPoll', true)
      break;
 }
```

- **R4 Calculo de indicador nMessages.** La regla anterior se encarga de introducir en el contexto de cada usuario un indicador *nMenssages* que contabiliza el número de mensajes escritos en el CHAT por cada usuario. Como puede verse en el listado 4.28, esta regla se dispara cada vez que un evento de envío de un nuevo mensaje (*chat.send.message*) es recibido desde la herramienta CHAT. El script asociado incrementa el número de mensajes y lo almacena en el contexto de usuario (❶).

Listado 4.28. Regla para el cálculo del indicador nMessages.

```
Evento: chat.send.message
Disparador: currentProyect.projectTemplate.name=='Eleccion de delegados'
Prioridad: 1
Script:
 def room = event.room
 def user = social.getUser (event.user)
 common.saveInContext (user, 'nMessages', ++nMessages) ❶
```

- **R5 Denegación del permiso de intervención en el debate.** Tal y como puede verse en las condiciones de disparo de esta regla, mostrada en el listado 4.29, cada vez que R4 es ejecutado, R5 también se ejecuta para comprobar si el número de mensajes emitido por el estudiante supera cierto valor umbral. Primero se recupera el indicador para cada usuario (❶) y el número total de mensajes (❷). Después se calcula el rango intercualtílico y si el valor del indicador supera 1,5 veces ese valor (❸), se le asigna el rol de observador (❹) y se arranca un cronómetro (❺).

Listado 4.29. Regla de denegación del permiso de intervención en el debate.

```
Evento: chat.send.message
Disparador: currentProyect.projectTemplate.name=='Eleccion de delegados'
Prioridad: 2
Script:
 def room = event.room
 def user = social.getUser (event.user)
 def group = social.getGroup (room)
 def activity = currentActivity
 def userNMessages = []
 def totalNMessages = 0
 for (aUser in group.users) {
   def aUserNMessages = common.loadFromContext (aUser, 'nMessages') ❶
   userNMessages.add (aUserNMessages)
   totalNMessages += aUserNMessages ❷
 }
 userNMessages.sort()
 def q1Index = (int) 0.25 * userNMessages.size
 def q1Value = userNMessages[q1Index]
 def q3Index = (int) 0.75 * userNMessages.size
 def q3Value = userNMessages[q3Index]
 def upperLimit = q3Value + 1.5 * Math.abs(q3Value - q1Value)
 if (totalNMessages > maxMessages) ❸
   if (nMessages > upperLimit) {
     def rObserver = activity.activityTemplate.getRole ('observador')
     collaboration.bindUserToRole (user, rObserver, activity) ❹
     common.startTimer (user,'ChatTimer',['room': room,'user': user, ❺
                        'activity': activity], chatLockedTime * 1000)
  }
```

- **R6 Concesión del permiso de intervención en el debate.** Transcurrido el tiempo de penalización configurado en el cronómetro que se arrancó en la regla R5 es necesario devolver al usuario el rol *participante*. En el listado 4.30 se muestra cómo es llevado esto a cabo. La regla está asociada al evento de finalización del cronometro y el disparador permite lanzar el script a ejecución siempre que la actividad del evento sea el debate electoral. El script comienza recuperando el rol participante (❶), lo asigna al usuario (❷) y finalmente restaura a 0 el valor del indicador *nMessages* (❸).

Listado 4.30. Regla de concesión del permiso de intervención en el debate.

```
Evento: pelican.core.timeOver
Disparador: event.activity.activityTemplate.name=='Debate electoral'
Script:
 def user = event.user
 def activity = event.activity
 def rParticipant = activity.activityTemplate.getRole ('participante') ❶
 collaboration.bindUserToRole (user, rParticipant, activity) ❷
 common.saveInContext (user, 'nMessages', 0) ❸
```

- **R7 Cálculo de indicador validPoll.** Cuando se produce un empate en la votación, AGORA emite un evento externo que es atendido por esta regla. Como se ve en el listado 4.31, el disparador permite que se ejecute el script cuando la actividad implicada sea la de elección de delegado. Éste por su parte, comienza poniendo a falso el indicador de validez de la votación (❶) y fuerza una transición en el motor para que se vuelva a la segunda actividad (❷). Después salva el motor de workflow en el contexto del profesor (❸) restaura el estado de las actividades 2 y 3 (❹) y envía un mensaje a los alumnos para que vuelvan al debate (❺).

Listado 4.31. Regla de concesión del permiso de intervención en el debate.

```
Evento: agora.session.exist.draw
Disparador: event.activity.activityTemplate.name=='Eleccion de delegado'
Script:
 def groupId = new Long (event.groupId)
 def group = social.getGroup (groupId)
 def teacher = group.getActorBinding ('profesor').users[0]
 common.saveInContext (teacher, 'validPoll', false) ❶
 def workflow = common.loadFromContext (teacher, 'workflow')
 workflow.next(currentActivity) ❷
 common.saveInContext (teacher, 'workflow', workflow) ❸
 def activity2 = currentProject.getActivity ('Debate electoral') ❹
 def activity3 = currentProject.getActivity ('Eleccion de delegados')
 common.update (Activity, activity2.id, 'state', new State ('DEVELOPMENT'))
 common.update (Activity, activity2.id, 'state', new State ('DEVELOPMENT'))
 for (user in group.users) ❺
   common.sendMessage (user, 'Hay empate. Por favor, vuelve a la actividad 2')
```

4.5.5. Estratificación social por evaluación de resultados

Tal y como se discutió en el capítulo 2, una de las primeras preocupaciones cuando se aborda el diseño de un escenario consiste en determinar el criterio de estratificación que se seguirá para formar los grupos. Algunos autores han afirmado que la mejor manera de llevar esto a cabo es buscar la heterogeneidad entre sus miembros mediante la realización de un test preliminar que ayude a caracterizar los conocimientos y estilos de aprendizaje de los alumnos de manera que se agrupen aquéllos con máximas diferencias [Jermann & Dillenbourg, 2002]. Inspirados en esta idea, hemos implantado este escenario en Pelican para ilustrar cómo las capacidades de intervención de la plataforma también pueden ser aplicadas para automatizar ciertas tareas de evaluación:

- **Descripción.** En este escenario el profesor comienza presentado un material a los estudiantes de una clase para que lo estudien individualmente y realicen una discusión sobre el mismo. Después, les somete a un test individual para obtener una puntuación de cada estudiante. Organizados éstos en un ranking en función de los resultados, se agrupa a los estudiantes por parejas juntando al primer estudiante de la primera mitad del ranking con el primero de la segunda mitad, al segundo con el segundo, y así sucesivamente. Por último, esta nueva agrupación es utilizada para realizar una actividad colaborativa por parejas. La hipótesis de partida es que esta estratificación maximizará la probabilidad de encontrar conflictos cognitivos entre los estudiantes

- **Contexto.** Esta experiencia se desarrolla en un contexto de educación formal con alumnos de enseñanza secundaria de edades comprendidas entre los 15 y 16 años. La experiencia comienza a nivel de clase y luego continúa con una actividad colaborativa realizada en parejas.

- **Dominio.** El dominio de conocimiento que hemos elegido para ilustrar el desarrollo de este escenario en nuestras explicaciones es sobre los principios generales de los circuitos y la corriente eléctrica. No obstante, esta especificación es fácilmente transladable a otro dominio siempre que sea posible elaborar un test de evaluación con respuestas únicas y precisas sobre el mismo para su corrección automática por parte de la plataforma.

- **Fases de desarrollo.** La realización de esta experiencia puede describirse en tres fases de desarrollo que aparecen representadas gráficamente en la figura 4.11 y que describimos brevemente a continuación:

  – *Presentación de la experiencia.* En la primera fase el profesor presenta la unidad didáctica que se va a abordar en la experiencia proporcionando a los alumnos referencias y material suficiente para que estudien.

  – *Desarrollo del test de estratificación.* En esta fase el profesor somete a los estudiantes a un test individual para evaluar su conocimientos previos y sus estilos de aprendizaje. Con estos resultados organiza a los alumnos en pares.

– *Realización de la actividad colaborativa.* Organizados en parejas, los estudiantes realizan una actividad colaborativa relacionada con el tema de la unidad didáctica.

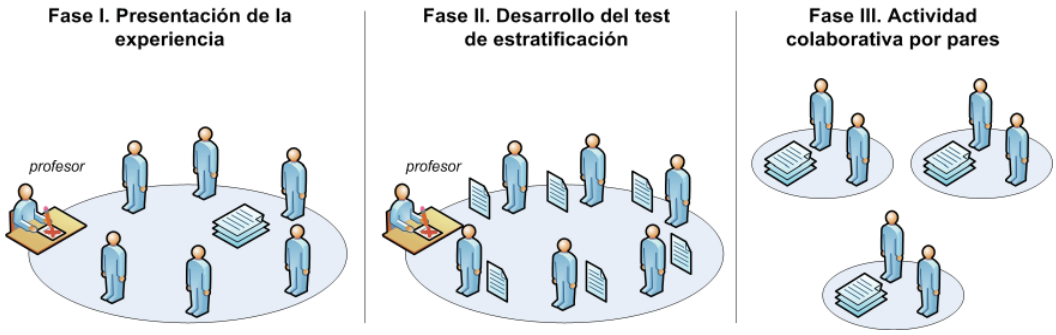

Figura 4.11. Fases de desarrollo del escenario de estratificación social por evaluación de resultados.

Diseño del modelo de sociedad

Para el desarrollo de este escenario haremos uso de los actores profesor y estudiante para representar a los dos tipos de usuarios implicados. Con respecto a los tipos de agrupaciones que son necesarias cabe destacar dos plantillas de grupo:

- **Clase.** La clase representa al colectivo de estudiantes donde se despliega la experiencia. En este escenario está formada por un profesor y una colección de estudiantes e incluye como plantilla de grupo hija a una colección de *pares*.
- **Par.** El par representa a un par de estudiantes desarrollando la última fase de este escenario. Su especificación indica que un par está formado exclusivamente por dos estudiantes.

Diseño del modelo de colaboración

De lo descrito en el apartado anterior se desprende que este escenario se desarrollo a lo largo de dos niveles sociales: la clase y el par. Para dar soporte a cada una de las fases descritas anteriormente es posible definir tres actividades colaborativas que se realizan de manera secuencial según es mostrado en el flujo de trabajo de la figura 4.12:

- **Ley de Ohm.** La actividad para introducir el tema de la corriente eléctrica hace una presentación de la ley de Ohm y proporciona a los estudiantes, a través de referencias Web, material didáctico relacionado.
- **Test sobre la ley de Ohm.** En esta actividad los estudiantes rellenan individualmente un test sobre la ley de Ohm que como veremos más adelante será corregido automáticamente por la plataforma para hacer la estratificación en pares.
- **Leyes de Kirchhoff.** La última actividad de este escenario, relacionada con las leyes de mallas y nudos del análisis de circuitos ha de realizarse por parejas.

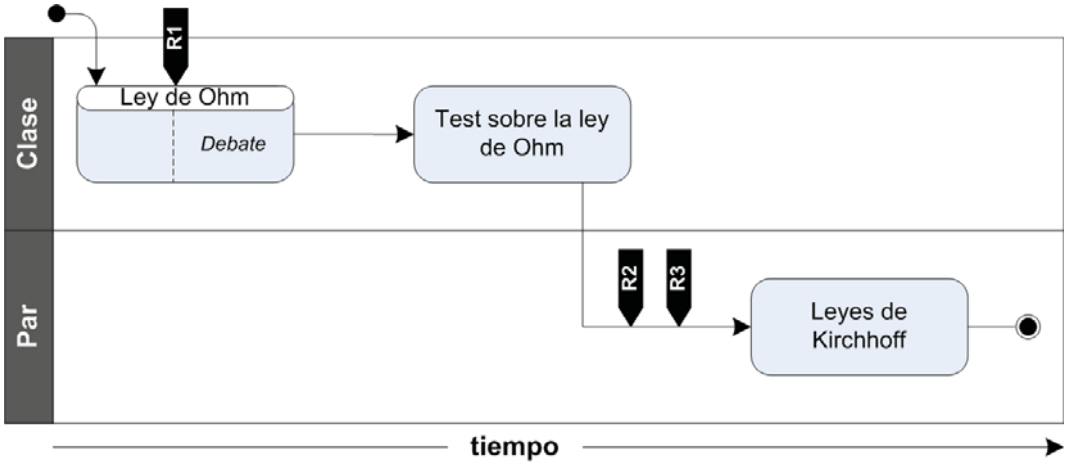

Figura 4.12. Flujo de trabajo instruccional del escenario de estratificación social por evaluación de resultados.

Diseño del modelo de integración

Para dar soporte al desarrollo de las actividades del flujo de instrucción, se proporcionan una serie de servicios de interacción que dan acceso a las siguientes herramientas del ecosistema tecnológico de ENLACE:

- **CHAT.** En la primera y última actividad de este escenario se proporciona un servicio de CHAT para permitir el debate entre los estudiantes. En las leyes de Kirchhoff el CHAT está accesible durante toda la actividad. Sin embargo, en la ley de Ohm, el servicio es activado automáticamente por el sistema de intervención cuando el profesor cambia el estado de la actividad de *inicio* a *en desarrollo*.
- **CARDS.** La herramienta CARDS permite a los estudiantes en la segunda actividad crear una ficha Test que se corresponde con el cuestionario que permite realizar la estratificación en grupos. Tanto la corrección como la formación de pares subsiguiente será realizada automáticamente a través de reglas políticas.
- **LOR.** En la segunda actividad los test individuales son almacenados en el repositorio individual del estudiante. En ella, hay definido un servicio de LOR para acceder al mismo. Al finalizar la actividad todos estos test son recopilados y almacenados en el repositorio del profesor para su procesamiento.

Diseño del modelo de intervención

El modelo de intervención incluye 3 reglas para cubrir los aspectos dinámicos (ver figura 4.12). Como puede verse, por claridad, éstas se encuentran comprometidas con el dominio de aplicación pero son fácilmente parametrizables para eliminar las dependencias literales (nombres de actividades, etc.) que provocan esta característica:

- **R1 Inicio de la sesión de debate.** Esta regla habilita el servicio de CHAT de la primera actividad cuando el profesor pone el estado de la misma a *en desarrollo*. Esto permite dedicar una primera parte de la misma al estudio individual del tema

abordado y una segunda para debatir colaborativamente sobre él. Como puede apreciarse en el listado 4.32, la regla está asociada al evento de cambio de estado de la actividad y el disparador prescribe que el script asociado sólo será ejecutado cuando la actividad corresponda con la primera de este escenario y el estado tras la transición sea *en desarrollo*. Por su parte el script comienza recuperando de la actividad el servicio de CHAT (❶) y utiliza la tarea *update* para habilitar el acceso al mismo desde los espacios de trabajo (❷).

Listado 4.32. Regla para el inicio de la sesión de debate.

```
Evento: pelican.workspace.activity.updateState
Disparador: event.newActivity.project.projectTemplate.name=='Ley de Ohm' &&
            event.newActivity.state=='DEVELOPMENT'
Script:
 def activity = event.newActivity
 def chatService = activity.getService('Discusion sobre la ley de Ohm') ❶
 common.update (Service, chatService.id, 'enabled', true) ❷
```

- **R2 Recolección de test de estudiantes.** La regla de recolección de test se ejecuta al finalizar la segunda actividad y se encarga de recuperar del repositorio privado de cada estudiante la ficha CARDS con el test individual y copiarla al repositorio del profesor. En el listado 4.33 puede verse que esta regla esta asociada a la regla de cambio de estado de la actividad y el disparador indica que la ejecución sólo se produce cuando la segunda actividad cambia al estado *pendiente*. El script invoca, para cada estudiante, al script de librería *Copy Learning Object* que toma como argumento el nombre de la ficha, el nombre del repositorio del estudiante y el profesor (❶).

Listado 4.33. Regla de recolección de test de estudiantes.

```
Evento: pelican.workspace.activity.updateState
Disparador: event.newActivity.project.projectTemplate.name=='Test sobre...' &&
            event.newActivity.state=='PENDING'
Prioridad: 1
Script:
 def activity = event['activity.new']
 def teacher = activity.getRoleBinding('profesor').users[0]
 def students = activity.getRoleBinding('estudiante').users
 for (student in students)
   common.callScript ('INT.LOR.Copy Learning Object', ❶
                      ['sourceRepository' : student.login,
                       'targetRepository' : teacher.login,
                       'learningObject' : 'TestOhm'])
```

- **R3 Formación de pares.** Tras de la regla R2, se ejecuta esta regla cuya misión es llevar a cabo la formación de los pares de estudiantes para realizar la actividad final. El listado 4.34 muestra el código de la misma. En él puede apreciarse cómo las condiciones de disparo son idénticas a la anterior a excepción de la prioridad

que fuerza a esta regla a ejecutarse en segundo lugar. El script asociado utiliza 3 funciones auxiliares. La función *getResponse* recibe como argumentos un nombre de un usuario y un nombre de repositorio y devuelve una lista con las respuestas de dicho usuario. La función *getAssessment* recibe las respuestas (correctas) del profesor y las de un alumno, las compara dos a dos para contabilizar el número de diferencias y devuelve como resultado un número entre 0 y 10 que indica la puntuación del alumno en el test. La función *orderByAssessments* recibe como argumento la lista de pares alumno–calificación y devuelve la lista de alumnos ordenada por orden decreciente de puntuación. Haciendo uso de estas tres funciones el script comienza recuperando las respuestas del profesor (❶). Después, para cada estudiante, obtiene sus respuesta (❷) y las contrasta con las del profesor (❸) anotando la puntuación de cada estudiante. Obtenidas todas las puntuaciones, construye el ranking de estudiantes (❹) y lo recorre hasta la mitad. En cada iteración del bucle, selecciona a un usuario de la primera mitad y a aquél que ocupa su misma posición relativa en la segunda mitad (❺). Con estos dos usuarios, y con el profesor forma un nuevo par. Una vez que éste está formado debe desplegarse la plantilla de actividad correspondiente a la última actividad del escenario sobre su espacio de trabajo del par. Para ello, se crea un nuevo proyecto y una actividad (❻) y se agregan con la tarea *addActivity*. Finalmente, se envía un mensaje a todos los miembros de la clase instándoles a que comiencen la actividad sobre las leyes de Kirchhoff.

Listado 4.34. Regla de formación de pares.

```
Evento: pelican.workspace.activity.updateState
Disparador: event.newActivity.project.projectTemplate.name=='Test sobre...' &&
            event.newActivity.state=='PENDING'
Prioridad: 2
Script:
 def activity = event.newActivity
 def project = activity.project
 def group = activity.project.workspace.owner
 def teacher = activity.getRoleBinding('profesor').users[0]
 def students = activity.getRoleBinding('estudiante').users
 def tResponse = getResponse (teacher, teacher.login)  ❶
 def sAssessments = [:]
 for (student in students) {
   def sResponse = getResponse (student, teacher.login) ❷
   def sAssessment = getAssessment (sResponse, tResponse) ❸
   sAssessments [student] = sAssessment
 }
 def ranking = orderByAssessments (sAssessments) ❹
 def activityTemplate = collaboration.getActivityTemplate ('Leyes de  Kirchhoff')
 def projectTemplate = project.projectTemplate
 def sActor = social.getActor ('estudiante')
 def tActor = social.getActor ('profesor')
 def groupTemplate = social.getGroupTemplate ('par')
```

```
def parentGroup = group
def middle = ranking.size / 2
def index = 0
while (index < middle) {
  def name = 'par ' + (index + 1)
  def pGroup = social.createGroup (name, name, groupTemplate, parentGroup)
  social.bindUserToActor (ranking[index], sActor, pGroup) ❺
  social.bindUserToActor (ranking[middle+index], sActor, pGroup)
  social.bindUserToActor (teacher, tActor, pGroup)
  def pProject = collaboration.createProject (projectTemplate, pGroup) ❻
  def pActivity = collaboration.createActivity (activityTemplate)
  collaboration.addActivity (pActivity, pProject)
  index = index + 1
}
for (user in group.users) ❼
  common.sendMessage (user, 'Ahora podeis desarrollar la actividad por pares')

def getResponse (user, repository) {
  def result = []
  def xmlTest = common.callScript ('INT.LOR.Get Card',
                                   ['repository' : repository,
                                    'name' : 'TestOhm',
                                    'metadata': ['lor.general.name': 'TestOhm',
                                                 'lor.general.author': user]])
  def test = tools.readXml (xmlTest)
  test.contents.field.value.collect() { result += it.text() }
  return result
}
def getAssessment (response, template) {
  def differences = 0
  def index = 0
  while (index < response.size) {
    if (response[index] != template[index])
      differences = differences + 1
    index = index + 1
  }
def orderByAssessments (assessments) {
  def users = [] + assessments.keySet()
  for (i = 0; i < users.size; i++)
    for (j = i; j < users.size; j++)
      if (assessments[users[j]] > assessments[users[i]]) {
         def auxUser =  users[j]
         users[j] = users[i]
         users[i] = auxUser
      }
  return users
}
```

4.6. Conclusiones

Como se ha podido comprobar a lo largo de este capítulo, el proyecto ENLACE ha constituido un excelente marco de pruebas donde poner a comprobar la viabilidad de la plataforma de aprendizaje proporcionando un rico ecosistema de herramientas externas de soporte a la interacción para poder llevar a cabo escenarios de aprendizaje colaborativo tanto en entornos controlados de ejemplo como en contextos pedagógicos reales de educación formal.

A lo largo de esta última sección describiremos los resultados y conclusiones generales extraídas de nuestras experiencias de evaluación con Pelican. Comenzaremos comentando la librería de scripts de adaptación que se presentó en la sección 4.3, continuaremos describiendo las conclusiones extraídas de las pruebas de evaluación de los cinco escenarios de ejemplo de la sección 4.5 y, finalmente, abordaremos los resultados de nuestra experimentación en los escenarios reales de ENLACE.

Resultados sobre las pruebas de la librería de scripts

La librería de scripts de adaptación que se presentó en la sección 4.3, es en realidad un resultado directo de nuestras primeras pruebas de evaluación para la implantación en Pelican de escenarios colaborativos complejos. En efecto, el diseño de tales escenarios y en particular la construcción del modelo de intervención, es una labor algo compleja y pesada sobre todo para diseñadores instruccionales con un bajo perfil tecnológico. Precisamente advirtiendo esta complejidad se decidió elaborar una librería de scripts que constituyesen procedimientos de transformación adaptativa prototípicos y probados para proporcionar una solución eficaz a los problemas que aparecen de forma recurrente en el diseño de escenarios colaborativos.

La forma en la que el modelo de intervención es construido en Pelican, según el cual para definir cada regla es preciso previamente especificar el script de adaptación asociado, fomenta el crecimiento incremental de esta librería. En efecto, aplicando los principios de bajo acoplamiento y cohesión modular descritos en las buenas prácticas de programación [Meyer, 1999 págs. 39-65] es posible convertir en un elemento reutilizable cada nuevo script de adaptación definido inicialmente para tratar reactivamente un evento en un escenario específico.

En este sentido cabe reflexionar acerca de cuál de todos los elementos que aparecen en el lenguaje para la especificación del modelo de intervención constituye la unidad de reutilización fundamental, si es el script de adaptación, la regla o incluso la política. A este respecto podemos concluir que existen tres grados de reutilización aplicados a sendos niveles de granularidad:

- **El script como elemento de reutilización.** Cada script de adaptación especificado de acuerdo a los principios descritos anteriormente constituye un elemento altamente reutilizable en diferentes contextos pedagógicos y por tanto es un buen candidato para pasar a formar parte de la librería. Esto es debido a su bajo nivel de granularidad ya que son independientes de dominio, resuelven pequeños

problemas de transformación típicamente recurrentes en multitud de escenarios colaborativos y resultan fácilmente adaptables por el uso de la parametrización.

- **La regla como elemento de reutilización.** Las reglas son elementos declarativos que vinculan la aplicación de un script de adaptación específico (con argumentos actuales para cada uno de sus parámetros) a un acontecimiento ocurrido durante el desarrollo de un flujo de trabajo instruccional. Esto reduce el nivel de reutilización de estos elementos con respecto al de los scripts ya que la posibilidad de aplicación de una regla queda limitada a que se encuentren las condiciones ambientales especificadas en el evento y el disparador de la misma. No obstante, tiene sentido considerar las reglas como unidad de reutilización cuando lo que se pretende aprovechar es precisamente el vínculo reactivo entre unas condiciones ambientales y la aplicación de un script.

- **La política como elemento de reutilización.** Como se explico en la subsección 3.5.1, las políticas son un elemento organizativo del lenguaje de intervención de Pelican. En principio, cada familia de reglas relacionadas semánticamente entre sí a través de un objetivo común deben ser definidas dentro de una misma política. El ejemplo por antonomasia de este caso consiste en definir todas las reglas que conforman el modelo de intervención de un escenario en una política pero pueden encontrarse otros ejemplos como el de políticas que describan las normas de ingreso a una determinada comunidad. En este caso, las políticas son elementos reutilizables de una situación a otra si lo que se pretende es reproducir un mismo modelo de comportamiento reactivo. Para fomentar la reutilización a este nivel las políticas permiten definir parámetros que son contempladas desde los scripts de adaptación de las reglas como valores constantes. Así por ejemplo, en las reglas de concesión de turnos del escenario de debate colaborativo, los tiempos de cada intervención procedían de parámetros especificados en la política.

La construcción reutilizable de scripts de adaptación es, a nuestro juicio, la forma más adecuada e inteligente de expresar modelos de intervención. A medida que se vayan desarrollando escenarios de aprendizaje colaborativo (u otras reglas relacionadas con distintos aspectos de la gestión del entorno) , la librería irá enriqueciéndose con nuevos scripts de adaptación. De hecho el propio lenguaje P# contempla una forma de dar soporte a este método de trabajo al proporcionar la tarea *callScript* que permite invocar un script desde otro mediante el paso de argumentos actuales en su llamada. Esta tarea hace posible llevar a cabo el desarrollo de nuevos scripts a partir del uso de otros ya existentes en la librería. En este sentido es posible distinguir dos tipos de scripts con sendos usos claramente diferenciados:

- **Scripts reutilizables.** Estos scripts son los elementos pertenecientes a la librería de scripts reutilizables que expresan transformaciones prototípicas como procedimientos parametrizados.

- **Scripts de invocación.** Los scripts de invocación, por el contrario son aquellos vinculados a las reglas y que se encargan de ejecutar una lógica transaccional que invoca a varios scripts reutilizables con argumentos actuales.

Resultados sobre las pruebas de los escenarios de ejemplo

Las pruebas realizadas con la implantación en Pelican de los cinco escenarios de ejemplo presentados en la sección 4.4, pretenden poner de manifiesto que las capacidades de la plataforma permiten llevar a cabo el diseño de escenarios de aprendizaje colaborativo. En este diseño, el subsistema social se encarga de dar soporte al entramado de grupos, actores y usuarios de la comunidad virtual de aprendizaje donde se desarrollan las experiencias. El subsistema de colaboración organiza el trabajo colaborativo de los estudiantes en términos de proyectos y actividades y proporciona una infraestructura jerárquica de espacios de trabajo donde circunscribir el desarrollo de los mismos. El subsistema de integración por su parte, ofrece mecanismos para la incorporación de servicios de soporte a la interacción colaborativa dentro de la plataforma y hace posible la orquestación de los mismos de acuerdo a protocolos definidos declarativamente para ofrecer un alto nivel de interoperabilidad que confiera a los estudiantes una sensación de continuidad en su uso. Y finalmente, el subsistema de intervención permite adaptar progresivamente el entorno de soporte (tanto la plataforma como las herramientas externas) para alinearlo con los requerimientos evolutivos del flujo de trabajo instruccional asociado al escenario.

Precisamente en este sentido, cada uno de los escenarios que describimos tenia como propósito centrarse en ilustrar diferentes aspectos del diseño. A continuación repasamos brevemente los mismos:

- **Procedimiento Jigsaw.** El objetivo de este escenario es ilustrar cómo la plataforma Pelican puede dar soporte a experiencias colaborativas caracterizadas por un entramado social inherentemente dinámico. A este respecto, el subsistema de intervención proporciona la infraestructura de tipos de grupos y actores necesaria para cubrir los aspectos estáticos de soporte mientras que el subsistema de intervención permite aplicar las transformaciones adaptativas necesarias sobre dicho entramado en el momento exacto en que lo requería el flujo instruccional para permitir continuar con el desarrollo de la experiencia. Muchos de los entornos de aprendizaje [Moodle] [Blackboard] [Alf] no permiten articular este tipo de cambios dinámicos sobre la estructura de la comunidad virtual, restringiendo el desarrollo de los escenarios a contextos sociales estáticos y forzando a parar el desarrollo para realizar manualmente cualquier tipo de alteración.

- **División colaborativa del trabajo.** Este escenario se centra en ilustrar cómo incluso el propio flujo de trabajo instruccional de un escenario puede alterarse a sí mismo mediante la conjugación de las capacidades del subsistema de colaboración y de intervención. El primero se utiliza para dar soporte a la definición del trabajo mientras que el segundo se encarga de articular el secuenciamiento de las actividades y de alterarlo para atender dinámicamente a la descomposición colaborativa determinada por los estudiantes durante el desarrollo de la experiencia. Esto constituye otra diferencia esencial con respecto a otras aproximaciones similares [IMS–SS] [Vantroys & Peter, 2003] que congelan desde el principio la lógica de secuenciamiento sin posibilidad de alterarla a tenor de acontecimientos.

- **Debate colaborativo.** El escenario de debate colaborativo muestra las capacidades de los mecanismos de integración proporcionados por Pelican para integrar herramientas externas en el entorno a un alto nivel de interoperabilidad sin comprometer el desarrollo de las mismas. En particular, aquí se ilustra cómo la plataforma da soporte al principio de integración orientado a roles en virtud del cual el control de la concesión de turnos en el CHAT puede ser delegado en Pelican y vinculado estrechamente así a la evolución del flujo de trabajo instruccional del escenario. La gran ventaja que pone de manifiesto este ejemplo es que gracias a las facilidades del subsistema de intervención es posible orquestar protocolos de integración expresados sobre las interfaces de usuario de Pelican sin obligar a los desarrolladores de herramientas externas a acogerse a ningún modelo arquitectónico de integración como sí ocurre en [IMS–TI] [OSGA] [Bote – Lorenzo et al.,2004] y [OSGi]. Todo ello es además conseguido sin sacrificar el nivel de interoperabilidad alcanzado.

- **Elección de delegados.** En la elección de delegados se demuestra cómo las capacidades proporcionadas por el subsistema de intervención permiten definir en Pelican un modelo de monitorización para calcular dinámicamente una colección de indicadores y definir reglas de control para intervenir puntualmente en el escenario cuando se alcancen ciertas condiciones sobre los mismos. Aunque el ejemplo de este escenario es pretendidamente sencillo nada impide, en principio, implantar uno más realista sobre todo si se delega el cálculo de indicadores a herramientas externas de monitorización. En este sentido, el subsistema de intervención facilita la integración de herramientas de análisis tales como [Barros & Verdejo, 2000].

- **Estratificación social por evaluación de resultados.** Este último escenario pretende demostrar que con los elementos del subsistema de intervención es posible automatizar muchas de las tareas que aparecen relacionadas con los procesos de evaluación de las experiencias colaborativas. En el ejemplo que aquí presentamos, tal automatización consiste en la recolección de una serie de test de los repositorios privados de cada estudiante y en su subsiguiente corrección automática para poder llevar a cabo una estratificación social que busca la formación de grupos con un alto nivel de heterogeneidad. En este sentido, Pelican también supone una ventaja al permitir a los administradores y diseñadores definir scripts para automatizar tareas de evaluación que por lo general resultan pesadas y tediosas.

Como se desprende de todos los ejemplos anteriores la especificación de escenarios requiere del diseño de cuatro modelos soportados por sendos subsistemas. El modelo de sociedad prescribe la estructura de grupos y tipos de usuarios de la que es necesario disponer para desarrollar la experiencia. El modelo de colaboración especifica el trabajo desarrollado por los estudiantes. El modelo de integración completa esta especificación indicando la colección de servicios de soporte a la colaboración que es necesario proporcionar en cada actividad. Y el modelo de intervención se encarga de capturar los aspectos dinámicos del escenario. En este sentido puede apreciarse cómo los tres primeros modelos, y por ende sus subsistemas asociados, dan cobertu-

ra vertical a parte de los requerimientos de un escenario de aprendizaje colaborativo. Sin embargo, el subsistema de intervención se aplica transversalmente en la especificación de un escenario para cubrir los requerimientos de intervención dinámica necesarios en cualquiera de los otros tres modelos.

Este subsistema, que es parte esencial de nuestra propuesta de tesis por su valor diferencial con respecto a otros entornos virtuales de aprendizaje tiene, no obstante una complejidad inherente que puede tal vez ser suavizada por el uso de la librería de scripts de adaptación discutida en la sección 4.3, (otras aproximaciones para la simplificación del diseño de escenarios serán discutidas en el capítulo siguiente de este trabajo).

Resultados sobre las pruebas de los escenarios reales

De las pruebas descritas en este capítulo parece deducirse que la propuesta que hacemos en este trabajo constituye una solución teórica aceptable para llevar a cabo la especificación computable completa de escenarios de aprendizaje colaborativo. No obstante, aún es necesario demostrar la viabilidad de la plataforma en situaciones reales de uso.

En este sentido, el proyecto ENLACE ha proporcionado un contexto de evaluación perfecto para desarrollar con Pelican – y con el resto de las herramientas del ecosistema tecnológico – diversos escenarios inmersos en un contexto de educación formal con alumnos de enseñanza secundaria obligatoria. Todos estos escenarios, que han sido descritos en la sección 4.4, han servido para ir refinando el desarrollo de la plataforma hacia los requerimientos de la situaciones reales y para poner a prueba el uso de la misma en entorno de explotación concurrentes.

5 Conclusiones y líneas futuras

A lo largo de esta tesis hemos presentado una plataforma de aprendizaje colaborativo para dar soporte a todas las tareas de diseño y desarrollo circunscritas a este tipo de experiencias instruccionales. En concreto, nuestra propuesta se articula a partir de cuatro subsistemas. El subsistema social se encarga de dar soporte a la estructuración social vinculada al desarrollo de experiencias colaborativas. El subsistema de colaboración permite definir escenarios de aprendizaje basados en proyectos y actividades para desplegarlos posteriormente sobre el entramado de espacios de trabajo. El subsistema de integración permite a herramientas externas integrarse fácilmente a dichos espacios a distintos niveles de interoperabilidad. Finalmente, el subsistema de intervención hace posible la captura de todos los aspectos de carácter dinámico que aparecen inherentemente vinculados a la mayoría de los escenarios de aprendizaje colaborativo.

Sin embargo, cabe preguntarse si con nuestro trabajo hemos cubierto los objetivos que se plantearon al inicio de esta tesis. En este sentido, es preciso formular preguntas tales como ¿Son suficientes las capacidades proporcionadas por la plataforma para el desarrollo de cualquier tipo de escenario de aprendizaje colaborativo independientemente de la complejidad de sus requerimientos? En otro caso, ¿Cuál es el alcance al que da soporte la plataforma? A su vez, también cabe preguntarse qué nuevas líneas de investigación futuras quedan abiertas como consecuencia del desarrollo de este trabajo. En este capítulo final trataremos de dar respuesta, a modo de conclusión a estas y otras cuestiones.

5.1. Introducción

En este trabajo de tesis hemos pretendido dar una solución para la especificación completa, formal y computable de escenarios de aprendizaje colaborativo. En esta línea, en el capítulo 2 se ha hecho una revisión multidisciplinar de todas las aportaciones relevantes realizadas dentro de la psicología, la pedagogía y la computación, en particular circunscritas al campo del CSCL.

En este sentido, para dar soporte computacional a todos los problemas de diseño y desarrollo relacionados con la puesta en práctica de escenarios de aprendizaje colaborativo, se ha construido Pelican, una plataforma de aprendizaje centrada en la actividad colaborativa cuya descripción arquitectónica y funcional se ha abordado en el capítulo 3. Cada uno de los cuatro subsistemas constituyentes se encargan de diferentes aspectos relacionados con la especificación de los escenarios de aprendizaje. El subsistema social proporciona el entramado de grupos necesario para constituir la comunidad virtual donde se desarrollan las experiencias colaborativas. El subsistema de colaboración permite organizar el trabajo de los estudiantes en términos de proyectos y actividades y ofrece la infraestructura de espacios de trabajo necesaria para su posterior despliegue. El subsistema de integración permite a herramientas externas integrarse con Pelican a bajo coste de desarrollo y obteniendo un alto nivel de interoperabilidad. Y finalmente, el subsistema de intervención captura todos los aspectos dinámicos inherentemente vinculados al desarrollo de las experiencias instruccionales colaborativas.

El capítulo 4 por su parte, ha presentado una revisión de los esfuerzos de desarrollo llevados a cabo dentro del proyecto ENLACE para poner a prueba la viabilidad tecnológica de la plataforma tanto en situaciones controladas de ejemplo como en contextos pedagógicos reales con múltiples estudiantes accediendo concurrentemente a ésta. Asimismo se han discutido allí brevemente las conclusiones de dichas pruebas.

Sin embargo, alcanzado este punto cabe preguntarse si nuestra propuesta tecnológica ha alcanzado los objetivos que quedaron establecidos en el capítulo primero de este trabajo. En este sentido, comenzaremos, en la siguiente sección haciendo una revisión de las cualidades de la plataforma Pelican, destacando sus valores diferenciales con respecto a otras propuestas existentes y señalando algunas mejoras que podrían ser objeto de estudio para versiones futuras. Finalmente, discutiremos si los mecanismos de soporte proporcionados por Pelican dan cobertura a los procesos identificados en el capítulo 2 (estratificación social, colaboración y cooperación, monitorización y control y evaluación).

De manera complementaria, otra de las preocupaciones que será tratada en la sección 5.3 está relacionada con la colección de nuevas líneas de investigación cuya germinación se encuentra centrada en el trabajo de esta tesis. En este sentido, haremos una revisión de las actividades de investigación previstas para los próximos años que tienen como objetivo ampliar y mejorar el soporte computacional a la especificación y desarrollo de escenarios de aprendizaje colaborativo.

5.2. Conclusiones

Pelican es una plataforma de aprendizaje que, como vimos en los capítulos anteriores, tiene como propósito dar soporte al diseño y desarrollo de las experiencias instruccionales colaborativas. Los actores involucrados en estas experiencias, a través del uso de Pelican, perciben a la plataforma desde tres perspectivas diferentes:

- **Pelican como un entorno virtual de aprendizaje.** La plataforma Pelican constituye un entorno virtual de aprendizaje donde los estudiantes, profesores y monitores se reúnen para desarrollar experiencias colaborativas de forma potencialmente telemática tanto síncrona como asíncronamente. La diferencia fundamental que caracteriza a Pelican con respecto a otros entornos virtuales tales como [Blackboard] [Moodle] o [Alf] radica precisamente en tres factores principales:

  – *Plataforma centrada en la actividad.* Como se ha podido comprobar a lo largo del capítulo 3, la plataforma se encuentra centrada en la actividad colaborativa. Es decir, para desarrollar cualquier tipo de proceso formativo en Pelican, el diseñador debe definir una familia de actividades dentro de un proyecto y, opcionalmente, secuenciarlas de acuerdo a un flujo de trabajo instruccional estructurado. Esta es una diferencia filosófica esencial con respecto al resto de las plataformas que han sido discutidas en el capítulo 2, puesto que mientras que en ellas se interpreta que el soporte al aprendizaje se realiza a través de la provisión de una colección de recursos y servicios formativos, e incluso a veces se da soporte al desarrollo de tareas y actividades, aquí se hace de manera normativa implicando al estudiante en un contexto realista de experimentación colaborativa, de acuerdo a los cánones del paradigma de aprendizaje.

  – *Plataforma orientada a la colaboración.* En este mismo sentido, la plataforma de aprendizaje está fuertemente orientada a la colaboración. En Pelican, todas las experiencias formativas son eminentemente colaborativas y la plataforma proporciona los elementos y mecanismos tecnológicos oportunos y necesarios para que los sucesos de esta naturaleza (interacción colaborativa, negociación, coordinación, etc.) tengan lugar. En efecto, los espacios de trabajo dan forma al entorno virtual para que un determinado contexto social se reúna a trabajar conjuntamente mientras que las herramientas externas desarrolladas por terceras partes e integradas, potencialmente, a un elevado nivel de interoperabilidad ofrecen servicios, muchas veces específicos de dominio y de la experiencia, para dar soporte a la colaboración. El resto de entornos virtuales de aprendizaje suelen limitarse por el contrario a proporcionar, marginalmente, una colección de herramientas empotradas y no adaptables de interacción tales como repositorios de recursos compartidos o foros de discusión, como valor añadido para permitir poner en contacto a los estudiantes.

  – *Plataforma dirigida por la adaptación continua.* El tercer elemento diferencial de la plataforma Pelican con respecto a otras propuestas es su capacidad para adaptar, tanto el propio entorno de la plataforma como el ecosistema tecnoló-

gico integrado en ella, a los requerimientos potencialmente cambiantes de los flujos de trabajo instruccional propios de cada escenario. Como se comprobó extensamente en la sección 4.5, la mayoría de las experiencias formativas de carácter colaborativo requieren realizar determinadas transformaciones bajo demanda según transcurre el desarrollo de las mismas para poder permitir su evolución. Por ejemplo, en el escenario de Jigsaw descrito en la subsección 4.5.1, tras la realización de la actividad experta, es preciso reorganizar a los estudiantes para formar los equipos de trabajo. Similarmente, en el escenario de división colaborativa del trabajo del la subsección 4.5.2, el flujo de trabajo instruccional no puede continuar, alcanzada la fase II, hasta que se procese el plan de descomposición del trabajo de los estudiantes para crear nuevos proyectos, actividades y subgrupos encargados de su desarrollo. En el resto de los entornos de aprendizaje esta adaptación debe ser realizada manualmente por profesores o administradores y comúnmente requiere paralizar el transcurso de la experiencia. Las capacidades adaptativas de Pelican por el contrario, permiten planificar la ejecución de las intervenciones para garantizar continuidad y fluidez en el desarrollo de la misma.

- **Pelican como una plataforma de integración.** Pelican constituye también una plataforma de integración orientada, naturalmente, al desarrollo de experiencias de aprendizaje. A este respecto, el subsistema de integración, proporciona cuatro mecanismos para incorporar herramientas en la plataforma a distinto nivel de compromiso tecnológico con la misma. En el nivel más bajo (integración ligera), servicios accesibles desde Internet tales como [GoogleMaps] o [YouTube] pueden integrarse en Pelican de manera completamente agnóstica sin necesidad de desarrollar ningún tipo de adaptador ni modificación en la plataforma. Pero además los desarrolladores de herramientas externas que deseen alcanzar un mayor nivel de cohesión e interoperabilidad con la plataforma pueden hacerlo a través del resto de mecanismos proporcionados por la misma. Esto permite a las herramientas centrarse en proporcionar buenos servicios para el soporte al desarrollo de los escenarios delegando en Pelican aspectos de gestiones tales como el control de las sesiones de trabajo de los estudiantes, la organización de éstos en grupos desempeñando ciertas responsabilidades sociales, el control del flujo de trabajo instruccional, la administración de los roles implicados en el desarrollo de cada actividad o el contexto que caracteriza el estado en curso de cada estudiante dentro de la plataforma. La mayoría de las propuestas de integración a este respecto, sin embargo [IMS–TI] [OGSA] [OSGi] [Bote–Lorenzo et al., 2004], se limitan a definir una arquitectura rígida que impone unas fuertes restricciones tecnológicas para integrar nuevos servicios. Estas propuestas tampoco ofrecen un modelo de información tan completo como el de Pelican para contextualizar, instruccionalmente, el uso de las herramientas.

- **Pelican como una herramienta de autor para el diseño de escenarios.** Las prestaciones de la plataforma no se limitan a proporcionar un entorno para el desarrollo de experiencias colaborativas y una colección de mecanismos para la integración

de herramientas externas. Pelican también constituye una herramienta de autor que permite a los diseñadores instruccionales definir los escenarios de aprendizaje donde se van a desarrollar dichas experiencias. Esta especificación se lleva a cabo definiendo cuatro modelos de diseño complementarios. La plataforma proporciona a este respecto sendos lenguajes de modelado que confieren cierto nivel de flexibilidad para describir los escenarios:

– *Modelo de sociedad.* El modelo de sociedad describe la infraestructura de grupos y tipos de usuarios, o actores, que aparecen implicados en el desarrollo de la experiencia que se está describiendo. El lenguaje de modelado social de Pelican permite llevar a cabo esta especificación de forma sencilla y completamente visual a través del uso de sus interfaces Web de diseño. Esta es una diferencia esencial con respecto al resto de entornos de aprendizaje donde no es posible definir tipos de agrupaciones y caracterizarlos en términos de su estructura y composición interna sino, en todo caso, determinar el tamaño de los mismos e indicar los usuarios que formarán cada uno de los colectivos que participarán en la experiencia. En Pelican, sin embargo, la descripción de todo escenario requiere, independientemente del contexto donde se desarrolle, indicar la infraestructura social requerida para su realización.

– *Modelo de colaboración.* El lenguaje de modelado de la colaboración permite a los diseñadores instruccionales definir el trabajo colaborativo que deberán llevar a cabo los estudiantes a lo largo de la experiencia y organizarlo en términos de proyectos y actividades de aprendizaje. Este lenguaje, también sencillo y visual, es pretendidamente simple y genérico para evitar sacrificar la flexibilidad expresiva de la plataforma. Nótese que muchas de las capacidades del mismo son conferidas por extensión desde el modelo de integración e intervención. Por ejemplo, a diferencia de como ocurre en otros sistemas como en [Vantroys & Peter, 2003] o [Miao et al., 2005], el modelo de colaboración no da soporte directo a la definición de un flujo de trabajo instruccional que defina el secuenciamiento de las actividades. En su lugar, en Pelican deben definirse elementos de intervención para dar soporte a este requerimiento. Esta característica de la plataforma, lejos de ser un inconveniente, supone una ventaja ya que no se limita a dar soporte a experiencias colaborativas caracterizadas por una fuerte estructuración sino que, adicionalmente, es posible articular escenarios más abiertos donde el orden de desarrollo de las actividades no quede prescrito en tiempo de diseño.

– *Modelo de integración.* El modelo de integración puede ser considerado en cierto sentido una extensión del modelo de colaboración encargado de proporcionar los elementos necesarios para dar soporte a la especificación de servicios de interacción colaborativa dentro de las actividades y espacios de trabajo de la plataforma. Estos servicios constituyen la definición de un uso contextualizado de una herramienta externa para una situación colaborativa específica. Además de los servicios, el lenguaje del modelo de integración utiliza los roles para caracterizar la responsabilidad colaborativa de los usuarios en

cada actividad y permitir proporcionar a las herramientas información acerca de cuál debe ser su comportamiento en función de dicho rol y del estado de la plataforma durante el desarrollo de la experiencia. A diferencia de como ocurre en otras especificaciones, como en [IMS–LD], la ventaja de Pelican en este sentido es que permite que la asignación de roles se modifique dinámicamente a lo largo del tiempo, bien sea de forma manual o automática, a través de las facilidades del modelo de intervención. En cualquier caso, esta característica permite a la plataforma tomar control para orquestar la interacción colaborativa que tiene lugar en el seno de las herramientas externas de soporte.

– *Modelo de intervención.* En este modelo se describen las intervenciones automáticas que serán llevadas a cabo por la plataforma sobre el resto de los otros tres modelos para adaptar el entorno puntualmente a las necesidades del flujo de trabajo instruccional y siempre en respuesta a los eventos disparados desde éste. En este sentido, cabe decir que este lenguaje se encuentra dirigido por eventos, está basado en reglas políticas y utiliza el lenguaje de dominio P# para la definición algorítmica de las transformaciones. La aplicación de las especificaciones a este nivel se realiza transversalmente a los otros tres modelos ya que aquí se capturan los aspectos dinámicos del escenario tanto en lo relacionado con la dimensión social como con la dimensión colaborativa y su extensión en la integración de herramientas externas. Esta es, sin lugar a dudas, una de las principales aportaciones de nuestro trabajo de tesis con respecto al resto de propuestas tecnológicas que fueron descritas en el capítulo 2. En efecto, las capacidades del modelo de intervención permiten dar continuidad en el desarrollo de las experiencias colaborativas al intercalar entre las actividades del flujo de trabajo instruccional la ejecución automática de scripts de adaptación que realizan tareas de administración, transformación y reconfiguración sobre el escenario sin necesidad de parar el desarrollo del mismo.

Si bien todas estas características ponen de manifiesto las ventajas del uso de la plataforma Pelican en el desarrollo de experiencias instruccionales de carácter colaborativo, lo cierto es que también es posible encontrar, dentro de su arquitectura, algunas mejoras que son dignas de mención. A continuación destacamos, para cada uno de los cuatro subsistemas constituyentes, aquéllas que han sido advertidas a lo largo del desarrollo de nuestras pruebas.

- **Subsistema social.** Este subsistema permite dar soporte a la estructuración social requerida para la mayoría de las experiencias de aprendizaje colaborativo. Sin embargo podemos encontrar dos inconvenientes fundamentales en el mismo:

  – *Limitación en la reutilización de plantillas de grupo.* Hemos de advertir una característica de diseño del subsistema social que va claramente en contra de la reutilización de los modelos de sociedad. Toda plantilla de grupo se inscribe en un contexto social de uso específico caracterizado por la relación de dependencia parento–filial establecida entre su plantilla padre y sus plantillas hijas. Por ejemplo, la *clase* es hija del *colegio* y padre de los *grupos de trabajo*.

Si bien esto es una decisión de diseño para forzar a mantener y respetar la estructura de la comunidad virtual subyacente, a la vez impide la definición de plantillas *libres* que puedan ser ubicadas en distintas localizaciones dentro de la estructura social. Por ejemplo, la plantilla *Par* como estructura organizativa (un par de estudiantes) podría ser requerida por dos escenarios distintos pero utilizada en diferentes puntos de la jerarquía social: el primero como manera de descomponer cada *grupo* en pares y el segundo para formar pares de trabajo dentro de la *clase*. Al mismo tiempo, al definir el modelo de sociedad para un escenario, es complicado extraer ciertas plantillas de grupo requeridas aislándolas de aquellas que no lo son dentro de la jerarquía social debido, precisamente, a las dependencias parento–filiales. Aunque este problema fue advertido en las fases iniciales de desarrollo de la plataforma, se decidió no alterar el diseño original del subsistema en aras a mantener sencillo el modelo de objetos del mismo y las interfaces de diseño Web para dar soporte a la definición de los aspectos sociales.

– *Limitación en la gestión simultánea de múltiples comunidades.* Aunque a lo largo de este trabajo hemos hablado de la posibilidad de mantener en Pelican varias comunidades virtuales asociadas al mismo o distintos modelos de sociedad, lo cierto es que para hacer esto posible es necesario crear una superestructura organizativa padre – una plantilla de grupo – que las contenga a todas ellas. Volviendo al ejemplo de la figura 3.2, para mantener la comunidad universitaria *UNED*, y los colegios *Diego Velázquez* y *Antonio Machado* de manera simultanea habría que crear una plantilla de grupo raíz constituida por plantillas de grupo *Universidad* y Colegio lo que permitiría luego crear las comunidades citadas. Todo esto está motivado porque el subsistema social no es capaz de manejar varias estructuras jerárquicas sociales. Esta es una mejora que será abordada en sucesivas versiones de la plataforma.

- **Colaboración.** El subsistema de colaboración permite articular la definición y posterior desarrollo de experiencias de aprendizaje colaborativo mediante el despliegue selectivo de actividades de aprendizaje sobre los espacios de trabajo de la plataforma. En este sentido podemos destacar los siguientes inconvenientes de la solución actual:

  – *Limitación en la especificación de los niveles de despliegue de actividades.* A la hora de especificar un modelo de colaboración, los diseñadores instruccionales pueden definir la familia de plantillas de actividad que forman parte de la experiencia e incluso, con ayuda el subsistema de intervención, prescribir un flujo de trabajo instruccional para secuencial el desarrollo de las mismas. Sin embargo, lo que no es posible prescribir dentro de la especificación, es el nivel social en el que cada plantilla de actividad debe ser desarrollada. Esto forma parte del conocimiento implícito que los profesores deben tener sobre la especificación a la hora de realizar el despliegue. Para solucionar este problema se requiere alinear el modelo de colaboración con el modelo de sociedad del escenario, lo cual pasa por dos posibilidades: o bien asociar una plantilla de

grupo a cada plantilla de actividad o, alternativamente, vincular a las plantillas de actividad del escenario un número ordinal relativo que indique el nivel al que éstas deben ser desplegadas, dentro de la estructura jerárquica del modelo de sociedad. En este segundo caso, puede aplicarse el despliegue de un proyecto de forma completa con un sólo golpe de ratón, indicando el nivel superior al que debe desplegarse la especificación lo que ofrece más flexibilidad en el desarrollo de la experiencia.

– *Limitación en la definición de actividades individuales.* Desde las interfaces Web del subsistema de colaboración no es posible desplegar, manualmente, actividades sobre los espacios de trabajo privados mantenidos por la plataforma sino solamente sobre los espacios compartidos. Esto hace imposible definir actividades para que se desarrollen de manera individual. Esta limitación esta presente, no obstante, solamente en la interfaz ya que el modelo de objetos del subsistema de colaboración si lo permite. De hecho, mediante el uso de tareas P# este tipo de despliegue sobre espacios de trabajo privados si es posible. En las siguientes versiones de la plataforma trabajaremos para solucionar este inconveniente.

- **Integración.** El subsistema de integración permite completar la especificación del modelo de colaboración mediante la definición de servicios de soporte a la colaboración proporcionados por herramientas externas. En este aspecto poder destacar dos mejoras:

  – *Limitación en la integración de servicios.* A diferencia de otras propuestas donde la integración se hace a nivel funcional incorporando capacidades a un entorno de trabajo [OSGi] o permitiendo la composición orquestal de las mismas de forma declarativa [BPEL4WS], en Pelican la integración consiste en la provisión de un punto de acceso a otras herramientas con su propia interfaz y funcionamiento. Esto es una característica definitoria de este subsistema ya que se persigue minimizar los esfuerzos de desarrollo de herramientas de soporte y maximizar la reutilización de aquéllas existentes en Internet. Sin embargo esto puede suponer una limitación por un doble motivo. En primer lugar porque el nivel de adaptación que se consigue con las propuestas de integración citadas es mucho más fino que el alcanzado con nuestra propuesta, y en segundo lugar, porque, a menudo, las necesidades de soporte a la interacción colaborativa son muy específicas del escenario en cuestión y resulta complicado encontrar herramientas para ello, más allá de las típicos servicios de carácter genérico (foros de discusión, repositorios de recursos, etc.). Esto requiere, en la práctica, implementar, o adaptar programáticamente versiones de herramientas existentes, lo que convierte a la reutilización en algo utópico.

  – *Limitación en el registro dinámico de herramientas.* Por otra parte, el registro de las herramientas sobre las cuales es posible definir servicios en Pelican es una labor que, como se discutirá en el apéndice B de este trabajo, se realiza en frío mediante la edición de un fichero de configuración, lo cual va en contra de

la tendencia general de la plataforma ya que el resto de elementos se definen en la misma a través de las interfaces Web (actores, plantillas de grupo, plantillas de proyecto y plantillas de actividad, etc.). Para futuras versiones este es un aspecto que podría ser mejorado e incluso contemplar la posibilidad de que el descubrimiento de servicios pueda ser realizado de forma automática.

- **Intervención.** El subsistema de intervención da soporte a la especificación de todos los aspectos dinámicos que aparecen vinculados a los escenarios de aprendizaje colaborativo. En relación a este subsistema pueden destacarse las siguientes mejoras:

  – *Especificación de los aspectos dinámicos.* Para especificar en Pelican un modelo de intervención relativamente complejo, como el de los ejemplos que fueron descritos en la sección 4.5, es necesario exigir a los administradores y diseñadores instruccionales nociones sobre programación orientada a objetos en P# y un conocimiento profundo de los artefactos del lenguaje tales como la colección de variables implícitas de dominio, tareas o eventos manejados por la plataforma y las herramientas externas. Este es un requerimiento poco realista en la práctica ya que los diseñadores suelen tener un perfil no técnico con lo que resulta complicado sacarle todo el partido posible a este subsistema. Precisamente para dar solución a este problema se propone, como se discutirá más ampliamente en las líneas de trabajo futuro (sección 5.3), un mecanismo para facilitar la tarea de diseño instruccional a partir de herramientas de diseño visual.

  – *Sincronización de contextos de usuario.* Como se discutió en el ejemplo de la figura 3.20, los contextos de usuario pueden ser conceptualizados como memorias de trabajo que almacenan variables definidas por el usuario para poder poner en contexto la ejecución de una familia de scripts que se ejecutan de forma secuencial. En la aplicación manual de los mismos, el monitor tiene que asegurarse de que los scripts siempre se apliquen sobre el mismo contexto de usuario para garantizar que cada uno encuentre las variables depositadas por los anteriores. En la aplicación automatizada, por el contrario, el contexto sobre el que se evalúa cada script de una regla corresponde al usuario que provocó el lanzamiento del evento asociado a la misma debido a algún tipo de acción o interacción que éste realizó sobre la plataforma o alguna herramienta externa. En la práctica, los causantes de la ejecución de los scripts que gobiernan un escenario desplegado dentro de un grupo son, potencialmente, todos los miembros del mismo, lo cual implica que cada script se evaluará arbitrariamente sobre alguno de los contextos de estos usuarios. Este hecho pone en peligro la correcta contextualización de los scripts. Para soslayar este problema, el diseñador instruccional, al programar cada script, debe seleccionar siempre un mismo contexto de usuario sobre el que almacenar y recuperar la información de contextualización (contadores, valores lógicos etc). Este es un defecto de diseño que podría solucionarse mediante la provisión de mecanismos de sincronización de contextos o la definición de *contextos de grupo.*

- *Efectos colaterales en la aplicación de reglas.* Como se explicó en la sección 3.5, la construcción de un plan de ejecución para determinar la secuencia ordenada de script que debe lanzarse a ejecución en respuesta a determinado evento es gestionada a través de la información de prioridad que, a tal efecto, se incluye en cada regla. Ante modelos de intervención bien diseñados, no deberían encontrarse situaciones conflictivas donde las reglas de un escenario, colisionasen de alguna forma con las de otro desarrollándose en paralelo. Por ejemplo porque la última regla anulase, total o parcialmente, los efectos de otras anteriores. Como reconocen algunos autores [Westerinen, 2001] [Moore, 2003] [Moore et al., 2001], la detección de colisiones y efectos colaterales resulta complicada de detectar de forma anticipada en los sistemas basados en reglas y además es un problema irresoluble sino es a través de una redefinición de las reglas.

- *Rendimiento en la aplicación concurrente de múltiples reglas.* El carácter interpretado del lenguaje de scripting P# hace que la evaluación de reglas y la subsiguiente ejecución potencial de los scripts asociados sea un mecanismo lento y poco escalable a medida que crece la cantidad de escenarios que están siendo manejados de forma automática a través de reglas (a medida que sube el número de reglas y scripts a ejecutar por unidad de tiempo). La solución de este problema no funcional pasa por articular un mecanismo mediante el cual la especificación de los aspectos dinámicos de un escenario, una vez establecida, pueda ser compilada a código objeto para obtener una ejecución más rápida.

- *Limitación en la reutilización del modelo de intervención.* El único mecanismo de reutilización del modelo de intervención consiste en la parametrización de las políticas definidas en Pelican. Esto permite adaptar un conjunto de reglas a un contexto particular de uso sustituyendo cada parámetro por un valor específico. Sin embargo, en Pelican no es posible aplicar un mismo modelo de intervención con dos parametrizaciones diferentes a contextos pedagógicos distintos de forma concurrente. Para ello seria necesario realizar una copia del modelo de intervención para cada escenario que, de forma concurrente, requiriese una parametrización diferente. Aunque esta necesidad no es muy común, entendemos que es una de las deficiencias más graves del subsistema de intervención. La solución sin embargo no es compleja: paralelamente a como ocurre con otros artefactos de la plataforma debería hablarse de plantillas de políticas, reglas y scripts que diesen lugar a políticas, reglas y scripts específicas cuando se desplegasen sobre un contexto social concreto.

Hasta ahora se han discutido las principales ventajas e inconvenientes de la plataforma Pelican. Antes de terminar esta sección conviene, sin embargo, señalar hasta qué punto las prestaciones proporcionadas por este sistema dan completa cobertura a los requerimientos de diseño y desarrollo relacionados con la puesta en práctica de experiencias de aprendizaje colaborativo. Para articular esta discusión se hará uso de los elementos conceptuales introducidos en la sección 2.2 (procesos, actores y fases):

- **Pelican y los actores del aprendizaje colaborativo.** En la descripción que presentamos en el capítulo 2, se definieron cinco actores principales, con distintas responsabilidades sociales que intervienen en el desarrollo de las experiencias de aprendizaje colaborativo. El administrador es el encargado de llevar a cabo las tareas de gestión del entorno virtual tales como registrar a los usuarios en la plataforma, crear los grupos de trabajo y enrolar a los primeros en los segundos de acuerdo a las necesidades. El diseñador instruccional se encarga de realizar las labores de diseño de los escenarios especificando cada uno de los cuatro modelos que conforman su definición en Pelican. El profesor, sobre todo en escenarios cooperativos, se encarga de gestionar el desarrollo de la experiencia, presentando las actividades, proporcionando discrecionalmente los recursos necesarios para su realización, arbitrando turnos de intervención si procede y administrando los tiempos de las sesiones de trabajo. El monitor realiza el seguimiento del desarrollo de las actividades en uno o varios grupos y aplica puntualmente estrategias de control para orientar a los estudiantes. Y finalmente, los estudiantes son los encargados de llevar a cabo las actividades que conforman la experiencia de aprendizaje. En este sentido, hablamos de actores y no de usuarios ya que comúnmente varias de estas responsabilidades son asignadas a una misma persona física. Por ejemplo, el profesor suele ser típicamente a la vez profesor, monitor e incluso diseñador instruccional. Ahora bien, ¿se permite en Pelican distinguir entre estos cinco estereotipos de usuarios? El subsistema social de la plataforma no solamente permite realizar tal distinción sino que además hace posible definir tantos actores como se crea conveniente para desarrollar el escenario, dentro del modelo de sociedad. La caracterización de las responsabilidades sociales de cada actor se hace a través de la asignación de una serie de permisos que incluso pueden ser dinámicamente modificados y/o extendidos, con respecto al vocabulario inicial, haciendo uso de las capacidades del subsistema de intervención.

- **Pelican y los procesos del aprendizaje colaborativo.** Tal y como se discutió en el capítulo 2, existen cuatro tipos de procesos complementarios relacionados con el diseño y desarrollo de experiencias instruccionales colaborativas. Los procesos de estratificación social abordan todos los aspectos que es necesario tener en cuenta para definir formalmente la estructuración de grupos y actores de un escenario. Los procesos de cooperación y colaboración toman en consideración todos aquellos elementos constituyentes de la descripción de actividad, sus tipos, niveles de estructuración, etc. Los procesos de monitorización y control, definen las tareas relacionadas con la definición de los indicadores de análisis que es posible utilizar para caracterizar distintos aspectos de una experiencia colaborativa y describen estrategias de intervención para controlar y reconducir el trabajo de los estudiantes. Finalmente, los procesos de evaluación abordan las cuestiones relacionadas con la evaluación formativa y sumativa, individual y grupal, del desarrollo de las experiencias. La pregunta en este punto acerca de cómo dan soporte a estos procesos las capacidades de la plataforma no resulta sencilla de responder ya que son muchos los artefactos tecnológicos que interactúan entre sí para dar cobertura a todos ellos. La tabla 5.1 resume las relaciones existentes en este sentido.

		Social	Colaboración	Integración	Intervención
Estructuración social	Diseño	Definición de los aspe tos estructurales de la comunidad virtual. Especificación de actores y plantillas de grupos	Definición de actividades orientadas a soportar estratificación social dirigida por los estudiantes	Definición de los servicios comunicando información del contexto social en el esquema de invocación	Definición de reglas y scripts de adaptación para crear, fusionar y absorber grupos, admitir, desadmitir, intercambiar usuarios etc.
	Desarrollo	Formación de dinámica de colectivos para dar soporte al desarrollo de las actividades	Uso de espacios de trabajo privados y compartidos para dar a los usuarios conciencia social en la comunidad	Adaptación de las herramientas en Pelican para transferir la consciencia social a las mismas	Aplicación de transformaciones para adaptar dinámicamente la estructura social atendiendo a las necesidades instuccionales
Cooperación y colaboración	Diseño	Definición de estructuras sociales para dar soporte a grupos formales e implicados en el desarrollo de una experiencia	Definición del trabajo de los estudiantes en términos de proyectos y actividades. Inclusión de referencias Web para completar la definición	Definición de servicios de interacción para contextualizar usos de herramientas. Especificación de roles para adaptar proporcionar información de contexto	Definición del flujo de trabajo instruccional para el secuenciamiento de las actividades. Orquestación de protocolos de integración
	Desarrollo	Formación y destrucción de grupos informales para realizar episodios colaborativos	Despliegue de proyectos y actividades en espacios de trabajo privados y compartidos	Niveles de interoperabilidad en las herramientas externas. Sensación de continuidad en su uso	Aplicación de las reglas para secuenciar actividades y para adaptar las herramientas al estado de desarrollo
Monitorización y control	Diseño	Definición de estructuras sociales para aplicar estrategias de monitorización recíproca	Definición de actividades orientadas a monitorización recíproca. Por ejemplo método de Pecera	Especificación de roles para definir las responsabilidades y permisos de acceso de estudiantes y monitores	Definición de reglas para el cálculo de indicadores de análisis y para la aplicación de estrategias de control.
	Desarrollo	Adaptaciones sociales para aplicar estrategias de control del desarrollo de las actividades	Activación y desactivación de proyectos, servicios y referencias	Bloqueo selectivo por roles de servicios y referencias. Control de la interacción basada en roles y en el uso de servicios Web externos	Aplicación de estrategias de control cuando los indicadores superan valores umbrales
Evaluación	Diseño	Definición de estructuras sociales para dar soporte a escenarios de enseñanza recíproca	Definición de actividades orientadas a realizar evaluación colaborativa como en el método Send a Problem.	Especificación de roles para definir las responsabilidades y permisos de acceso de estudiantes y profesores	Definición de reglas para la evaluación automática de ciertos productos. Asignación de calificaciones individuales y grupales
	Desarrollo	Adaptaciones sociales para permitir la promoción de los estudiantes de un curso al siguiente.	Control del estado de proyectos y actividades a través de su ciclo de vida	Apoyo a tareas de administración para la evaluación basada en productos	Realización de tareas de administración tales como la promoción de estudiantes o la recolección de productos del repositorio para su evaluación

Tabla 5.1. Relación entre subsistemas de Pelican y tipos de procesos

- **Pelican y las fases del aprendizaje colaborativo.** En cada uno de los cuatro tipos de procesos, las tareas y sucesos se organizan en dos fases complementarias que se desarrollan potencialmente en paralelo a lo largo del tiempo. En la fase de diseño tienen lugar las relacionadas con el diseño de los escenarios mientras que en la fase de desarrollo se inscriben aquéllas vinculadas a la realización de la actividad. Por ejemplo, en los procesos de cooperación y colaboración, la definición del prompt de una actividad es una tarea necesaria para definir la misma. Por su parte, cada episodio de negociación durante su realización, es un suceso circunscrito a la fase de desarrollo. Estas dos fases conceptuales de nuestro modelo teórico quedan perfectamente identificadas en Pelican a través de los actores que desarrollan cada tarea o que aparecen involucrados en cada suceso. Así las tareas y sucesos orientados a la especificación donde participan principalmente diseñadores instruccionales y administradores, y en menor medida, profesores y monitores, pertenecen a la fase de diseño mientras que aquéllas, relacionadas con la realización de las actividades, donde participan profesores y monitores y sobre todo estudiantes pertenecen a la fase de desarrollo. Las principales responsabilidades de cada subsistema de la plataforma con respecto a los cuatro tipos de procesos se han distribuido en la tabla 5.1, de acuerdo a estas dos fases complementarias.

5.3. Líneas futuras

El desarrollo de este trabajo de tesis nos ha permitido adquirir una visión de conjunto de la problemática que aparece asociada a la puesta en práctica de escenarios de aprendizaje colaborativo soportados computacionalmente. En este sentido, la revisión de las aportaciones de la comunidad científica que fue desarrollada en el capítulo 2 permite conocer el estado de madurez de las soluciones tecnológicas lo que nos conduce a establecer algunas líneas de investigación futuras de nuestro interés. En los siguientes apartados entraremos a discutir algunas de las más relevantes.

Definición de un modelo teórico de aprendizaje colaborativo y una metodología

Definir un modelo teórico acerca del aprendizaje colaborativo que resuma todas las tareas de diseño y desarrollo que es necesario realizar para poner en práctica experiencias colaborativas es un ejercicio de análisis de los aspectos teóricos en la dimensión pedagógica de este paradigma que merece la pena abordar en aras a obtener una metodología para la especificación formal de escenarios de aprendizaje colaborativo. En efecto, este modelo teórico no sólo puede considerarse como un modelo de referencia para orientar a diseñadores instruccionales y profesores en las tareas de especificación de experiencias formativas de esta naturaleza sino que, adicionalmente, constituye una lengua franca para facilitar la comunicación entre investigadores especializados en el terreno de la psicología, la pedagogía y la informática. En este sentido, seria conveniente formular el mismo en términos procedimientales, describiendo la secuencia de tareas de diseño que es preciso llevar a cabo para obtener la especificación completa de un escenario, e indicar, para cada una de ellas, los productos entrantes y salientes y los actores que aparecen vinculados en el desarrollo de las mismas.

Además podría definirse un perfil de UML como lenguaje visual para dar soporte a tal especificación formal de los escenarios. Esto redunda en ventajas ya que los actores involucrados podrían discutir sus diseños utilizando una base conceptual común que evite ambigüedades y malentendidos muy frecuentes en las especificaciones en lenguaje natural.

Simplificación de la especificación del modelo de intervención

De todos los lenguajes de modelado proporcionados por la plataforma Pelican, el más complicado es el que se utiliza para definir el modelo de intervención. Esto se debe a dos motivos fundamentales. En primer lugar, que la especificación de comportamientos reactivos en la plataforma para capturar los aspectos dinámicos de un escenarios resulta inherentemente compleja ya que requiere de la especificación de algoritmos de indiquen a Pelican cómo debe adaptarse para tratar cada evento ocurrido durante el desarrollo del flujo de trabajo instruccional y esto requiere cierto perfil técnico del que muchos diseñadores instruccionales no disponen. En segundo lugar, que tal especificación no es visual, como en los otros tres lenguajes, sino dirigida por la sintaxis de un lenguaje orientado a objetos, P#, del que es preciso conocer su normas de escritura, el conjunto de variables implícitas de dominio que se utiliza para interpretar los scripts y la colección de tipos de eventos, internos y externos, junto con sus atributos característicos, para la definición de reglas.

La simplificación de estas labores de diseño no resulta trivial aunque si necesaria si queremos hacer viable el uso de la plataforma en condiciones realistas; esto es, con ausencia de técnicos de soporte. En este sentido, las arquitecturas dirigidas por modelos ofrecen una solución para definir herramientas de diseño visuales que permitan a los diseñadores instruccionales especificar fácilmente escenarios de aprendizaje colaborativo. La responsabilidad de tales herramientas sería la de traducir el modelado visual del diseñador en un conjunto de reglas y scripts P# que se instalen automáticamente en la plataforma a través de los servicios Web de la misma.

En este sentido es necesario advertir no obstante, que en computación cualquier proceso de abstracción dirigido a acercar una especificación desde un lenguaje de bajo nivel – en nuestro caso P# – a un lenguaje próximo al humano – el lenguaje de modelado visual – tiene una penalización con respecto a tres factores:

- **Eficiencia en tiempo de ejecución.** Las reglas y scripts de adaptación generados de forma automática por esta nueva herramienta de modelado instruccional serían correctas en tanto que alcanzarían los objetivos perseguidos en el diseño visual. Sin embargo, el código de las mismas sería probablemente menos eficiente que el obtenido mediante la especificación directa de un programador.
- **Eficiencia en tamaño de código.** De forma similar, el código producido por la herramienta de diseño genera, como ocurre en cualquier proceso de compilación, código redundante con fragmentos frecuentemente innecesarios. Esto supone un mayor tamaño de la especificación expresado en términos de reglas y scripts y también una mayor consumición en memoria para llevar a cabo su ejecución.

- **Expresividad del lenguaje.** El lenguaje visual de la herramienta de diseño resulta más sencillo porque ofrece constructores de mayor nivel de abstracción para describir los escenarios. No obstante, como contrapartida, cada abstracción implica que el nivel de detalle al que pueden realizarse las especificaciones es menor reduciéndose así la flexibilidad y potencia expresiva del lenguaje.

Construcción automática asistida de herramientas de soporte a la interacción

De nuestras experiencias durante el desarrollo de actividades colaborativas en escenarios reales dentro del proyecto ENLACE pueden extraerse dos importante conclusiones. En primer lugar, que idealmente la especificación de cada escenario colaborativo requiere de una familia de servicios de interacción específicos que comúnmente son dependientes de dominio y siempre están fuertemente vinculados con el tipo de interacción colaborativa particular a la que se pretende dar soporte. En segundo lugar que los costes de desarrollo de estas herramientas son el cuello de botella para la especificación de escenarios.

En este sentido, cabe preguntarse si sería posible definir una solución tecnológica para permitir a los diseñadores instruccionales, de forma visual y sencilla, definir protocolos de interacción y generar automáticamente, a partir de tal especificación, herramientas de soporte a la colaboración. Para ello habría que determinar cómo debería ser el lenguaje para la especificación de los protocolos, cómo debería ser la herramienta que los interpretase para convertirlos en un servicio de interacción y cómo debería modificarse la plataforma para integrar los mismos dentro de su entorno.

Mejora del soporte al análisis de la interacción colaborativa

Pelican constituye una plataforma de integración, orientada a la realización de experiencias de aprendizaje colaborativo, que permite a los desarrolladores de herramientas externas implementar servicios de soporte a la interacción aprovechando las facilidades, mecanismos e información de contexto proporcionados por ésta. De esta manera, tanto las herramientas externas como la propia plataforma establecen entre sí una relación simbiótica de alta cohesión e interoperabilidad que ofrece a los estudiantes un entorno integrado para la realización de sus actividades colaborativas. En efecto, mientras que la plataforma se encarga de controlar los aspectos administrativos de la comunidad virtual de aprendizaje y de la organización del trabajo de los estudiantes, las herramientas proporcionan el medio tecnológico donde tiene lugar la interacción entre los mismos.

Los resultados del análisis de la interacción colaborativa de los estudiantes son un producto que interesa a ambas partes, tanto a la plataforma como a las propias herramientas. A la plataforma porque el estado de la colaboración condiciona potencialmente las prestaciones que ésta debe proporcionar a los usuarios. A las herramientas porque éstas pueden alterar su comportamiento para controlar y reconducir la interacción colaborativa. Los subsistemas de integración y de intervención proporcionan las capacidades pertinentes para dar soporte a esquemas de monitorización y control tal y como se demostró en el escenario de la subsección 4.5.4.

La línea de trabajo futuro que se abre en este sentido es la de investigar la viabilidad de definir modelos generales de análisis [Dimitrakopoulou, 2004] [Dimitrakopoulou, 2005] para la definición de indicadores y el cálculo de forma dinámica y automática según transcurre el desarrollo de la experiencia asi como el uso de procedimiento de control estándar que puedan aplicarse cuando se alcancen determinadas condiciones ambientales.

Bibliografía

[Alf]. http://innova.uned.es

[Ariadne]. Ariadne Learning Object Repository. http://www.ariadne-eu.org/index.php?option=com_content&task=view&id=27&Itemid=38

[Aronson et al., 1978]. Aronson E. Blaney N. Stephan C. Snapp M. (1978) "The jigsaw classroom". Beberly Hills. CA: Sage.

[Arrow, 1951] Arrow K.J. (1951). "Social Choice and Individual Values". Wiley, New York.

[Austin, 1962]. Austin, J. L. (1962). "How to Do Things With Words". Oxford University Press: Oxford, England.

[Ausubel et al., 1983]. Ausubel D.P. Novak J.D. Hanesian H. (1983). "Psicología educativa: Un punto de vista cognitivo." México: Trillas.

[Baker, 1991]. Baker M.J. (1991) "The influence of the generation processes on the generation of the student's collaborative explanations for simple physical phenomena". Proceedings of the international conference on the learning sciences. Evanston, Illinois, USA. pags. 9-19

[Barkley et al., 2005] Barkley E. F. Cross K. P. Major C. H. "Collavorative Learning Techniques. A handbook for college faculty". Willey & Son Eds.

[Barros & Verdejo, 2000] Barros B. & Verdejo F. (2000) "Analysing student interaction processes in order to improve collaboration: The DEGREE approach". Journal of Artificial Intelligence in Education, 11, 211-241

[Barros et al., 2003] Barros B. Vélez J. Verdejo M.F. (2003) "Experiencias de aplicación de la Teoría de la Actividad en el desarrollo de Sistemas Colaborativos de Enseñanza y Aprendizaje". Actas del congreso AEPIA 2003.

[Berkowitz & Gibbs, 1983]. Berkowitz M.W. & Gibbs J.C. (1983) "Measuring the development of features of moral discussion". Merril–Palmer Quarterly. 29. pags. 399-410.

[Bichler, 2000]. Bichler M. (2000) "A roadmap to auction based negotiation protocols for electronic commerce". Proceedings of the 33rd Hawaii International Conference on System Science. 2000

[Blackboard]. http://www.blackboard.com

[Blackburn et al., 2001] Blackburn P. Rijke M. Venema Y. (2001). "Modal Logic." Cambridge University Press.

[Bote – Lorenzo et al., 2004]. Bote-Lorenzo M. L. Vaquero-González L. M. Vega-Gorgojo G. Dimitriadis Y. Asensio-Pérez J. I.Gómez-Sánchez E. Hernández-Leo D. (2004) "A Tailorable Collaborative Learning System that Combines OGSA Grid Services and IMS-LD Scripting" Proceedings of the X International Workshop on Groupware, CRIWG 2004.Springer-Verlag, LNCS 3198, 305-321, San Carlos, Costa Rica, September 2004.

[Bote–Lorenzo et al., 2008]. Bote-Lorenzo M.L. Gómez-Sánchez E. Vega-Gorgojo, G. Dimitriadis Y. Asensio-Pérez J.I. Jorrín-Abellán I.M. Gridcole: a tailorable grid service based system that supports scripted collaborative learning Computers & Education. 51(1):155-172, August 2008.

[Brookfield & Presskill, 1999]. Brookfield S.D. Presskill S. (1999) "Discussion as a way of teaching: Tools and techniques for democratic collaborative learning classrooms" San Francisco: Jossey–Bass.

[Bruner, 1997]. Bruner J. (1997) "La educación, puerta de la cultura". Visor (Ed.) España: Madrid.

[Brusilovsky, 1999]. Brusilovsky P. (1999) “Adaptive and Intelligent Technologies for Web-based Education”, Special Issue on Intelligent Systems and Teleteaching, Künstliche Intelligenz, 4, pp19-25.

[Blaye, 1988]. Blaye A. (1988) "Confrontation socio–cognitive et résolution de problèms". Doctoral dissertation. Centre de Recherche en Psychologie. Universite de Provence, 13261 Aix en Provence. France.

[Boom, 1956]. Bloom B.S. (1956) "Taxonomy of educational objetives: Book 1, cognitive domain". New York: Longman Bloom B.S. (Eds.)

[Bote-Lorenzo et al., 2004]. Bote – Lorenzo, M. L. Hernández-Leo, D. Dimitriadis, Y. Asensio – Pérez, J. I. Gómez – Sánchez, E. Vega – Gorgojo, G. Vaquero González, L. M. 2004. “Towards reusability and tailorability in collaborative Learning systems using IMS – LD and grid services advanced technology for learning.” 1 (3). pp. 129-138. September, 2004.

[BPEL4WS]. http://www.ibm.com/developerworks/library/specification/ws-bpel

[Butterworth, 1982]. Butterworth G. (1982) "A brief account of the conflict between the individual and the social in models of cognitive growth". In G. Butterworth & P. Light (Eds.) Social Cognition. pags. 3-16. Brighton, Sussex: Harvester Press.

[Caeiro et al.,2008]. Caeiro M. "PoEML: a separation-of-concerns proposal to instructional design". In Botturi, L., & Stubbs, T. (Eds.), Handbook of Visual Languages for Instructional Design: Theories and Practices. IDEA Group Inc., 2008.

[Caeiro et al. 2006]. Caeiro, M. Llamas, M. Anido-Rifón, L.E. "The PoEML Proposal to Model Services in Educational Modelling Languages". CRIWG 2006, 187-202.

[Caeiro et al., 2003]. Caeiro, M. Anido, L. Llamas, M. 2003. "A critical analysis of IMS Learning Design". In Wasson, B. Anido L. & Hoppe, U. (Eds.), Proceedings of the International Conference on Computer Support for Collaborative Learning, Bergen: Kluwer Academic Publishers, 363-367

[Clancey, 1997]. Clancey W.J. (1997) "Situated cognition: On human knowledge and computer representations". Cambridge University Press. New York. USA.

[Clark & Brennan, 1991]. Clark H.H. Brennan S.E. (1991) "Grounding in communications". In L. Resnik J. Levine & S. Tasley (Eds.) Perspectives on Socially Shared Cognition. pags. 127-149. Hyattsville MD: Americal Psychological Association.

[Cohen, 1994]. Cohen E.G. (1994) "Discourse about ideas: Monitoring and regulation in face to face and computer–mediated environments". Interactive Learning Environments. 6 (1-2). pags. 96-113.

[Coi et al., 2008]. Coi J.L. Kärger P. Koesling A.W. Olmedilla D. (2008) "Control youy eLearning Environment: Exploiting Policies in an Open Infraestructure for Lifelong Learning". IEEE Transactions on Learning Technologues. Vol 1. No 1. January-March 2008.

[COLDEX]. http://www.coldex.info

[Constantino-González & Suthers, 2002]. Constantino-González M. Suthers D. (2002) "Coaching collaboration incomputer-mediated learning." In G. Stahl (Ed.), Computer support for collaborative learning: foundations for a CSCL community. Pags. 583-584. Mahwah, NJ: Lawrence Erlbaum Associates.

[Davis, 1993] Davis B.G. (1993) "Tools for teaching" San Francisco: Jossey-Bass.

[Dillenbourg, 2002]. Dillenbourg P. (2002)."Over-scripting CSCL: The risks of blending collaborative learning with instructional design". In P. A. Kirschner (Ed). Three worlds of CSCL. Can we support CSCL (pp. 61-91). Heerlen, Open Universiteit Nederland.

[Dillenbourg, 1999]. Dillenbourg P. (1999) "What do you mean by 'colaborative learning'?". In P. Dillembourg (Ed.) Collaborative–learning: Cognitive and Computational Approaches. pags. 1-19. Oxford: Elsevier.

[Dillenbourg et al., 1996]. Dillembourg P. Baker M. Blaye A. O'Malley C. (1995) "The evolution on research on collaborative learning". In E. Spada and P. Reiman (Eds.) Learning in humans ans machine: toward an interdisciplinary learning science. pags. 189-211. Oxford: Elsevier.

[Dillenbourg & Baker, 1996]. Dillembourg P. Baker M. (1996) "Negotiation Spaces in Human–Computer Collaborative Learning". Proceedings of the International Conferene on Cooperative Systems. COOPS'96. Juan Les Pins (Fance).

[Dillenbourg & Jermann, 2004]. Dillenbourg P. Jermann P. (2004) "A model for designing CSCL scripts" In Fisher, Hund, Haake & Koller (Eds.) Scripting Computter Supported Communication of Knowledge Cognitive, Computational and Educational Perspectives..

[Dillenbourg & Schneider, 1995]. Dillenbourg P. Schneider D (1995) "Mediating the mechanisms which makes collaborative learning sometimes effective". International Journal of Educational Telecomunication. 1 (2/3) pags. 131-146.

[Dimitrakopoulou, 2004]. Dimitrakopoulou, A. 2004. "State of the art on interaction and collaboration analysis." Deliverable D.26.1.1 ICALTS Project. Prepared for the European Commission, DG INFSO, under contract ITS 507838 as a deliverable from WP 26.

[Dimitrakopoulou, 2005] Dimitrakopoulou, A. 2005. "State of the art of interaction analysis for metacognitive support and diagnosis." Deliverable D.31.1.1 IA Project. Prepared for the European Commission, DG INFSO, under contract ITS 507838 as a deliverable from WP 26.

[dotLRN]. http://dotlrn.org

[Doyse & Mugny, 1984]. Doyse W. Mugny G. (1984) "The social development of the intellect". Oxford: Pergamon.

[ENLACE]. http://enlace.uned.es

[EsDeLibro] http://esdelibro.es

[Fischer, 2001]. Fischer F. (2001) "Collaborative knowledge construction. Analysis and facilitation in computer–superted collaborative scenarios". Professorial disertation. Ludwig Maximilian University of Munich.

[Fischer et al., 2002] Fischer F. Bruhn J. Gräsel C. Mandl H. (2002) "Fostering collaborative knowledge construction with visualization tools". Learning and Instruction. 12. pags. 213-232.

[Fowler, 2005] Fowler M "Generating Code for DSLs". (2005). On the Web at http://martinfowler.com/articles/codeGenDsl.html

[Fox, 1987]. Fox B.(1987) "Interactional reconstruction in real–time language processing". Cognitive Science. 11 (3). Pags. 365-387

[Gaudioso, 2002]. Gaudioso E. (2002) "Contribuciones al modelado de usuario en entornos adaptativos de aprendizaje y colaboración a través de Internet mediante técnicas de aprendizaje automático". PhD. Thesis UNED. Facultad de Ciencias (España)

[Gómez–Sánchez et al., 2009]. Gómez-Sánchez E. Bote-Lorenzo M.L. Jorrín-Abellán I.M. Vega-Gorgojo G. Asensio-Pérez J.I. Dimitriadis Y. "Conceptual framework for

design, technological support and evaluation of collaborative learning". International Journal of Engineering Education. 25(3):557-568, May 2009.

[GoogleMaps]. http://maps.google.es

[Governatori et al., 2001]. Governatori G. Dumas M. ter Hofstede A.H.M. Oaks P. (2001) "A formal approach to protocols and strategies for (legal) negotiation". In Henry Prakken, editor, Procedings of the 8th International Conference on Artificial Intelligence and Law, pags. 168-177. IAAIL, ACM Press, 2001

[Groovy]. http://groovy.codehaus.org

[Hatano & Inagaki, 1991]. Hatano G. Inagaki K. (1991) "Sharing cognition through collective comprehension activity". In L.B. Resnick J.M Levine & S.D. (Eds.) Teasley. Perspectives on socially shared cognition. pags. 331-348. Washington D.C. Americal Psychological Association.

[Henri & Pudelko, 2003]. Henri F. Pudelko B. (2003) "Understanding and analysing activity and learning in virtual communities" Journal of Computer Assisted Learning, 19. Pags. 474-487

[Hernandez–Leo et al., 2008] Hernández-Leo, D., Santos, P., Villasclaras-Fernández, E.D., Navarrete, T., Asensio-Pérez, J.I., Blat, J., Dimitriadis, Y. Educational patterns as a guide to create units of learning and assessment Proceedings of the 8th IEEE International Conference on Advanced Learning Technologies, ICALT 08, 1055-1056, Santander, Spain, July 2008.

[Hernandez–Leo et al., 2006]. Hernández-Leo D. Villasclaras-Fernández E. D. Asensio-Pérez J. I. Dimitriadis Y. & Retalis, S. (2006). "CSCL scripting patterns: Hierarchical relationships and applicability. In Proceedings of the sixth IEEE international conference on advanced learning technologies, ICALT 2006 (pp. 388–392). Kerkrade, The Netherlands.

[Hernandez–Leo et al., 2005]. Hernández – Leo, D. Asensio – Pérez, J. I. Dimitriadis, Y. 2005. “Computational representation of collaborative Learning for patterns using IMS Learning Design educational technology & society. 8(4) pp.75-89, October, 2005.

[Hogan et al., 2000]. Hogan K. Nastasi B.K. Pressley M. (2000) "Discurse patters and collaborative scientific reasoning in peer and teacher–guided discusions". Cognition and Instruction. 17 (4). pags. 379-432.

[Hoppe et al., 2005]. Hoppe U. Pinkwart N. Oelinger M. Zeini S. Verdejo M. F. Barros B. Mayorga J.I. (2005) "Building bridges within learning communities through ontologies and "thematic objects" In Proceedings of th 2005 conference on Computer support for collaborative learning

[Hoppe & Ploetzner, 1999]. Hoppe H.U. Ploetzner R. (1999) "Can analytic models support learning in groups?". In P Dillenbourg (Ed.) Collaborative Learning: cognitive and computational approach. Pags: 147-168. Oxford Elsevier Science Publisher.

[IMS–CP]. IMS Global Learning Consortium (2004) "IMS Content Packaging Information Model". At http://www.imsglobal.org/content/packaging

[IMS–EM]. IMS Global Learning Consortium (2002) "IMS Enterprise Information Model". At http://www.imsglobal.org/enterprise

[IMS–LD]. IMS Global Learning Consortium (2003) "IMS learning design information model". At http://www.imsglobal.org/learningdesigns

[IMS–LIP]. IMS Global Learning Consortium (2006) "IMS learner information Package". At http://www.imsglobal.org/profiles/index.html

[IMS–LOM] IMS Global Learning Consortium (2002) "IMS learning resource metadata specification". At http://www.imsproject.org/metadata

[IMS–SS]. IMS Global Learning Consortium (2003) "IMS Simple Sequencing Information and Behavior Model". At http://www.imsglobal.org/simplesequencing

[IMS–TI]. IMS Global Learning Consortium (2003) "IMS Tools Interoperability Guidelines". At http://www.imsglobal.org/ti

[Java]. http://www.java.com

[Jermann & Dillenbourg, 2002]. Jermann P Dillenbourg P (2002) "Elaborating new arguments through a CSCL script". In Andriessen J. Baker M. Suthers D. (Eds.) Arguing to learn: Confronting cognitions in Computer–Supported Collaborative Learning Environments. Pags. 1-6 2002 Kluwer Academic Publisher.

[Johnson & Johnson, 1994]. Johnson D.W. Johnson R.T. (1994) "Learning together". In Handbook of Cooperative Learning Methods. Sharan S. (Ed.) Greenbook Press.

[Kagan & Kagan, 1994]. Kagan S. Kagan M. (1994) "The structural approach: Six keys to cooperative learning". In S. Sharan (Ed.) Handbook on Cooperative Learning Methods. Greenwood Press.

[Keefer et al., 2000] Keefer M.W. Zeitz C.M. Resnick L.B. (2000) "Judgin the quality of peer–led student dialogues". Cognition and Instruction 18 (1). pags. 53-81.

[King, 1999]. King A. (1999) "Discourse patterns for madiating peer learning". In A.M. O'Donnell & A. King (Eds.) Cognitive Perspectives on Peer Learning.Pags. 87-115. Mahwah NJ: Erlbaum

[Kollar et al., 2005] Kollar I. Fischer F. and Slotta J.D. (2005) "Internal and external collaboration scripts in web-based science learning at schools". In Proceedings of Th 2005 Conference on Computer Support For Collaborative Learning: Learning 2005: the Next 10 Years! (Taipei, Taiwan, May 30 - June 04, 2005). Computer Support for Collaborative Learning. International Society of the Learning Sciences, 331-340.

[Kollar et al., 2003]. Kollar I. Fischer F & Hesse F.W. (2003). "Cooperation scripts for computersupported collaborative learning". In B. Wasson, R. Baggetun, & U. Hoppe, (Eds.), Proceedings of the computer supported collaborative learning 2003, Community events – communication and interaction (pp. 59–61). Bergen, Norway.

[König et al., 2007]. König D. Glover A. King P. Laforge G. Skeet J. 2007) "Groovy in action". Manning (Ed.)

[Kreijns et al., 2002]. Kreijns K. Kirschner P.A. Jochems W. (2002) "The sociability of computer supported collaborative learning environments". In Educational Technology & Society. 5 (1). 2002

[Kreijns et al., 2003]. Kreijns K. Kirschner P.A. Jochems W. (2003) "Identifying the pitfalls for social interaction in computer supported collaborative learning environments: a review of the research". In Computers in Human Behabiour. 19. 2003. pags. 335-353

[Krochmann, 1996] Krochmann T. (1996) "CSCL: Theory and practice of an emergent paradigm". Lawrence Erlbaum Associates.

[Kruger, 1992]. Kruger A. (1992) "The effect of peer and adult–child transactive discussion on moral reasoning". Merril–Palmer Quarterly. 38. pags 191-211.

[Kruger & Tomasello, 1986]. Kruger A. Tomasello M. (1986) "Transactive discussion with peer and adults". Developmental Psychology. 22. pags. 681-685

[Larrañaga et al., 2002]. Larrañaga M. Rueda U. Elorriaga J.A. & Arruarte A. (2002). "Using CM-ED for the Generation of Graphical Exercises Based on Concept Maps." In Kinshuk Lewis R. Akahori K. Kemp R. Okamoto T. Henderson L. & Lee C.H. (Eds.) Proceedings of ICCE'2002 pags. 173-177.

[Lave, 1988]. Lave J. (1988) "Cognition in practive". Cambridge: Cambbridge University Press.

[LotusNotes]. http://www-01.ibm.com/software/es/lotus/

[McKeachie et al., 1986]. McKeachie W. J. Pintrich P. R. Lin Y. Smith D. A. (1986) "Teaching and learning in the collage classroom: A review of the research literature" Ann Arbor: University of Michigan, National center of research to improve postsecondary teaching and learning

[Meyer, 1999] Meyer B. (1999) "Construcción de software orientada a objetos" Segunda edición. Prentice Hall, Madrid.

[Merlot] http://www.merlot.org/merlot

[Miao et al., 2005]. Miao, Y., Hoeksema, K., Hoppe, H. U., & Harrer, A. (2005). CSCL scripts: Modelling features and potential use. In Koschmann, T., Suthers, D., & Chan, T. W. (Eds.), Proceedings of the Computer Supported Collaborative Learning 2005: The Next 10 Years! Mahwah, NJ, USA: Lawrence Erlbaum, 423-432.

[Millis & Cottel, 1998] Millis B.J. Cottel P.G. "Cooperative Learning for Higher Education Faculty" American Council of Education. Phoenix. AZ, Oryx Press. (1998)

[Moodle]. http://moodle.org

[Moore, 2003]. Moore B. (2003) "Policy Core Information Model Extensions". RFC 3460.

[Moore et al., 2001] Moore B. Elleson E. Strassner J. Westerinen A. (2001) "Policy Core Information Model Specification". RFC 3060

[MySQL]. http://mysql.com

[Northrup, 2001]. Northrup P (2001) "A framework for designing interactivity in Web based instruction". In Educational Technology. 41 (2). pags. 31-39.

[O'Donnell & Dansereau, 1992]. O'Donnell A. M. Dansereau, D. F. (1992) "Scripted cooperation in student dyads: A method for analyzing and enhancing academic learning and performance". In R. Hertz-Lazarowitz and N. Miller (Eds.), Interaction in cooperative groups: The theoretical anatomy of group learning pp. 120-141. London: Cambridge University Press.

[OGSA]. The Globus Alliance (2008) "Open Grid Services Architecture". At http://www.globus.org/ogsa/

[OSGi]. OSGi Alliance. (1999) "Open Services Gateway Initiative". At http://www.osgi.org

[Palomino–Ramirez et al., 2008]. Palomino-Ramírez L. Bote-Lorenzo M.L. Asensio-Pérez J.I. Dimitriadis Y. "LeadFlow4LD: Learning and Data Flow Composition-based Solution for Learning Design in CSCL". Proceedings of the Proceedings of the 14th International Workshop on Groupware, CRIWG 2008, 266-280, Omaha, NE, USA, September 2008.

[Panitz, 1997]. Panitz, T. (1997) "Collaborative versus cooperative learning. A comparision of the two concepts with will help us understand the underlying nature od interactive learning". On the Web at http://www.lgu.ac.uk/deliberations/collab.learning/panitz2.html

[Pea, 1993]. Pea R.D. (1993) "Learning scientific concepts through material and social activities: Conversational analysis meets conceptual change". Educational Sychologyst 28 (3). Pags. 265-277.

[Pearce & Cronen, 1980]. Pearce, W. B., & Cronen, V. (1980). "Communication, action, and meaning: The creation of social realities". New York: Praeger.

[Perret–Clermont et al., 1991]. Perret–Clermont A.N. Perret F.F. Bell N. (1991) "The social construction of meaning and cognitive activity in elementary school children". In L. Resnick J. Levine & S. Teasley (Eds.) Perspectives on Socially Shared Cognition. Pags. 41-62. Hyattsville, MD: Americal Psycological Association.

[Peter & Vantroys, 2005]. Peter Y. & Vantroys T. (2005). Platform Support for Pedagogical Scenarios. Educational Technology & Society, 8 (3), 122-137.

[Piaget, 1928]. Piaget P. (1928) "Judgment and reasoning in the child". London: Routledge.

[Piaget, 1985]. Piaget J. (1985) "The equilibrium of cognitive structures: The central problem of intellectua development". Chicago. University of Chicago Press.

[Renkl, 1997]. Renkl A. (1997) "Learning through teaching central mechanisms in cooperative learning". Wiesbaden: Deutscher Universitäts–Verlag.

[Renkl & Mandl, 1995]. Renkl A. Mandl H. (1995) "Cooperative Learning. A question of what is necessary and replaceable". Unterrichtswissenschaft. 23. pags. 292-300

[Resnik, 1991]. Resnik L. (1991) "Shared cognition: Thinking as social practice". In L. Resnick J. Levine & S. Teasley (Eds.) Perspectives on Socially Shared Cognition. Pags. 1-22. Hyattsville, MD: Americal Psycological Association.

[Rodriguez & Verdejo, 2004] Rodriguez M. Verdejo M.F. (2004) "Modelling educational content: The cognitive approach of the palo language". In Journal of Educational Technology & Society. 7 (3)

[Rogoff, 1991]. Rogoff B. (1991) "Social interactions as apprenticeship in thinking: Guided participation in spatial planning". In L. Resnick J. Levine & S. Teasley (Eds.) Perspectives on Socially Shared Cognition. Pags. 349-364. Hyattsville, MD: Americal Psycological Association.

[Roschelle, 1992]. Roschell J. (1992) "Learning by collaborating: Convergent conceptual change". Journal of the learning science. 2. Pags. 235-276.

[Roschelle & Teasley, 1995]. Roschell J. Teasley S. (1995) "The construction of shared knowledge in collaborative problem solving". In C.O'Malley (Ed.) Computer Supported Collaborative Learning. (128) pags. 67-97. Berlin: Springer.

[Seo–BirdLife]. http://www.seo.org

[Sharan & Sharan, 1994]. Sharan Y. Sharan S. (1994). "Groop investigation in the cooperative classroom". In S. Sharan (Ed.) Handbook of Cooperative Learning Methods. Greenwook Press.

[Skinner, 1982] B.F Skinner (1982) "Skinner for the Classroom" R. Epstein (Ed.)

[Slavin, 1983]. Slavin R.E. (1983) "Team-Assisted Individualization: A Cooperative Learning Solution for Adaptive Instruction in Mathematics". Johns Hopkins Univ., Baltimore, MD. Center for Social Organization of Schools.

[Slavin, 1991]. Slavin R.E. (1991) "Student team learning. A practical guide to cooperative learning". National Education Association Professional Library West Haven.

[Slavin, 1993]. Slavin R.E. (1993) "Synthesis of research in cooperative learning". In A.E. Woolfolk (Ed.) Readings and Cases in Educational Psychology. pags. 170-178. Needham Heights: Allyn & Bacon.

[Smith, 1996]. Smith K.A. (1996) "Cooperative Learning: Making 'group work' work". In T.E. Sutherland & C.C. Bonwell (Eds.) Using active learning in college classes: A range of option for faculty. pags. 71-82. New directions for teaching and learning, 67. San Francisco Josey–Bass.

[Suchman, 1987]. Suchman L.A. (1987) "Plans and situated accions: The problem of human–machine communication". Cambridge: Cambridge University Press.

[Tan et al., 2006] Tan I. Sharan S. Lee C. (2006) "Group Investigation and Student Learning: An speriment in Singapoor schools". Marshall Caverdish Academics.

[Teasley, 1997]. Teasley S. (1997) "Talking about reasoning: How important is the peer in peer collaboration?". In L.B. Resnick R. Säljö C. Montecorvo & B. Burge (Eds.) Discurse, tools and reasoning: Essays on situated cognition. pags. 361-384. Berlin: Springer.

[Teasley & Roschelle, 1993]. Teasley S. Roschelle J. (1993) "Constructing a joint problem space: The computer as a tool for sharing knowledge". In S.P. Lajoie & S.J. Derry (Eds.) Discurse, tools, and reasoning: Essays on sitiated cognition. pags. 229-258. Berlin: Springer.

[Tomcat]. http://tomcat.apache.org

[Van de Velde et al., 2004]. Van de Velde W. Häkkinen P. Järvela S. Stegmann K. "Examples of CSCL Scripts within organisational framework". NoE. Kaleidodoscope JEIRP MOSIL (Mobile Support for Integrated Learning). Deliverable 23.3.1

[Vantroys & Peter, 2003]. Vantroys T. & Peter Y. (2003) “COW, A flexible platform for the enactment of Learning scenarios“. Lecture notes in Computer Science.Springer Berlin Volume 2806/2003.

[Vélez, 2005]. Vélez, J. (2005). "Diseño de fichas". Informe Técnico Proyecto ENLACE. Tarea 1.2 (TIN 2004 - 04232)

[Vélez et al., 2005]. Vélez J. Mayorga J. I. Verdejo M. F. (2005). "A Metamodel for Defining and Managing Web Based Communities." IADIS International Conference. Web Based Communities 2005. pp.183-190 ISBN: 972-99353-7-8

[Vélez et al., 2005b]. Vélez J. Barros B. Verdejo M. F. (2005). "Mechanism for auto-organization and regulation of virtual communities. Taking into account the context" UNED contribution to Deliverable D.31.3.1 MOSIL Project. Prepared for the European Commission, DG INFSO, under contract ITS 507838 as a deliverable from WP 31.

[Vélez & Celorrio, 2005]. Vélez J. Celorrio C. 2005. "Formatos de fichas y tipos de fichas". Informe Técnico Proyecto ENLACE. Tarea 1.2 (TIN 2004 - 04232)

[Vélez, 2006]. Vélez J. (2006). "Mecanismos de integración en Pelican". Informe Técnico Proyecto ENLACE. Tarea 3.2 (TIN 2004 - 04232)

[Vélez et al., 2006]. Vélez J. Barros B. Verdejo M. F. (2006). "A Framework to define Web Based Communities." International Journal of Web Based Communities 2006 Vol. 2, nº 3 pp. 339 - 359 ISSN: 1477-8394 IJWBC 2006 Journal

[Vélez, 2007]. Vélez, J. (2007). "Escenarios de integración en ENLACE". Informe Técnico Proyecto ENLACE. Tarea 3.2 (TIN 2004 - 04232)

[Vélez & Verdejo, 2007]. Vélez J. Verdejo M.F. (2007) "Modelado arquitectónico de la plataforma Pelican para la implantación de entornos colaborativos de aprendizaje", Revista Iberoamericana de Informática Educativa", Diciembre 2007 nº 6, págs. 47-62, ISSN 1699-4574

[Vélez & Verdejo, 2007b]. Vélez J. Verdejo M. F. (2007) "Una plataforma para la integración automática de herramientas de soporte a la colaboración" Actas del II Congreso Español de Informática (CEDI 2007). VIII Simposio Nacional de Tecnologías de la Información y las Comunicaciones en la Educación (SINTICE 2007). Zaragoza (España). Págs. 61-68 ISBN: 978-84-9732-597-4

[Verdejo et al., 2002]. Verdejo M. F. Barros B. Read T. Rodriguez M. (2002) "A system for the specification and development of an evironment for distributed CSCL scenarios". Cerri, S.A. Gouardères, G, Paraguaçu, F. (Eds.) ITS 2002 LNCS 2002. Spring-Verlag Berlin Heidelberg 2002.

[Verdejo et al., 2009] Verdejo M. F. Celorrio C. Lorenzo E. J. Millán, M. Prados, S. Vélez J. (2009) "Constructing mobile technology-enabled environments in support of an integrated learning approach". In Innovative Learning and Technologies. Hokyoung Ryu & David Parsons (Eds.) Chapter 8 pags. 145-172. ISBN: 978-1-60566-063-9

[Villasclaras et al., 2009]. Villasclaras-Fernández E.D. Hernández-Leo D. Asensio-Pérez J.I. Dimitriadis Y. "Incorporating assessment in a pattern-based design process for CSCL scripts". Computers in Human Behavior. , 2009.

[Vygotsky, 1978]. Vygotsky L.S. (1978) "Mind in society. The development of higher psychological processes". In M. Cole V. John–Steiner S. Scribner E. Souberman (Eds.) Harvard University Press: Cambridge, Massachusetts.

[Watson, 1913]. B. Watson (1913) "Psychology as the Behaviorist Views it". Psychological Review, 20, pags. 158-177.

[WebCT]. http://www.webct.com

[Webb, 1989]. Webb (1989) "Peer interaction and learning in small groups". International Journal of Educational Reseach. 13. pags. 21-39.

[Web et al. 1986]. Webb N.M. Ender P. Lewis S. (1986) "Problem–solving strategies and group processes in small groups learning computer programming". American Edicational Research Journal. 23 (2). pags. 243-261

[Weinberger, 2003]. Weinberger A. (2003) "Scripts for Computer–Suppoted Collaborative Learning. Effects of social and epistemic cooperation scripts on collaborative knowledge construction". Ph. D. Thesis. Munchen.

[Weinberger et al., 2005] Weinberger A Bernhard E. Fischer F. and Mandl H. (2004) "Epistemic and social scripts in computer–supported collaborative learning." In Instructional Science (2005) 33: 1-30 Springer.

[Wertsch, 1991]. Wertsch J. V. (1991) "A socio–cultural approach to socially shared cognition". In L. Resnick J. Levine & S. Teasley (Eds.) Perspectives on Socially Shared Cognition. Pags. 85-100. Hyattsville, MD: Americal Psycological Association.

[Westerinen, 2001] Westerinen A. Schnizlein J. Strassner J. Scherlin M. Quinn B. Herzog S. Huynh H. Carlson M. Perry J. Waldbusser S. (2001) "Terminology for Policy–Based Management". RFC 3189

[Wiley, 2000]. Wiley. D. A. (2000) "Connecting learning objects to instructional design theory: A definition, a metaphore and a taxonomy". On the Web at http://wesrac.usc.edu/wired/bldg-7_file/wiley.pdf

[Winograd & Flores, 1986]. Winograd T. Flores F. (1986) "Understanding Computers and Cognition: A New Foundation for Design". Intellect Books (Eds.).

[WfMC]. Workflow Management Coalition. "Workflow Process Definition Interface – XML Process Definition Language", WfMC-TC-1025, http://www.wfmc.org/standards/docs

[WSDL]. http://www.w3.org/TR/wsdl

[XMI]. http://www.omg.org/technology/documents/formal/xmi.htm

[WS–CDL]. http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217

[XSLT]. http://www.w3.org/TR/xslt

[YouTube]. http://www.youtube.com

A Integración de herramientas externas

Como se ha discutido a lo largo de esta tesis, Pelican puede ser considerada, entre otras perspectivas, como una plataforma de integración que permite a herramientas externas, potencialmente desarrolladas por terceras partes, incorporarse a de los espacios de trabajo de la misma. De esta manera, la plataforma se centra en proporcionar la infraestructura tecnológica de soporte a la formación de comunidades virtuales y al desarrollo de experiencias de aprendizaje colaborativo mediadas por computador mientras que las herramientas proporcionan servicios de interacción colaborativa. En esta división de responsabilidades simbiótica, la plataforma se enriquece de medios de interacción potentes y potencialmente adaptados a cada escenario particular mientras que las herramientas aprovechan las capacidades de gestión de los aspectos socio–colaborativos de Pelican.

Este apéndice va dirigido a los desarrolladores de herramientas externas que deseen integrar las mismas con la plataforma Pelican. No se pretende describir la forma de funcionamiento de los mecanismos de integración, ya que ellos fueron discutidos en detalle en la sección 3.6 de este trabajo. En su lugar, abordaremos, en estas páginas, la descripción de las mecanismos específicos de integración que han sido utilizados por las herramientas que forman parte del ecosistema tecnológico del proyecto ENLACE que, como dijimos en el capítulo 4, ha servido de marco para la experimentación en escenarios pedagógicos reales de aprendizaje colaborativo. La descripción de los mecanismos utilizados por estas herramientas servirá de ejemplo para ilustrar la forma de explotarlos en otras situaciones.

A.1. Introducción

Como se describió en la sección 3.6, Pelican proporciona cuatro mecanismos de integración que pueden ser utilizados por las herramientas externas para incorporarse a los espacios de trabajo de la plataforma y alcanzar distintos niveles de interoperabilidad. El más sencillo de ellos, la integración ligera o integración de nivel A, permite a los diseñadores instruccionales definir servicios de acceso a las herramientas adaptados a un contexto particular de uso. La integración dirigida por la plataforma o integración de nivel B, requiere que las herramientas externas publiquen una colección de servicios Web para la adaptación programática de sus modelos internos y opcionalmente emitan eventos externos para avisar a la plataforma de acontecimientos pedagógicamente relevantes ocurridos dentro de las mismas. De esta manera pueden definirse, dentro de la plataforma y a través del uso de las facilidades del subsistema de intervención, protocolos de orquestación que conecten unas herramientas con otras ofreciendo al usuario final una sensación de continuidad en el uso de las mismas. En la integración dirigida por las herramientas, o integración de nivel C, son las herramientas las encargadas de consultar periódicamente, mediante los servicios Web proporcionados por Pelican, el estado de los modelos internos de la plataforma y adaptarse puntualmente al mismo según sea conveniente.

A lo largo de este apéndice describiremos en detalle cómo pueden ser utilizadas estas capacidades de Pelican. En concreto, en la siguiente sección proporcionamos una relación de todas las operaciones que pueden ser accedidas desde los servicios Web proporcionados por la plataforma para facilitar la integración a nivel C. Y en la sección A.3, describiremos en concreto cómo cada herramienta del ecosistema tecnológico de ENLACE explota estas capacidades.

A.2. Servicios Web de Pelican

Pelican constituye una plataforma de integración donde una colección de herramientas externas potencialmente desarrolladas por terceras partes son incorporadas a los espacios de trabajo para ofrecer diferentes tipos de servicios de valor añadido relacionados con los requerimientos de interacción del escenario colaborativo donde aparecen implicados. Como se discutió a lo largo de la sección 3.6, esta integración se basa en distintos mecanismos que permiten establecer una comunicación bidireccional entre las herramientas y el entorno colaborativo. La figura A.1 ilustra este hecho.

Como puede apreciarse, las herramientas externas disponen de tres posibles alternativas apara acceder a los servicios Web de Pelican. En el de más bajo nivel (❶), la herramienta hace uso de su propias librerías internas para lanzar una solicitud a la plataforma. La respuesta será obtenida como un mensaje XML sobre SOAP que deberá ser digerido por ésta. Alternativamente, si la herramienta está implementada en Java, ésta puede incorporar a su código la librería *PelicanProxy.jar* que se proporciona con Pelican. El propósito de la misma es simplificar el proceso de comunicación con la plataforma. En este caso existen dos posibilidades: el programador de la herramienta puede utilizar la api de servicios Web (❷) o utilizar una api orientada a objetos de

más alto nivel (❸). La api de servicios Web ofrece una capa de transparencia para acceder a los servicios Web de Pelican devolviendo los resultados como mensajes XML puros. La api orientada a objetos, por su parte, proporciona las respuesta en forma de objetos java apoderados de los objetos de negocio de la plataforma y con comportamiento de carga perezosa.

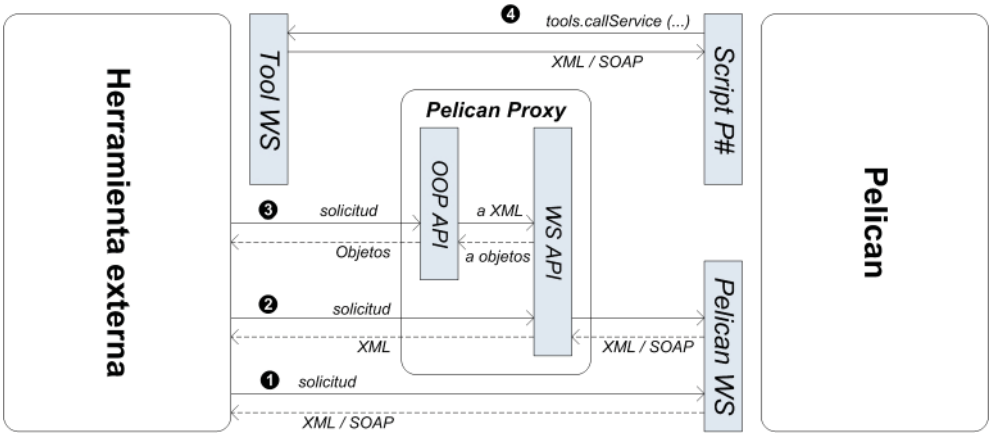

Figura A.1. Comunicación entre la plataforma y las herramientas integradas.

En el otro sentido, Pelican invoca los servicios Web de las herramientas externas a través de los scripts de adaptación que forman parte del modelo de intervención del escenario colaborativo (❹). En concreto se utiliza la tarea *callService*, perteneciente a la librería *tools* (consulte tabla 3.3). El resultado devuelto es un mensaje SOAP cuyo contenido puede venir descrito en diferentes formatos, dependiendo de la herramienta (típicamente XML). En cualquier caso es responsabilidad del script procesar este mensaje, aunque diversas tareas de la librería *tools* asisten a este proceso.

Pelican ofrece una colección de operaciones de acceso organizadas en 4 servicios Web con responsabilidades específicas: *SocialService, CollaborationService, PolicyService* y *SessionService* (consulte la sección 3.6.1 para obtener detalles al respecto). El punto de acceso a estos servicios, caso de que se utilice la interfaz ❶ de la figura A.1, es:

```
http://nombre-servidor:8080/pelican/services/nombre-servicio
```

donde *nombre-servidor* hace referencia al nombre DNS o dirección IP de la máquina donde se encuentra instalado Pelican (consulte el apéndice B para obtener detalles sobre la instalación) y *nombre-servicio* debe corresponderse con alguno de los cuatro nombres enumerados anteriormente, respetando mayúsculas y minúsculas.

Todas las interfaces de la figura A.1 para la comunicación con Pelican son exactamente iguales entre sí. Están formadas por operaciones con signaturas idénticas que sólo se diferencian en el formato de la respuesta. Para describir en conjunto estas interfaces, utilizaremos la tabla A.1. La primera columna muestra la cabecera común del método en todos los casos. La segunda describe su propósito. La tercera, la DTD del resultado para el caso de las interfaces ❶ y ❷. Y la cuarta, el tipo de objeto devuelto al utilizar la interfaz ❸.

Servicios Web de Pelican				
	Operación	Descrición	DTD del resultado en XML	Resultado en objetos
SocialService	getUser (long id)	Recupera un usuario por identificador	<!ELEMENT user (login, password, name, address, zipCode, country city, e-Mail, Phone)> <!ELEMENT login (#PCDATA)> <!ELEMENT password (#PCDATA)> <!ELEMENT name (#PCDATA)> <!ELEMENT address (#PCDATA)> <!ELEMENT zipCode (#PCDATA)> <!ELEMENT country (#PCDATA)> <!ELEMENT city (#PCDATA)> <!ELEMENT e-Mail (#PCDATA)> <!ELEMENT phone (#PCDATA)> <!ATTLIST user id CDATA #REQUIRED>	User
	getUser (String name)	Recupera un usuario por nombre		
	getAllUsers ()	Recupera todos los usuarios registrados	<!ELEMENT users (user*)> <!ELEMENT user EMPTY> <!ATTLIST user login CDATA #REQUIRED> <!ATTLIST user ref CDATA #REQUIRED>	List <User>
	getActor (long id)	Recupera un actor por identificador	<!ELEMENT actor (name, description, permissions)> <!ELEMENT name (#PCDATA)> <!ELEMENT description (#PCDATA)> <!ELEMENT permissions (permission*)> <!ELEMENT permission (#PCDATA)> <!ATTLIST user id CDATA #REQUIRED>	Actor
	getActor (String name)	Recupera un actor por nombre		
	getAllActors ()	Recupera todos los actores definidos	<!ELEMENT actors (actor*)> <!ELEMENT actor EMPTY> <!ATTLIST actor name CDATA #REQUIRED> <!ATTLIST actor ref CDATA #REQUIRED>	List <Actor>
	getAllActors (String login)	Recupera los actores que desempeña un usuario		
	getGroup (long id)	Recupera un grupo por identificador	<!ELEMENT group (name, description, groupTemplate, parentGroup, subgroups, actorBindings)> <!ELEMENT name (#PCDATA)> <!ELEMENT description (#PCDATA)> <!ELEMENT groupTemplate EMPTY> <!ELEMENT parentGroup EMPTY> <!ELEMENT subgroups (subgroup*)> <!ELEMENT actorBindings (actorBinding)> <!ELEMENT subgroup EMPTY> <!ELEMENT actorBinding (actor, group, users)> <!ELEMENT actor EMPTY> <!ELEMENT group EMPTY> <!ELEMENT users (user*)> <!ELEMENT user EMPTY> <!ATTLIST groupTemplate name CDATA #REQUIRED> <!ATTLIST groupTemplate ref CDATA #REQUIRED> <!ATTLIST parentGroup name CDATA #REQUIRED> <!ATTLIST parentGroup ref CDATA #REQUIRED> <!ATTLIST subgroup name CDATA #REQUIRED> <!ATTLIST subgroup ref CDATA #REQUIRED> <!ATTLIST actor name CDATA #REQUIRED> <!ATTLIST actor ref CDATA #REQUIRED> <!ATTLIST group name CDATA #REQUIRED> <!ATTLIST group ref CDATA #REQUIRED> <!ATTLIST user name CDATA #REQUIRED> <!ATTLIST user ref CDATA #REQUIRED>	Group
	getGroup (String name)	Recupera un grupo por nombre		
	getAllGroups ()	Recupera todos los grupos definidos	<!ELEMENT groups (group*)> <!ELEMENT group EMPTY> <!ATTLIST group name CDATA #REQUIRED> <!ATTLIST group ref CDATA #REQUIRED>	List <Group>
	getAllGroups(String login)	Recupera los grupos en que participa un usuario		
	getGroupTemplate (long id)	Recupera una plantilla de grupo por identificador	<!ELEMENT groupTemplate (name, description, parentTemplate, subgroupTemplates, actors, cardinality)> <!ELEMENT name (#PCDATA)> <!ELEMENT description (#PCDATA)> <!ELEMENT parentTemplate EMPTY> <!ELEMENT subgroupTemplates (subgroupTemplate*)> <!ELEMENT subgroupTemplate EMPTY> <!ELEMENT actors (actor*)> <!ELEMENT actor EMPTY> <!ELEMENT cardinality EMPTY> <!ATTLISTparentTemplate name CDATA #REQUIRED> <!ATTLISTparentTemplate ref CDATA #REQUIRED> <!ATTLIST subgroupTemplate name CDATA #REQUIRED> <!ATTLIST subgroupTemplate ref CDATA #REQUIRED> <!ATTLIST actor name CDATA #REQUIRED> <!ATTLIST actor ref CDATA #REQUIRED> <!ATTLIST cardinality maxAllowed CDATA #REQUIRED> <!ATTLIST cardinality minAllowed CDATA #REQUIRED>	GroupTemplate
	getGroupTemplate (String name)	Recupera una plantilla de grupo por nombre		
	getAllGroupTemplates ()	Recupera todos los plantillas de grupo definidos	<!ELEMENT groupTemplates (groupTemplate*)> <!ELEMENT groupTemplate EMPTY> <!ATTLIST groupTemplate name CDATA #REQUIRED> <!ATTLIST groupTemplate ref CDATA #REQUIRED>	List <GroupTemplate>
CollaborationService	getWorkspace (long id)	Recupera un espacio de trabajo por nombre	<!ELEMENT workspace (owner, users, projects, services)> <!ELEMENT owner (#PCDATA)> <!ELEMENT users (user*)> <!ELEMENT projects (project*)> <!ELEMENT services (service*)> <!ELEMENT user EMPTY> <!ELEMENT project EMPTY> <!ELEMENT service EMPTY> <!ATTLIST workspace id CDATA #REQUIRED> <!ATTLIST workspace private CDATA #REQUIRED> <!ATTLIST user name CDATA #REQUIRED> <!ATTLIST user ref CDATA #REQUIRED> <!ATTLIST project name CDATA #REQUIRED> <!ATTLIST project ref CDATA #REQUIRED> <!ATTLIST service type CDATA #REQUIRED> <!ATTLIST service label CDATA #REQUIRED> <!ATTLIST service ref CDATA #REQUIRED>	Workspace
	getAllWorkspaces ()	Recupera todos los espacios de trabajo	<!ELEMENT workspaces (workspace*)> <!ELEMENT workspace EMPTY> <!ATTLIST workspace owner CDATA #REQUIRED> <!ATTLIST workspace ref CDATA #REQUIRED>	List <Workspace>

Servicios Web de Pelican (continuación)				
	Operación	Descrición	DTD del resultado en XML	Resultado en objetos
CollaborationService	getProject (long id)	Recupera un proyecto por identificador	<!ELEMENT project (projectTemplate, status, startingDate, activities)> <!ELEMENT projectTemplate EMPTY> <!ELEMENT status (#PCDATA)> <!ELEMENT startingDate (#PCDATA)> <!ELEMENT activities (activity*)> <!ELEMENT activity EMPTY> <!ATTLIST project id CDATA #REQUIRED> <!ATTLIST activity name CDATA #REQUIRED> <!ATTLIST activity ref CDATA #REQUIRED>	Project
	getProject (String name)	Recupera un proyecto por nombre de plantilla		
	getAllProjects ()	Recupera todos los proyectos desplegados	<!ELEMENT projects (project*)> <!ELEMENT project EMPTY> <!ATTLIST project name CDATA #REQUIRED> <!ATTLIST project ref CDATA #REQUIRED>	List <Project>
	getAllProjects (String name)	Recupera todos los proyectos de una plantilla		
	getActivity (long id)	Recupera una actividad por identificador	<!ELEMENT activity (activityTemplate, startingDate, roleBindings, services, state)> <!ELEMENT activityTemplate EMPTY> <!ELEMENT startingDate (#PCDATA)> <!ELEMENT roleBindings (roleBinding*)> <!ELEMENT services (service*)> <!ELEMENT roleBinding (role, users)> <!ELEMENT state (#PCDATA)> <!ELEMENT role (name, description)> <!ELEMENT users (user*)> <!ELEMENT name (#PCDATA)> <!ELEMENT description (#PCDATA)> <!ELEMENT user EMPTY> <!ELEMENT service (type, label, description, parameters, state)> <!ELEMENT type (#PCDATA)> <!ELEMENT label (#PCDATA)> <!ELEMENT description (#PCDATA)> <!ELEMENT parameters (parameter*)> <!ELEMENT state (#PCDATA)> <!ELEMENT parameter EMPTY> <!ATTLIST activity id CDATA #REQUIRED> <!ATTLIST activityTemplate name CDATA #REQUIRED> <!ATTLIST activityTemplate ref CDATA #REQUIRED> <!ATTLIST role id CDATA #REQUIRED> <!ATTLIST user name CDATA #REQUIRED> <!ATTLIST user ref CDATA #REQUIRED> <!ATTLIST parameter name CDATA #REQUIRED> <!ATTLIST parameter value CDATA #REQUIRED>	Activity
	getActivity (String name)	Recupera una actividad por nombre de plantilla		
	getAllActivities ()	Recupera todos las actividades desplegadas	<!ELEMENT activities (activity*)> <!ELEMENT activity EMPTY> <!ATTLIST activity name CDATA #REQUIRED> <!ATTLIST activity ref CDATA #REQUIRED>	List <Activity>
	getAllActivities (String name)	Recupera todos las actividades de una plantilla		
	getActivityTemplate (long id)	Recupera una plantilla de actividad por identificador	<!ELEMENT activityTemplate (name, description, references,roles, services)> <!ELEMENT name (#PCDATA)> <!ELEMENT description (#PCDATA)> <!ELEMENT references (reference*)> <!ELEMENT roles (role*)> <!ELEMENT services (#service*)> <!ELEMENT reference (name, description, uri)> <!ELEMENT role (name, description)> <!ELEMENT service EMPTY> <!ELEMENT uri (#PCDATA)> <!ATTLIST activity id CDATA #REQUIRED> <!ATTLIST reference id CDATA #REQUIRED> <!ATTLIST role id CDATA #REQUIRED> <!ATTLIST service type CDATA #REQUIRED> <!ATTLIST service label CDATA #REQUIRED> <!ATTLIST service ref CDATA #REQUIRED>	ActivityTemplate
	getActivityTemplate (String name)	Recupera una plantilla de actividad por nombre		
	getAllActivityTemplates ()	Recupera todas las plantillas de actividad	<!ELEMENT activityTemplates (activityTemplate*)> <!ELEMENT activityTemplate EMPTY> <!ATTLIST activityTemplate name CDATA #REQUIRED> <!ATTLIST activityTemplate ref CDATA #REQUIRED>	List <ActivityTemplate>
	getProjectTemplate (long id)	Recupera una plantilla de proyecto por identificador	<!ELEMENT projectTemplate (name, description, activityTemplates)> <!ELEMENT name (#PCDATA)> <!ELEMENT description (#PCDATA)> <!ELEMENT activityTemplates (activityTemplate*)> <!ELEMENT activityTemplate EMPTY> <!ATTLIST projectTemplate id CDATA #REQUIRED> <!ATTLIST activityTemplate name CDATA #REQUIRED> <!ATTLIST activityTemplate ref CDATA #REQUIRED>	ProjectTemplate
	getProjectTemplate (String name)	Recupera una plantilla de proyecto por nombre		
	getAllProjectTemplates ()	Recupera todas las plantillas de proyecto	<!ELEMENT projectTemplates (projectTemplate*)> <!ELEMENT projectTemplate EMPTY> <!ATTLIST projectTemplate name CDATA #REQUIRED> <!ATTLIST projectTemplate ref CDATA #REQUIRED>	List <ProjectTemplate>
	getReference (long id)	Recupera una referencia por identificador	<!ELEMENT reference (name, description, uri)> <!ELEMENT name (#PCDATA)> <!ELEMENT description (#PCDATA)> <!ELEMENT uri (#PCDATA)> <!ATTLIST reference ref CDATA #REQUIRED>	Reference
	getReference (String name)	Recupera una referencia por nombre		
	getAllReferences ()	Recupera todas las referencias definidas	<!ELEMENT references (reference*)> <!ELEMENT reference EMPTY> <!ATTLIST reference name CDATA #REQUIRED> <!ATTLIST reference ref CDATA #REQUIRED>	List <Reference>
	getRole (long id)	Recupera un rol por identificador	<!ELEMENT role (name, description, properties)> <!ELEMENT name (#PCDATA)> <!ELEMENT description (#PCDATA)> <!ELEMENT properties (property*)> <!ELEMENT property EMPTY> <!ATTLIST role ref CDATA #REQUIRED> <!ATTLIST property name CDATA #REQUIRED> <!ATTLIST property value CDATA #REQUIRED>	Role
	getRole (String name)			
		Recupera un rol por nombre		
	getAllRoles ()	Recupera todos los roles definidos	<!ELEMENT roles (role*)> <!ELEMENT role EMPTY> <!ATTLIST role name CDATA #REQUIRED> <!ATTLIST role ref CDATA #REQUIRED>	List <Role>
	getAllRoles (String name)	Recupera los roles de una plantilla de actividad		

Servicios Web de Pelican (continuación)				
	Operación	Descrición	DTD del resultado en XML	Resultado en objetos
CollaborationService	getService (long id)	Recupera un servicio por identificador	<!ELEMENT service (type, label, description, parameters, state)> <!ELEMENT type (#PCDATA)> <!ELEMENT label (#PCDATA)> <!ELEMENT description (#PCDATA)> <!ELEMENT parameters (parameter*)> <!ELEMENT state (#PCDATA)> <!ELEMENT parameter EMPTY> <!ATTLIST reference id CDATA #REQUIRED> <!ATTLIST parameter name CDATA #REQUIRED> <!ATTLIST parameter value CDATA #REQUIRED>	Service
	getService (String name)	Recupera un servicio por nombre		
	getAllServices ()	Recupera todos los servicios definidos	<!ELEMENT services (service*)> <!ELEMENT service EMPTY> <!ATTLIST service name CDATA #REQUIRED> <!ATTLIST service ref CDATA #REQUIRED>	List <Service>
PolicyService	getPolicy (long id)	Recupera una política por identificador	<!ELEMENT policy (name, description, policyRules)> (*) <!ELEMENT name (#PCDATA)> <!ELEMENT description (#PCDATA)> <!ELEMENT policyRules (policyRule*)> <!ELEMENT policyRule (name, description, event, trigger, script)> <!ELEMENT event (#PCDATA)> <!ELEMENT trigger (#PCDATA)> <!ELEMENT script (#PCDATA)> <!ATTLIST policy id CDATA #REQUIRED> <!ATTLIST policyRule id CDATA #REQUIRED>	Policy
	getPolicy (String name)	Recupera una política por nombre		
	getAllPolicies ()	Recupera todas políticas definidas	<!ELEMENT policies (policy*)> <!ELEMENT policy EMPTY> <!ATTLIST policy name CDATA #REQUIRED> <!ATTLIST policy ref CDATA #REQUIRED>	List <Policy>
	installPolicy (String xmlPolicy (*))	Instala una nueva regla política	-	void
	uninstallPolicy (longid)	Desinstala una nueva regla política		
	executePolicyScript (String login, String script)	Ejecuta un script sobre un contexto de		
SessionService	getCurrentUser (String login)	Recupera el usuario en curso	Mismo formato que en getUser (long id)	User
	getCurrentActor(String login)	Recupera el actor en curso	Mismo formato que en getActor (long id)	Actor
	getCurrentGroup (String login)	Recupera el grupo en curso	Mismo formato que en getGroup (long id)	Group
	getCurrentGroupTemplate (String login)	Recupera la plantilla de grupo en curso	Mismo formato que en getGroupTemplate (long id)	GroupTemplate
	getCurrentWorkspace (String login)	Recupera el espacio de trabajo en curso	Mismo formato que en getWorkspace (long id)	Workspace
	getCurrentProject (String login)	Recupera el proyecto en curso	Mismo formato que en getProject (long id)	Project
	getCurrentProjectTemplate (String login)	Recupera la plantilla de proyecto en curso	Mismo formato que en getProjectTemplate (long id)	ProjectTemplate
	getCurrentActivity (String login)	Recupera la actividad en curso	Mismo formato que en getActivity (long id)	Activity
	getCurrentActivityTemplate (String login)	Recupera la plantilla de actividad en curso	Mismo formato que en getActivityTemplate (long id)	ActivityTemplate
	getCurrentRole (String login)	Recupera el rol en curso	Mismo formato que en getRole (long id)	Role

Tabla A.1. Servicios Web de Pelican.

A.3. Integración de herramientas de ENLACE

A lo largo de esta sección describiremos, para cada una de las herramientas que forman parte del ecosistema tecnológico de ENLACE, la manera en que se integran con Pelican en cada uno de los 3 niveles (A, B y C) citados en la introducción. En el nivel A proporcionaremos el detalle de todos los servicios definidos de la herramienta en cuestión adjuntando, para cada uno de ellos, los parámetros constituyentes de su esquema de invocación asociado. En el nivel B, si procede presentaremos la colección de servicios Web proporcionados por la herramienta y en el nivel C, detallaremos la colección de eventos externos que emita la misma.

A.3.1. La herramienta LOR

A. Integración Ligera

La herramienta LOR es un repositorio de objetos de aprendizaje que proporciona servicios de almacenamiento (carga y descarga) y control de versiones de los mismos. La tabla A.2 describe los servicios proporcionados por LOR.

Servicio		Esquema		
Nombre	Descripción	Parámetro	Descripción	Valor por defecto
Browse	Permite navegar por un repositorio y acceder a los objetos almacenados	User	Login de un usuario con acceso	$currentUser.login
		Password	Clave de un usuario con acceso	$currentUser.password
		Repository Name	Nombre del repositorio al que se accede	-
View	Muestra el contenido de un objeto almacenado en el repositorio	User	Login de un usuario con acceso	$currentUser.login
		Password	Clave de un usuario con acceso	$currentUser.password
		Id	Identificador del objeto	-
Upload	Permite subir un contenido como objeto de aprendizaje al repositorio	User	Login de un usuario con acceso	$currentUser.login
		Password	Clave de un usuario con acceso	$currentUser.password
		Repository Name	Nombre del repositorio al que se accede	-
		LO Type	Nombre del tipo de objeto	-
Download	Permite descargar el contenido de un objeto de aprendizaje del repositorio	User	Login de un usuario con acceso	$currentUser.login
		Password	Clave de un usuario con acceso	$currentUser.password
		Id	Identificador del objeto	-

Tabla A.2. Servicios proporcionados por la herramienta LOR.

B. Integración dirigida por la plataforma

La herramienta LOR alcanza un nivel de interoperabilidad elevado con Pelican puesto que ofrece una colección de operaciones en forma de servicios Web que permiten hacer una integración dirigida por la plataforma. En concreto este sistema proporciona dos servicios ideados para trabajo y administración respectivamente cuyos puntos de acceso son los siguientes. En la tabla E.2 se resumen estas operaciones.

http:*//server-name:8080/lor/services/LORService*
http:*//server-name:8080/lor/services/LORAdminService*

	Servicios Web de LOR			
	Cabecera	Descripción	DTD del resultado en XML	Resultado en objetos
LORService	login(String login, String password)	Abre una sesión de trabajo para operar con el servicio	<!ELEMENT success (logged)> <!ELEMENT logged (#PCDATA)>	String
	logger ()	Determina si la sesión de trabajo está abierta		
	downloadLO (String id)	Recupera el contenido de un objeto de aprendizaje por identificador	-	byte []
	downloadLastVersionOfLO (String id)	Recupera la última versión de un objeto por identificador		
	uploadLO (byte[] data, String fileName, String type, String repositoryId)	Sube un objeto de aprendizaje al un repositorio	<!ELEMENT success (#PCDATA)>	String
	uploadNewVersion (byte[] data, String fileName, String loId)	Sube un objeto de aprendizaje como una nueva versión de otro	<!ELEMENT success (#PCDATA)>	
	uploadModification (byte[] data, String fileName, String loId)	Sobrescribe un objeto de aprendizaje	<!ELEMENT success (#PCDATA)>	
	searchLOR (String query) (query sigue la DTD de la respuesta)	Devuelve una colección de referencias a objetos de aprendizaje	<!ELEMENT query (entity*)> <!ELEMENT entity (slot, value*)> <!ELEMENT slot EMPTY> <!ELEMENT value (#CDATA)> <!ATTLIST slot name CDATA required>	
	deleteLO (String loId)	Borra un objeto del repositorio	<!ELEMENT success (#PCDATA)>	
	killLO (String loId)	Borra un objeto del repositorio de forma irrecuperable	<!ELEMENT success (#PCDATA)>	
	copyLO (String loId, String repositoryId)	Copia un objeto de aprendizaje a un repoisitorio	<!ELEMENT success (#PCDATA)>	
	revertLOVersion (String loId)	Convierte un objeto a su versión anterior	<!ELEMENT success (#PCDATA)>	

Servicios Web de LOR (continuación)				
	Cabecera	Descripción	DTD del resultado en XML	Resultado en objetos
LORService	getAllValuesOfMetadata (String mdItem, String repositoryId)	Recupera todos los valores asociados a un metadato	<!ELEMENT success (metadatavalues)> <!ELEMENT metadatavalues (value*)> <!ELEMENT value (#PCDATA)>	String
	getAllValuesOfMetadata (String mdItem)			
	getLearningObjectType (String type)	Devuelve la descripción de un tipo de objeto de aprendizaje	<!ELEMENT success (lotypes)> <!ELEMENT lotypes (type*)> <!ELEMENT type EMPTY> <!ATTLIST type name CDATA required> <!ATTLIST type mime CDATA required> <!ATTLIST type toolUrl CDATA required> <!ATTLIST type toolIcon CDATA required>	
	getRepositoryById (String repositoryId)	Recupera un repositorio por identificador	<!ELEMENT success (repositories)> <!ELEMENT repositories (repository*)> <!ELEMENT repository (description)> <!ELEMENT description (#PCDATA)> <!ATTLIST repository id CDATA required> <!ATTLIST repository name CDATA required>	
	getRepositoryById (String name)	Recupera un repositorio por nombre		
LORService	createRepository (String name, String description)	Crea un nuevo repositorio		
	editRepository (String repositoryId, String name, String description)	Edita las propiedades de un repositorios		
	deleteRepository (String repositoryId)	Borra un repositorio	<!ELEMENT success (repository)> <!ELEMENT repository EMPTY> <!ATTLIST repository id CDATA required>	
	grantUserPermission (String login, String repositoryId, String level)	Concede un nivel de permisos a un usuario de un repositorio	<!ELEMENT success (#PCDATA)>	
	grantGroupPermission (String login, String repositoryId, String level)	Concede un grupo de permisos a un usuario de un repositorio		
	removeUserPermission (String login, String repositoryId)	Elimina un nivel de permisos a un usuario de un repositorio		
	removeGroupPermission (String login, String repositoryId)	Elimina un grupo de permisos a un usuario de un repositorio		
	login(String login, String password)	Abre una sesión de trabajo para operar con el servicio	<!ELEMENT success (logged)> <!ELEMENT logged (#PCDATA)>	
	logger ()	Determina si la sesión de trabajo está abierta		

Tabla A.3. Familia de eventos de la herramienta LOR.

LOR también adquiere un grado de compromiso con la plataforma para alcanzar un nivel de interoperabilidad elevado con ésta. En concreto cada vez que se produzca en el repositorio un acontecimiento relevante se emite un evento externo para que sea procesado Por Pelican. En concreto LOR distingue dos tipos de eventos: los eventos transaccionales, generados cuando los estudiantes interactúan con la herramienta y los de administración propios de tareas de administración. La tabla A.4 los resume.

C. Integración dirigida por la herramienta

La herramienta LOR utiliza los mecanismos de integración dirigidos por la herramienta para comprobar la autenticidad de las credenciales de usuario que le son proporcionadas en la invocación de los servicios consultando la información del subsistema social accesible desde los servicios Web de Pelican. Este proceso, constituye una medida de seguridad ya que, aunque en la mayoría de las ocasiones los valores de las credenciales son proporcionados por la propia plataforma a través de la integración ligera en otras ocasiones la invocación puede ser directa (recuérdese que LOR es una herramienta autónoma y agnóstica del agente que invoca sus servicios).

Eventos externos de *LOR*		
Tipo	**Lanzado cuando...**	**Atributos**
lor.transaction.download	Se descarga un objeto de aprendizaje del repositorio	- learningObject.name: El nombre del objeto - learningObject.id: El identificador del objeto - learningObject.type: El tipo del objeto
lor.transaction.delete	Se elimina un objeto de aprendizaje del repositorio	- repositoryName: El nombre del repositorio - repositoryId: El identificador del repositorio - learningObject.name: El nombre del objeto - learningObject.id: El identificador del objeto - learningObject.type: El tipo del objeto
lor.transaction.upload	Se carga un objeto de aprendizaje en el repositorio	- repositoryName: El nombre del repositorio - repositoryId: El identificador del repositorio - learningObject.name: El nombre del objeto - learningObject.id: El identificador del objeto - learningObject.type: El tipo del objeto
lor.transaction.seach	Se realiza una búsqueda de objetos de aprendizaje sobre todos los repositorios	- repositoryQuery: La consulta lanzada - repositoryResults: Los resultados obtenidos
lor.transaction.copy	Se copia un objeto de aprendizaje de un repositorio a otro	- learningObject.name: El nombre del objeto - learningObject.id: El identificador del objeto - learningObject.type: El tipo del objeto - repository.target.id: Identificador repositorio de destino - repository.target.name: Nombre repositorio de destino - repository.source.id: Identificador repositorio origen - repository.source.name: Nombre repositorio destino
lor.transaction.move	Se mueve un objeto de aprendizaje de un repositorio a otro	- learningObject.name: El nombre del objeto - learningObject.id: El identificador del objeto - learningObject.type: El tipo del objeto - repository.id: Identificador repositorio de destino - repository.name: Nombre repositorio de destino
lor.transaction.newVersion	Se crea una nueva versión de un objeto de aprendizaje	- learningObject.name: El nombre del objeto - learningObject.id: El identificador del objeto - learningObject.version: La versión del objeto - learningObject.type: El tipo del objeto - repository.id: Identificador repositorio - repository.name: Nombre repositorio
lor.transaction.revert	Se vuelve a la versión anterior del objeto de aprendizaje	- learningObject.name: El nombre del objeto - learningObject.id: El identificador del objeto - learningObject.version: La versión del objeto - learningObject.type: El tipo del objeto - repository.id: Identificador repositorio - repository.name: Nombre repositorio
lor.transaction.kill	Se elimina de forma permanente un objeto de aprendizaje	- learningObject.name: El nombre del objeto - learningObject.id: El identificador del objeto - learningObject.type: El tipo del objeto - repository.id: Identificador repositorio - repository.name: Nombre repositorio
lor.transaction.overwrite	Se sobrescribe un objeto de aprendizaje con nuevo contenido	- learningObject.name: El nombre del objeto - learningObject.id: El identificador del objeto - learningObject.type: El tipo del objeto - repository.id: Identificador repositorio - repository.name: Nombre repositorio

Tabla A.4. Familia de eventos de la herramienta LOR.

A.3.2. La herramienta AGORA

A. Integración ligera

La herramienta AGORA da soporte a procesos de negociación basados en votación. Aunque proporciona varios perfiles de uso, tal y como se describió en la sección 4.2 (monitor, candidato, y estudiante), desde el punto de vista de Pelican la herramienta ofrece un único servicio tal y como se muestra en la tabla A.5.

Servicio		Esquema		
Nombre	Descripción	Parámetro	Descripción	Valor por defecto
Agora	Permite realizar una sesión de votación	User Id	Identificador del usuario participante	$currentUser.id
		Group Id	Identificador del grupo implicado	$currentGroup.id

Tabla A.5. Servicios de la herramienta AGORA.

B. Integración dirigida por la plataforma

Agora es la otra herramienta del ecosistema tecnológico de ENLACE que explota el mecanismo de integración dirigido por la plataforma. Las operaciones proporcionadas por esta herramienta, que aparecen descritas en la tabla A.6, se centran en la gestión de las sesiones de votación y son accesibles a través del punto de acceso siguiente, donde *server–name* hace referencia al nombre DNS o dirección IP del servidor donde está instalada la herramienta.

http:*//server-name:8080/agora/services/AgoraService*

	Servicios Web de AGORA			
	Cabecera	Descripción	DTD del resultado en XML	Resultado en objetos
AgoraService	createNewSession (String name, String login, long groupId)	Crea una nueva sesión de votación	<!ELEMENT AgoraService (#PCDATA)>	String
	addStudentAsCandidate (long sessionId, String login)	Añade un estudiante como candidato de la sesión de votación		
	addAllStudentsAsCandidates (long sessionId)	Añade a todos los estudiantes del grupo como candidatos		
	addTextCandidate (long sessionId, String text)	Añade un texto como candidato a la sesión de votación		
	addImageCandidate (long sessionId, String imageUrl)	Añade una imagen como candidato a la sesión de votación		
	addExternalCandidate (long sessionId, String login, String resourceId)	Añade una recurso externo como candidato a la sesión de votación		
	deleteSession (long sessionId)	Borra una sesión de votación		
	setStudentParticipationMode (long sessionId, String mode)	Establece el modo de participación de la sesión		

Tabla A.6. Servicios Web de la herramienta AGORA.

AGORA también ofrece una colección de eventos para informar a la plataforma de los acontecimientos relevantes ocurridos durante la votación. En concreto, los eventos que AGORA comunica a Pelican permiten hacer un seguimiento de todas las fases de desarrollo por las que atraviesa una sesión de votación. Tal vez los eventos más útiles y frecuentemente utilizados son aquéllos que indican el final de una votación o el alcance de una situación de empate. En la tabla A.7 se describen en detalle.

Eventos externos de *AGORA*		
Tipo	**Lanzado cuando...**	**Atributos**
agora.session.create	Se crea una nueva sesión de votación	- groupId: Identificador del grupo implicado - sessionId: Identificador de la nueva sesión
agora.session.start	Comienza la sesión de votación	- groupId: Identificador del grupo implicado - sessionId: Identificador de la nueva sesión
agora.session.entering.state.start	Se presenta la votación. Los estudiantes ya pueden acceder a la misma	- groupId: Identificador del grupo implicado - sessionId: Identificador de la nueva sesión
agora.session.candidate.selection.add.all.students.as.candidates	Los estudiantes presentan sus candidatos	- groupId: Identificador del grupo implicado - sessionId: Identificador de la nueva sesión
agora.session.candidate.selection.remove.candidate	Se elimina un candidato	- groupId: Identificador del grupo implicado - sessionId: Identificador de la nueva sesión - imageCandidateDescription: Imagen candidata borrada - textCandidate: Texto candidato borrado - imageCandidateUrl: URI candidata borrada - cardCandidate: Ficha candidata borrada
agora.session.candidate.selection.remove.all.candidates	Se borran todos los candidatos de la sesión	- groupId: Identificador del grupo implicado - sessionId: Identificador de la nueva sesión
agora.session.exists.draw	Se llega a un empate en la votación	- groupId: Identificador del grupo implicado - sessionId: Identificador de la nueva sesión
agora.session.end	Se alcanza el final de la votación	- groupId: Identificador del grupo implicado - sessionId: Identificador de la nueva sesión - loId: identificador del LO con los resultados

Tabla A.7. Familia de eventos de la herramienta AGORA.

C. Integración dirigida por la herramienta

AGORA hace uso del mecanismo de integración dirigido por la herramienta para obtener información del subsistema social de Pelican relativa a la composición del grupo que participa en la sesión de votación.

A.3.3. La herramienta CHAT

A. Integración ligera

La herramienta CHAT ofrece un servicio de discusión que puede ser configurado en diversos modos de interacción: debate abierto (*SIMPLE_CHAT*), para permitir que todos los usuarios participen libremente, debate por turnos (*TURN_BASED_CHAT*), donde cada estudiante debe esperar a que se le conceda el turno de palabra para intervenir y debate por solicitud de turno (*TURN_REQUEST_BASED_CHAT*) donde cada usuario tiene turno de palabra solo si lo solicita. La tabla A.8 muestra este servicio.

Servicio		Esquema		
Nombre	Descripción	Parámetro	Descripción	Valor por defecto
Chat	Permite realizar sesiones de debate con varios modos de participación	Nick	Nombre del usuario en la sesión	$currentUser.login
		Room	Nombre de la habitación de discusión	$currentUser.password
		mode	Modo de debate	SIMPLE_CHAT
		Events	Flag para indicar si se envían eventos	false

Tabla A.8. Servicios de la herramienta CHAT.

B. Integración dirigida por la plataforma

Los eventos de la herramienta CHAT permiten hacer un seguimiento de la interacción síncrona que los usuarios hacen sobre ella. La tabla A.9 describe todos ellos.

Eventos externos de *CHAT*		
Tipo	**Lanzado cuando...**	**Atributos**
Chat.login	El usuario entra en el salón de chat	- user: Login del usuario que entra al chat - mode: Modo de charla - room: Identificador de habitación a la que entra
chat.send.message	El usuario escribe un mensaje	- user: Login del usuario que entra al chat - room: Identificador de habitación a la que entra - message: Mensaje del usuario
chat.turn.request	El usuario solicita un turno de intervención	- user: Login del usuario que entra al chat - room: Identificador de habitación a la que entra
chat.turn.release	El tiempo de intervención del usuario en el turno de palabra finaliza	- user: Login del usuario que entra al chat - room: Identificador de habitación a la que entra
chat.send.permissionChange	Se cambian los permisos de acceso de un usuario (lectura / escritura) al chat	- user: Login del usuario que entra al chat - room: Identificador de habitación a la que entra - permission: Permiso mofificado - value: Nuevo valor del permiso

Tabla A.9. Familia de eventos de la herramienta CHAT.

C. Integración dirigida por la herramienta

CHAT consulta el rol con el que acceden los usuarios al debate y se adapta en función de dos propiedades lógicas del rol: *chat.read* y *chat.write* que indican sí el usuario puede o no recibir y enviar mensajes de otros usuarios respectivamente.

A.3.4. La herramienta CARDS

A. Integración ligera

La herramienta CARDS permite diseñar y rellenar fichas para registrar datos de las observaciones fruto del desarrollo de las actividades colaborativas. Desde el punto de vista de Pelican, la herramienta ofrece 3 servicios resumidos en la tabla A.10.

Servicio		Esquema		
Nombre	Descripción	Parámetro	Descripción	Valor por defecto
Design	Permite diseñar una nueva plantilla de ficha	User	Login de un usuario con acceso	$currentUser.login
		Password	Clave de un usuario con acceso	$currentUser.password
		Repository Name	Repositorio que almacena la ficha	-
		Encoded	Indica si los parámetros se encriptan	no
New	Permite rellenar una nueva ficha	User	Login de un usuario con acceso	$currentUser.login
		Password	Clave de un usuario con acceso	$currentUser.password
		Repository Name	Repositorio que almacena la ficha	-
		Template Name	Nombre de la plantilla de ficha	-
		Encoded	Indica si los parámetros se encriptan	no
Edit	Permite modificar una ficha que ya existente	User	Login de un usuario con acceso	$currentUser.login
		Password	Clave de un usuario con acceso	$currentUser.password
		Repository Name	Repositorio que almacena la ficha	-
		Template Version	Número de versión de la plantilla de ficha	-
		Id	Identificar de la ficha a editar	-
		Encoded	Indica si los parámetros se encriptan	no

Tabla A.10. Servicios de la herramienta CARDS.

B. Integración dirigida por la plataforma

La herramienta CARDS no emite eventos externos ni tampoco ofrece una familia de servicios Web con lo cual no es posible hacer uso de los mecanismos de integración dirigidos por la plataforma para este caso.

C. Integración dirigida por la herramienta

La herramienta CARDS utiliza los servicios Web de Pelican para autentificar las credenciales de usuario que le son proporcionadas como parámetros en la invocación de sus servicios, similarmente a como lo hace LOR. Además en sus últimas versiones CARDS adapta su comportamiento en función de la propiedad de rol *cards.mode* que puede tener valores *edit* o *design*. En el primer caso, el usuario tiene permisos solamente para editar fichar fichas cuyas plantillas han sido creadas por usuarios accediendo en el modo de diseño.

A.3.5. La herramienta COMPO

A. Integración Ligera

La herramienta COMPO permite a los diseñadores definir un itinerario sobre un mapa para permitir a los estudiantes incluir fichas CARDS relacionadas con observaciones encontradas a lo largo del mismo. En Pelican esta funcionalidad se articula a partir de los servicios que se describen en la tabla A.11.

Servicio		Esquema		
Nombre	Descripción	Parámetro	Descripción	Valor por defecto
Compo	Permite rellenar un itinerario con COMPO	Login	Login de un usuario con acceso	$currentUser.login
		Password	Clave de un usuario con acceso	$currentUser.password
		Repository Name	Repositorio que almacena los objetos	-
		Composition Name	Nombre del objeto de composicion	-
		Image Name	Nombre del la imagen (mapa) de fondo	-
		Tipo	Tipo de usuario	OWNER
		Max propousal time	Tiempo máximo de propuestas	15
		Max Board time	Tiempo máximo de pizarra	15
Admin Compo	Permite administrar COMPO	Login	Login de un usuario con acceso	$currentUser.login
		Password	Clave de un usuario con acceso	$currentUser.password

Tabla A.11. Servicios de la herramienta COMPO.

B. Integración dirigida por la plataforma

COMPO no hace uso de las capacidades que proporciona Pelican para realizar integración dirigida por la plataforma puesto que no emite eventos externos ni tampoco proporciona una familia de servicios Web.

C. Integración dirigida por la herramienta

Esta herramienta, al igual que otras pertenecientes al ecosistema tecnológico de ENLACE como LOR o CARDS, utilizan integración dirigida por la herramienta en la fase de autentificación de las credenciales de usuario.

A.3.6. La herramienta COMET

A. Integración Ligera

COMET es una herramienta para la creación colaborativa de mapas conceptuales que puede incluir como elementos objetos de aprendizaje almacenados en el repositorio LOR, entre los que destacan fichas de CARDS. Desde Pelican el acceso a esta herramienta se hace a través de un servicio cuyo esquema de invocación pide únicamente las credenciales del usuario (tabla A.12).

Servicio		Esquema		
Nombre	Descripción	Parámetro	Descripción	Valor por defecto
Comet	Permite acceder a la herramienta COMET	Login	Login de un usuario con acceso	$currentUser.login
		Password	Clave de un usuario con acceso	$currentUser.password

Tabla A.12. Servicios de la herramienta COMET.

B. Integración dirigida por la plataforma

La herramienta COMET no hace uso de las facilidades del mecanismo de integración dirigida por la plataforma puesto que no proporciona eventos ni servicios Web para su control.

C. Integración dirigida por la herramienta

De forma similar COMET hace integración dirigida por la herramienta únicamente para autentificar las credenciales del usuario en los parámetros de invocación del servicio.

B Instalación y configuración de Pelican

El entorno de aprendizaje colaborativo Pelican que hemos presentado a lo largo de esta tesis es, desde el punto de vista tecnológico, una aplicación Web implementada sobre la plataforma de desarrollo JEE de [Java] que permite a un conjunto de usuarios potencialmente elevado interactuar entre sí de forma sencilla y ubicua sin necesidad de realizar ninguna instalación previa en la máquina local del cliente.

Como contrapartida, estas soluciones siguen un modelo arquitectónico cliente–servidor que tiende a delegar todo el trabajo de configuración y administrador en la parte servidora. Esto implica que, para instalar Pelican, los administradores deben seguir una serie de pasos no necesariamente inmediatos para un usuario con perfil no técnico.

Por su parte, la mayoría de las propiedades de la plataforma Pelican pueden ser declarativamente configuradas en frío mediante la edición de uno o varios fichos dispuestos a tal efecto. Esta tampoco es una tarea evidente si no se conocen los detalles de funcionamiento interno del sistema.

Precisamente por estos motivos hemos querido incluir en este trabajo de tesis este último apéndice, que explicará como instalar paso a paso la plataforma de aprendizaje colaborativo Pelican para llevarla a explotación en un entorno real. Asimismo se describirán aquí los principales elementos de configuración que pueden ser modificados por los administradores – antes del arranque de la plataforma – para definir el comportamiento de Pelican.

B.1. Introducción

Para terminar con la colección de apéndices que describen la plataforma de aprendizaje colaborativo Pelican en su dimensión más tecnológica, hemos creído conveniente dedicar un último espacio a explicar brevemente cómo puede ser llevado este sistema a explotación en un entorno de producción real. En la siguiente sección trataremos de ofrecer una descripción de los pasos que es necesario llevar a cabo para instalar la plataforma en una máquina servidora. La última sección de este apéndice, por su parte, se centrará en discutir los principales ficheros de configuración que pueden ser modificados por los administradores antes del arranque de la plataforma para ajustar algunos aspectos específicos de la misma.

B.2. Instalación de la plataforma Pelican

Desde el punto de vista tecnológico Pelican es una aplicación Web implantada sobre la plataforma de desarrollo JEE de [Java]. Esto trae consigo, como principal ventaja, que el trabajo de instalación que deben hacer los usuarios en sus máquinas locales es nulo, fomentándose así la utilización ubicua e independiente de dispositivo del entorno de aprendizaje.

Esta aproximación tecnológica impone, no obstante, el modelo arquitectónico cliente–servidor, que en el caso de Pelican además hace uso de un sistema gestor de bases de datos relacional para persistir toda la información mantenida por la aplicación. En suma todo ello implica que, para instalar Pelican necesitemos disponer de dos máquinas servidoras: una con visibilidad en el espacio de Internet donde se desea desplegar la plataforma que llamaremos *servidor de aplicaciones* y otra con acceso por red desde la primera donde instalar el sistema gestor de bases de datos, que llamaremos *servidor de bases de datos* (ambas máquinas pueden y típicamente suelen ser la misma). Seleccionadas estas dos máquinas la instalación de Pelican puede resumirse en los siguientes pasos:

- **Instalación del contenedor de aplicaciones Web.** El primer paso para instalar Pelican consiste en seleccionar un contenedor de aplicaciones Web e instalarlo en el servidor de aplicaciones siguiendo las instrucciones del fabricante. Un contenedor de aplicaciones es una herramienta software encargada de gestionar el ciclo de vida de las aplicaciones Web instaladas sobre él (donde posteriormente instalaremos Pelican). Existen en el mercado numerosos productos de este tipo. Nosotros en nuestras pruebas hemos utilizado el contenedor Web de libre distribución [Tomcat] en su versión 5.5.

- **Instalación del sistema gestor de bases de datos.** Para manejar la persistencia de datos instalamos, en la máquina servidora de bases de datos, un sistema gestor de bases de datos relacionales. Existen en la actualidad diferentes tipos de sistemas de pago y de libre distribución. Dado que, en teoría todos estos ellos se adscriben a un mismo estándar no debería haber problemas por escoger cualquiera de ellos, No obstante nosotros hemos optado por [MySQL] en su versión 5.0.

- **Despliegue de la aplicación Web Pelican en el contenedor de aplicaciones.** Siguiendo las instrucciones del fabricante del contenedor de aplicaciones despliegue la aplicación Pelican sobre éste. Esta aplicación es un fichero de extensión *.war* que puede encontrarse en el CD que se adjunta con esta tesis.

- **Instalación del esquema de bases de datos de Pelican.** En el CD adjunto también se proporciona un fichero llamado *pelican_db.sql* que contiene un script de instalación del esquema de la base de datos necesaria para ejecutar Pelican. Siga las instrucciones del sistema gestor de bases de datos que utilice para ejecutar este script.

- **Configuración de Pelican.** Edite los ficheros de configuración de Pelican para ajustar los parámetros necesarios que le permitirán poner en marcha la plataforma de aprendizaje. En la siguiente sección se hace una descripción detallada de todo este trabajo.

- **Arranque del contenedor de aplicaciones.** Siguiendo las instrucciones de su fabricante arranque el contenedor de aplicaciones Web y abra un navegador Web. Sobre la barra de navegación escriba la ruta que se muestra a continuación (donde nombre–servidor hace referencia al nombre DNS o dirección IP del servidor Web donde está instalado el contenedor). Si aparece una pantalla requiriendo un nombre de usuario y una contraseña, el proceso de instalación ha concluido con éxito. Por defecto el administrador de la plataforma Pelican tiene por nombre *manager* y clave *manager*, a no ser que se haya modificado esto en el paso de configuración previo.

```
http://nombe-servidor:8080/pelican
```

B.3. Configuración de la plataforma Pelican

La configuración de la plataforma de aprendizaje Pelican se realiza a través de la edición de cuatro ficheros diferentes. El primero de ellos está relacionado con la base de datos y los otros tres restantes son propios de la plataforma. A lo largo de los siguientes apartados de esta sección describiremos el propósito de cada uno de ellos.

Fichero de configuración hibernate.properties

El fichero de configuración *hibernate.properties* es un recurso que se encuentra en la ruta Web que se muestra a continuación (relativa al contexto donde se instala la aplicación) y que contiene una colección de pares atributo–valor que sirven para configurar determinados parámetros de conexión con la base de datos relacional.

```
WEB-INF/hibernate/hibernate.properties
```

Las principales entradas de este fichero se muestran en el listado B.1. En primer parámetro hace referencia al dialecto sql que se utilizará para comunicarse con la base de datos. El segundo, debe indicar la clase que implementa el controlador para conectarse al sistema gestor. Si se utiliza un sistema distinto al proporcionado por defec-

to, deberá copiar en el directorio *WEB-INF/lib* el jar del controlador para su sistema gestor. El tercer parámetro hace referencia a la URL que sirve de cadena de conexión para la base de datos. Y finalmente, los parámetros *username y password* son utilizados para autenticarse en la base de datos. Los valores que tiene puesto este fichero corresponden, como dijimos al sistema de MySQL.

Listado B.1. Extracto del fichero de configuración hibernate.properties.

```
hibernate.dialect                = ...
hibernate.connection.driver_class = ...
hibernate.connection.url         = ...
hibernate.connection.username    = ...
hibernate.connection.password    = ...
...
```

Fichero de configuración society-config.xml

El fichero de configuración *society-config.xml* sirve para especificar todos los aspectos configurables de la plataforma relacionados con el subsistema social. En el listado B.2 puede verse un extracto de su contenido. Como puede apreciarse, el documento comienza con una especificación de los parámetros por defecto de los actores que indica cuál es el nombre, descripción y permisos que se debe asignar por defecto a un actor recién creado (❶). Después se incluye una descripción de los actores definidos por defecto. En la distribución que se proporciona en el CD existen dos actores que se crean al arrancar por primera vez Pelican: el *manager*, con todos los permisos concedidos y el *guest* con permisos de acceso únicamente a su espacio de trabajo privado (❷). A continuación se definen los valores por defecto para los usuarios recién creados, indicando nombre, descripción y datos de perfil por defecto (❸). Luego se definen los usuarios que se crearán por defecto al arrancar por primera vez Pelican. Para ellos se indica su perfil de usuario y el actor que encarnan. En la distribución proporcionada se crean dos usuarios *manager* y *guest* que encarnan los actores de su mismo nombre (❹). Finalmente, se define una configuración similar de valores por defecto para las plantilla de grupo (❺) y los grupos (❻).

Listado B.2. Extracto del fichero de configuración society-config.xml.

```
<society-config>
  <actors>
    <actor-settings> ❶
      <default-name>New actor</default-name>
        <default-description>New actor</default-description>
        <default-permissions>
          <permission>projects.view</permission> ...
        </default-permissions>
    </actor-settings>
    <default-actors> ❷
      <actor name="manager">
        <description> ... </description>
```

```
        <permissions>
          <permission> ... </permission>  ...
        </permissions>
      </actor>
      <actor name="guest"> ... </actor>  ...
    </default-actors>
  </actors>
  <users>
    <user-settings apply="false"> ❸
        <default-name>...</default-name>
        <default-address>...</default-address>
        <default-zip-code>...</default-zip-code>
        <default-country>...</default-country>
        <default-city>...</default-city>
        <default-e-mail>...</default-e-mail>
        <default-phone>...</default-phone>
    </user-settings>
    <default-users> ❹
      <user login="manager" password="manager">
        <profile>  ...  </profile>
        <actor>manager</actor>
      </user>
      <user login="guest" password="guest">
        <profile>  ...  </profile>
        <actor>guest</actor>
      </user> ...
    </default-users>
  </users>
  <group-templates> ❺
    <group-templates-settings>
       <default-name>New group template</default-name>
       <default-description>New group template</default-description>
    </group-templates-settings>
    <root-group-template name="societyTemplate">
      <description>... </description>
    </root-group-template>
  </group-templates>
  <groups> ❻
    <groups-settings>
      <default-name>New group</default-name>
      <default-description>New group</default-description>
    </groups-settings>
    <root-group name="society">
      <description>...</description>
    </root-group>
  </groups>
</society-config>
```

Fichero de configuración collaboration-config.xml

El fichero de configuración *collaboratión-config.xml* se encarga de establecer todos los valores configurables relacionados con el subsistema de colaboración y de integración. Como puede verse en el extracto de la figura B.3, este documento define el nombre y descripción por defecto de las plantillas de actividad (❶) y proyecto (❷) creadas en Pelican. Sin embargo, la sección más importante de este fichero es aquella que permite definir las herramientas que se integran en la plataforma para ofrecer sus servicios. A cada servicio (tool) debe asignársele un id único y debe proporcionarse la url donde ésta está instalada asi como otros parámetros como su nombre, descripción o si es proveedora de servicios Web (❸). En la sección de parámetros se especifican los parámetros que conforman el esquema de invocación, indicando para cada uno de ellos su nombre, etiqueta (nombre en la interfaz de usuario) y valor por defecto (que puede ser una expresión P#) (❹). Finalmente, en caso de que el servicio proporcione eventos externos éstos deben declarase indicando su nombre y tipo (❺).

Listado B.3. Extracto del fichero de configuración society-config.xml.

```
<collaboration-config>
  <activities>
    <activity-settings> ❶
      <default-name>New activity</default-name>
      <default-description>New activity</default-description>
    </activity-settings>
  </activities>
  <projects>
    <project-settings> ❷
      <default-name>New project</default-name>
      <default-description>New project</default-description>
    </project-settings>
  </projects>
  <tools>
    <tool id="chat.enlace.ltcs.lsi.uned.es"  ❸
          url="http://localhost:8080/chat/login"
          webService="false">
      <name> Chat </name>
      <description> ... </description>
      <icon> /images/service.gif </icon>
      <parameters>
        <parameter name="..." label="..." value="..."/> ... ❹
      </parameters>
      <event-types>
        <event-type name="..." type="..."/> ... ❺
      </event-types>
    </tool> ...
  </tools>
</collaboration-config>
```

Fichero de configuración policy-config.xml

El fichero de configuraciónsestablece la configuración relacionada con el subsistema de intervención de Pelican. Como puede verse en el extracto de la figura B.4, en él se describen todos los tipos de eventos que puede disparar la plataforma Pelican (recuérdese que los eventos externos fueron especificados en el documento anterior). Si nos fijamos detenidamente cada entrada *event-type* contiene un nombre, un tipo, una etiqueta, una descripción y un atributo *fire* (❶). Éste último sirve para indicar a la plataforma si se desea que se lancen eventos de ese tipo. Su valor por omisión es *true*. Dentro de la estructura context se puede definir de forma declarativa la colección de objetos de dominio que se desea que se incluyan en el evento como variables implícitas (❷). Para cada una de ellas se indica su nombre y su tipo, la referencia (id) al objeto que debe ser capturado, el ámbito del que se debe extraer la información y cuándo debe extraerse (antes o después de procesarse la acción que causó el evento). Toda esta es una configuración muy potente que evita tener que cablear la construcción de las variables implícitas de los eventos. No obstante su semántica es complicada de entender sin conocer en detalle el funcionamiento interno de Pelican y la modificación de estos valores puede suponer que se produzcan errores en la construcción de las variables. Por ello recomendamos que esta zona de la especificación no sea alterada al menos sin la supervisión de un experto.

Listado B.4. Extracto del fichero de configuración society-config.xml.

```
<policy-config>
  <event-types>
    <event-type name="update-state-activity"
                type="pelican.workspace.activity.updateState"
                fire="true"> ❶
      <label> Updte activity state </label>
      <description> ... </description>
      <context> ❷
        <variable
           name="activity.old"
           type="pelican.policy.context.bean.BeanPolicyVariable"
           innerType="pelican.collaboration.workspace.Activity"
           id="collaboration.workspace.activity.id"
           scope="request"
           when="before"/> ...
       </context>
     </event-type>
  </event-types>
</policy-config>
```