Modelado arquitectónico de la plataforma Pelican para la implantación de entornos colaborativos de aprendizaje[*]

Javier Vélez, Mª Felisa Verdejo

Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática., Universidad Complutense de Madrid
28040 Madrid
jvelezre@fdi.ucm.es

Departamento de Lenguajes y Sistemas Informáticos
ETS Ingeniería Informática, UNED
28040 Madrid
felisa@lsi.uned.es

Resumen: El aprendizaje colaborativo es una aproximación pedagógica que cada vez goza de mayor aceptación en nuestros días. Este paradigma promueve la puesta en práctica de sesiones de trabajo donde los estudiantes se organizan en grupos para llevar a cabo el desarrollo de actividades de aprendizaje. Pero cuando las actividades se realizan por mediación de la tecnología es preciso proporcionar herramientas interactivas para dar soporte a la colaboración. Estas herramientas deben integrarse dentro de un entorno de aprendizaje con el nivel adecuado de interoperabilidad para facilitar a los estudiantes el uso de las mismas. En este artículo presentamos Pelican, una plataforma para el aprendizaje que proporciona una arquitectura de integración ligera y flexible de herramientas externas y da soporte a los procesos de diseño social y colaborativo propios de escenarios de aprendizaje grupal.
Palabras clave: Colaboración, aprendizaje colaborativo, integración, diseño instruccional.

Abstract: Collaborative learning is a pedagogical approach with a broad acceptance nowadays. This paradigm fosters the development of working sessions where students are arranged into groups in order to undertake a collection of learning activities. But when learning activities are mediated by technology it is necessary to provide a set of interactive tools to support collaboration. These tools should be integrated within a learning environment with the proper level of interoperability to offer students an easy and straightforward use of them. In this paper we present Pelican, a learning platform providing a lightweight and flexible architecture for external tool integration. In addition it supports social and collaborative design processes taking place within group learning scenarios.
Key words: Collaboration, collaborative learning, integration, instructional design.

1. Introducción

El aprendizaje colaborativo es una aproximación pedagógica que promueve el desarrollo de sesiones de trabajo donde los estudiantes se organizan en grupos para realizar, conjuntamente, una serie de actividades [Sharan, 1994]. Este nuevo enfoque está cobrando gran aceptación en nuestros días por cuanto trae consigo una serie de ventajas. Por un lado, concede a los estudiantes un rol más participativo que

[*] Artículo seleccionado del VIII Simposio Nacional de Tecnologías de la Información y las Comunicaciones en la Educación – SINTICE 2007, (Zaragoza, España 2007), extendido y revisado para su publicación en **IE** Comunicaciones.

aquél que se les otorgaba en la enseñanza convencional ya que éstos deben encontrar mediante el consenso una solución a los problemas planteados [Dillenbourg & Baker, 1996]. A esto se suma el hecho de que los alumnos, al trabajar entre iguales, se muestran más dispuestos a colaborar en las sesiones de trabajo de lo que se mostrarían si lo hicieran en presencia de un adulto o un experto. Por otro lado, el desarrollo de las actividades constituye un marco idóneo donde es posible aprender en escenarios próximos a situaciones reales [Suchman, 1987]. Y además, la colaboración ofrece la posibilidad de desarrollar habilidades y destrezas sociales que cada vez están cobrando una mayor relevancia como objetivos pedagógicos [Johnson et al., 1998].

Pero cuando las experiencias de aprendizaje colaborativo se llevan a cabo haciendo uso de la tecnología es necesario proporcionar a los estudiantes suficientes medios de interacción que permitan una colaboración efectiva. En este sentido, la mayoría de las herramientas actuales que se inscriben dentro del campo del CSCL (Computer Supported Collaborative Learning) se centran en ofrecer una colección de mecanismos de interacción independientes entre sí tales como foros de discusión, sistemas de mensajería instantánea, repositorios de recursos [Moodle] [BlackBoard] [WebCT] o herramientas de modelado y simulación para dominios específicos [Collab] [MSpace].

Sin embargo, la experiencia demuestra que el desarrollo de actividades de aprendizaje colaborativo se lleva a cabo a través del uso transversal de una serie de servicios proporcionados por herramientas heterogéneas entre sí [Van Joolinger et al, 2007] [Bollen et al., 2007] [Vélez & Verdejo, 2007]. Por tanto resulta conveniente proporcionar un mecanismo mediante el cual se facilite la integración de las mismas dentro de un entorno común de manera que se garantice cierto nivel de interoperabilidad entre ellas. Esto proporcionará a los usuarios una sensación de continuidad en el uso de las mismas que se adapta a los requerimientos de los flujos de instrucción colaborativos.

Aunque, en la actualidad ya existen algunas soluciones en esta línea [Caeiro et al., 2006] [Bote-Lorenzo et al., 2004] [OGSi] [IMS – IT] lo cierto es que, como se discutirá más adelante, en la sección 4 de este artículo, la mayoría de estas propuestas imponen unas fuertes restricciones tecnológicas que se traducen en unos costes elevados para conseguir un adecuado nivel de integración. Para salvar esta limitación, y en este mismo sentido, se ha desarrollado la herramienta Pelican, una plataforma de aprendizaje colaborativo que da soporte a todos los procesos que intervienen en este tipo de experiencias y que permite integrar, con unos mínimos requerimientos, una gran variedad de herramientas externas desarrolladas por terceras partes con un carácter heterogéneo entre sí.

Este artículo presenta, desde una perspectiva tecnológica, la plataforma Pelican discutiendo cada uno de los artefactos necesarios que incorpora para dar soporte a las necesidades que surgen en el seno de las experiencias de aprendizaje colaborativo. Aquellos lectores con un perfil no técnico pueden omitir la descripción de los detalles de implementación más específicos y en particular de los diagramas de clases e interacción utilizados en la sección 2 para ilustrar la propuesta.

En concreto comenzaremos presentando, en la sección 2, la plataforma de integración Pelican, desde un punto de vista arquitectónico. En la sección 3 se ilustra el uso de la plataforma para articular un escenario de aprendizaje arquetípico: la división colaborativa del trabajo. La sección 4 discute diferentes propuestas de integración alternativas comparándolas con la solución aquí presentada. Y finalmente, en la sección 5 se exponen las conclusiones de este trabajo.

2. La plataforma Pelican

Como comentamos en la introducción, el desarrollo de experiencias de aprendizaje colaborativo mediadas por computador requiere de la existencia de un entorno informático que, por un lado, sea capaz de atender a los aspectos de diseño de escenarios de aprendizaje y, por otro, proporcione una familia de medios de soporte a la colaboración. Algunos autores ya han apuntado, en esta línea, la necesidad de independizar estos dos aspectos. Es decir, separar las herramientas de soporte a la colaboración del propio entorno donde se inscribe su uso [Dimitrakopoulou, 2005]. Pelican es una plataforma de integración que

responde precisamente a esta separación proporcionando un mecanismo para la integración de las mismas.

La plataforma de integración Pelican es un entorno para la colaboración que está formado por 4 sistemas interrelacionados. Éstos responden a cada una de las prestaciones que ofrece la plataforma:

- El sistema social. Este sistema permite la definición de comunidades virtuales de aprendizaje y la gestión y administración de los grupos y usuarios que las constituyen.
- El sistema de colaboración. El sistema de colaboración está basado en el aprendizaje dirigido por proyectos, según el cual los estudiantes se suscriben a un proyecto que está constituido por una colección de actividades para realizar conjuntamente con otros estudiantes.
- El sistema de intervención. Con el sistema de intervención, los administradores y diseñadores instruccionales pueden definir el comportamiento dinámico de la plataforma para que responda a las necesidades puntuales de los flujos de instrucción colaborativos.
- El sistema de integración. El sistema de integración proporciona un mecanismo para la integración de diferentes herramientas externas desarrolladas por terceras partes.

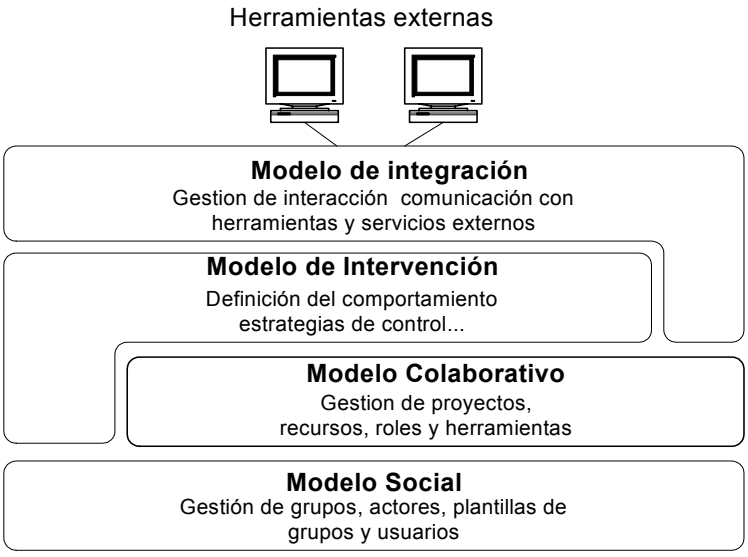

Figura 1. Arquitectura de Pelican

Como se puede apreciar en la figura 1, el sistema social constituye la base arquitectónica de la plataforma Pelican. Sobre él se ubica el sistema de colaboración que permite definir las experiencias de aprendizaje. El sistema de intervención actúa sobre estos dos sistemas para redefinir puntualmente la organización social y colaborativa del entorno para una cierta actividad y finalmente, el sistema de integración se apoya en los tres anteriores para articular los mecanismos oportunos de interoperabilidad entre las herramientas. A lo largo de los siguientes apartados entraremos a describir cada uno de estos sistemas en detalle.

2.1. El sistema social

El sistema social cubre todos los aspectos relacionados con la dimensión social del aprendizaje colaborativo [Vélez et al., 2006] [Vélez et al., 2005]. Gracias a él, se puede dar soporte en Pelican a comunidades virtuales de aprendizaje con una complejidad estructural potencialmente elevada. La hipótesis de partida de este sistema es que todo colectivo de usuarios constituye una sociedad caracterizada a partir de una colección de estructuras organizativas sociales dinámicamente cambiantes. Dentro de ellas puede haber diferentes tipos de miembros que tienen asociadas ciertas responsabilidades sociales. Para articular esta idea, el sistema social de Pelican ofrece funcionalidades para describir *modelos de sociedad* específicos atendiendo a unos requerimientos organizativos y estructurales. Después es posible crear sociedades reales de usuarios que se ajusten a dichos modelos. Esta separación en dos niveles – nivel de diseño y nivel de administración – redunda en una serie de ventajas importantes.

En primer lugar, se consigue independizar las tareas de modelado de un tipo de sociedad (nivel de diseño) de la gestión y mantenimiento de sociedades específicas que siguen dicho modelo (nivel de administración), lo que posibilita que diferentes equipos – diseñadores por un lado y administradores por otro – dediquen sus esfuerzos a tareas diferentes. En segundo lugar, estos dos tipos de tareas generan dos tipos de elementos software distintos aunque interdependientes: el *modelo de sociedad* y las *sociedades específicas* que se adscriben a él. Los

modelos constituyen un artefacto que permite prescribir y entender cuál es la estructura organizativa de una sociedad. Por su parte, cada sociedad mantiene información acerca de cómo cada colectivo de usuarios se organiza en torno a las restricciones estructurales impuestas por el modelo.

Como consecuencia de esta dualidad aparece una tercera ventaja. Por un lado, es posible realizar cambios dentro de una sociedad específica alterando su composición sin necesidad de modificar el modelo de sociedad y por tanto manteniendo intacto la organización estructural de otras sociedades que sigan el mismo modelo. Por otro, dado que cada modelo centraliza la descripción estructural de todas las sociedades vinculadas a él, un cambio en un modelo de sociedad afecta dinámicamente a todas sus sociedades lo cual facilita enormemente las labores administrativas. Por último, los modelos de sociedad descritos en Pelican se convierten en productos que pueden ser reutilizados en distintos contextos y escenarios pedagógicos.

La figura 2 ilustra el diagrama conceptual en UML que utiliza el sistema social. En él se conjugan las entidades del nivel de diseño que sirven para dar soporte a la descripción de los modelos de sociedad (Plantillas de grupo, Actores y Permisos) con aquellas otras que son necesarias para representar sociedades específicas (Grupos y Usuarios). Como puede apreciarse, existe un claro paralelismo entre estas dos familias de artefactos. Los grupos se ajustan a una plantilla de grupo y los usuarios encarnan uno o varios actores en diferentes grupos. A continuación describimos en detalle cada uno de los elementos en el modelo:

- Actor. Los actores permiten representar los diferentes arquetipos de usuarios que intervienen en una comunidad virtual (profesores, estudiantes, etc.). Este artefacto sirve para ofrecer a los usuarios una ubicación precisa dentro de la misma y conferirles una serie de responsabilidades y competencias sociales.
- Permission. Para dar soporte a la idea de responsabilidad social se permite definir los actores en términos de una colección de permisos de acceso a las diferentes funcionalidades de la plataforma.

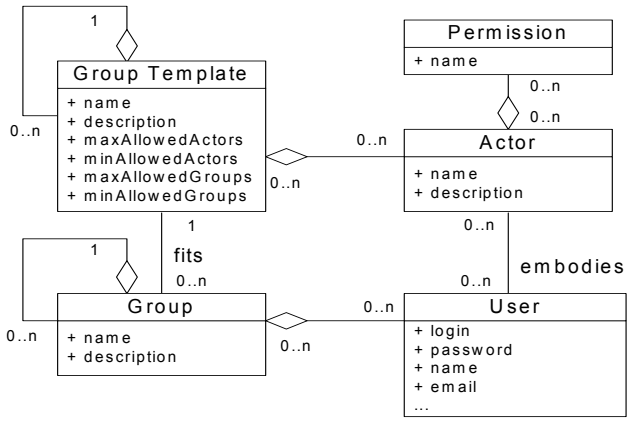

Figura 2. Modelo social de Pelican

- Group Template. Las plantillas de grupo indican a la plataforma qué tipos de grupos existen en un modelo de sociedad y cuál es su estructuración interna. Ésta es una especificación jerárquica que describe las plantillas de grupo hijas que pueden aparecer en su interior así como la colección de actores permitidos junto con sus cardinalidades mínima y máxima.
- Group. Los grupos representan a cada uno de los colectivos de usuarios que aparecen en una sociedad particular. Estas entidades sirven para indicar las diferentes agregaciones de usuarios y cómo éstos encarnan a diferentes actores dentro de cada grupo. Además, cada grupo adjunta información del conjunto de grupos hijos que contiene.
- User. Cada usuario de una sociedad en Pelican queda representado por un ejemplar de la entidad User. Este artefacto constituye un modelo de usuario que almacena tanto información del usuario como aquella relativa al estado actual del mismo.

2.2. El sistema de colaboración

El sistema de colaboración permite describir experiencias de aprendizaje con un grado suficiente de abstracción y generalidad como para garantizar su reutilización en diferentes contextos sociales y educativos. Para ello, proporciona un marco conceptual que permite organizar tal descripción en términos de proyectos y actividades de aprendizaje que luego pueden ser desplegados dentro de una colección de espacios de trabajo compartidos que se

alinean con la estructura jerárquica de grupos soportada por el sistema social. De esta forma, un espacio de trabajo puede entenderse como un entorno virtual donde tienen lugar las actividades colaborativas: un conjunto de proyectos de aprendizaje y, por otro, un conjunto de herramientas específicas de soporte a la colaboración.

Al igual que antes, es posible distinguir aquí dos niveles de uso de los artefactos del sistema de colaboración. En el nivel de diseño los diseñadores instruccionales definen proyectos de aprendizaje colaborativo indicando las actividades que deben realizar los estudiantes y proporcionando el material y el conjunto de herramientas externas necesarias para llevarlas a cabo. Estos proyectos constituyen estereotipos reutilizables que describen escenarios y/o experiencias de aprendizaje que podrán ser llevados a cabo por diferentes grupos de estudiantes. En el nivel de administración, los administradores despliegan estos proyectos sobre uno o varios espacios de trabajo compartidos para que los desarrollen los estudiantes.

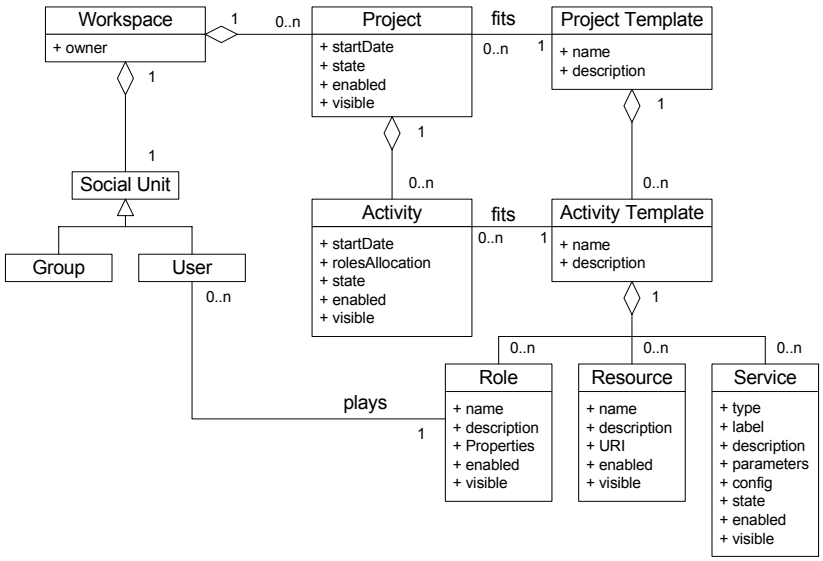

Figura 3. Modelo de colaboración de Pelican

Como puede apreciarse en la figura 3, en el sistema de colaboración coexisten dos tipos de elementos: aquellos que sirven para dar soporte a la descripción abstracta de proyectos y actividades, tales como Project Template, Activity Template, Resource, Role y Service, y aquellas otras entidades que soportan el estado de desarrollo de las experiencias colaborativas en diferentes grupos, entre las que destacan Project, Activity y Workspace. A continuación discutimos en detalle el propósito de cada una de estos artefactos dentro del sistema de colaboración:

- Project Template. Las plantillas de proyecto constituyen estereotipos de proyectos. Cada uno consiste en una colección de plantillas de actividad e incluyen un nombre y una descripción que detalla los propósitos generales del mismo.
- Activity Template. Una plantilla de actividad describe una actividad de aprendizaje. Esta especificación consiste en un nombre y una descripción de los objetivos de la misma. Además incluye la colección de roles, recursos y servicios que son necesarios para llevarla a cabo.
- Resource. Los recursos representan una referencia Web que se incluye como parte de la descripción de una plantilla de actividad. Estas entidades tienen por objeto ofrecer a los estudiantes ayuda o material de apoyo para el desarrollo de la misma.
- Service. Un servicio es una configuración específica de una herramienta externa para que ofrezca una funcionalidad adaptada al contexto social y colaborativo en el cual se inscribe su uso. Esta entidad consta básicamente, de una dirección URL que apunta a la herramienta que proporciona el servicio y de un *esquema de invocación*. Este último es una colección de pares parámetro – valor que indica una invocación correcta de la herramienta.
- Role. Las actividades de aprendizaje colaborativo se desarrollan, comúnmente, a partir de un proceso de división del trabajo que asocia una serie de responsabilidades a cada miembro del grupo. Los roles se exponen en Pelican como una manera de dar soporte a esta asignación de responsabilidades dentro de una actividad (moderador, cronometrador, anotador, etc.).
- Project. Cuando los administradores de la plataforma despliegan una plantilla de proyecto sobre el espacio de trabajo compartido de un grupo, provocan la creación de un proyecto. La

función del proyecto es mantener una referencia a la plantilla de proyecto correspondiente dentro del espacio de trabajo e indicar cuál es su estado actual de desarrollo.

- Activity. Las actividades describen el estado de desarrollo de cada una de las plantillas de actividad incluidas en la plantilla de proyecto. En concreto, una actividad consta de una referencia a la plantilla de actividad correspondiente y contiene información acerca de la asignación de los roles a los miembros del grupo.
- Workspace. El espacio de trabajo es una entidad que representa el entorno virtual dentro del cual se inscribe la actividad de los estudiantes cuando realizan proyectos de aprendizaje y ofrece acceso a aquéllos desplegados sobre él. Con estos elementos el espacio de trabajo permite utilizar las herramientas externas de soporte integradas en él y navegar por cada uno de los proyectos desplegados así como por sus actividades internas.

2.3. El sistema de intervención

Los dos sistemas que hemos presentado en las secciones anteriores permiten definir la organización estructural del entramado social y colaborativo sobre el que se soportan los escenarios de aprendizaje. En muchos casos realizar una definición estática e inmutable de esta organización puede ser suficiente. No obstante muchas otras experiencias pedagógicas requieren que la organización social y colaborativa inicial de un escenario se modifique dinámicamente para adaptarse a ciertas necesidades que surjan sobre la marcha.

El sistema de intervención proporciona, en este sentido, un mecanismo que permite a los diseñadores instruccionales definir, mediante las interfaces Web de Pelican, *scripts adaptativos* haciendo uso de un lenguaje de dominio propio llamado P#. Un script adaptativo es la especificación formal de una colección de transformaciones elementales aplicada sobre las entidades de los sistemas social y colaborativo[1]. Cuando un actor con permisos en el sistema – típicamente un profesor o un monitor – lo cree conveniente, puede lanzar un script a ejecución para que los cambios surtan efecto.

Los scripts adaptativos se convierten así en un elemento inherente a la definición de muchos escenarios colaborativos en tanto que se encargan de capturar todos los aspectos dinámicos de estos. En efecto, gran parte de la descripción de los aspectos dinámicos de un escenario puede expresarse por medio de la planificación de una cadena de transformaciones adaptativas que deberán ser aplicadas puntualmente dentro del flujo de instrucción colaborativo. Por ejemplo iterar el desarrollo de una actividad una serie de veces y alterar la asignación de roles en cada iteración para garantizar que cada estudiante desempeñe al menos una vez cada rol dentro de la misma.

Para permitir la planificación temporal de la aplicación de las transformaciones el sistema de intervención proporciona un mecanismo basado en reglas y dirigido por eventos. Cada acontecimiento relevante ocurrido dentro del flujo instruccional orquestado por Pelican provoca el disparo de un evento. Los diseñadores instruccionales pueden definir reglas que vinculen el disparo de dichos eventos a scripts adaptativos previamente definidos. Este procedimiento garantiza un seguimiento adaptativo de la evolución de flujo de instrucción a lo largo del tiempo. En las siguientes líneas discutiremos los artefactos que intervienen en el sistema de intervención, cuyo modelo en UML se presenta en la figura 4, y describiremos la colaboración que se produce entre dichas entidades.

- Policy. La política es el elemento central del modelo de intervención. Una política consiste en una encapsulación etiquetada de una colección de reglas que permiten definir determinado comportamiento de la plataforma. En efecto, los escenarios relativamente complejos que requieren aplicar diferentes transformaciones a lo largo del tiempo para adaptar el entorno a sus necesidades se expresan, comúnmente, a partir de una colección de reglas que se encuentran conceptualmente vinculadas entre sí en tanto que todas ellas capturan en su conjunto la

[1] *No debe confundirse este tipo de scripts adaptativos con los scripts colaborativos propios del CSCL. Los primeros describen, desde una perspectiva tecnológica, una forma de modificar la configuración social y colaborativa de la plataforma. Los segundos tienen por objeto describir, desde una perspectiva conceptual, una forma de proceder para desarrollar un conjunto de actividades colaborativas.*

especificación de los aspectos dinámicos de un mismo escenario.

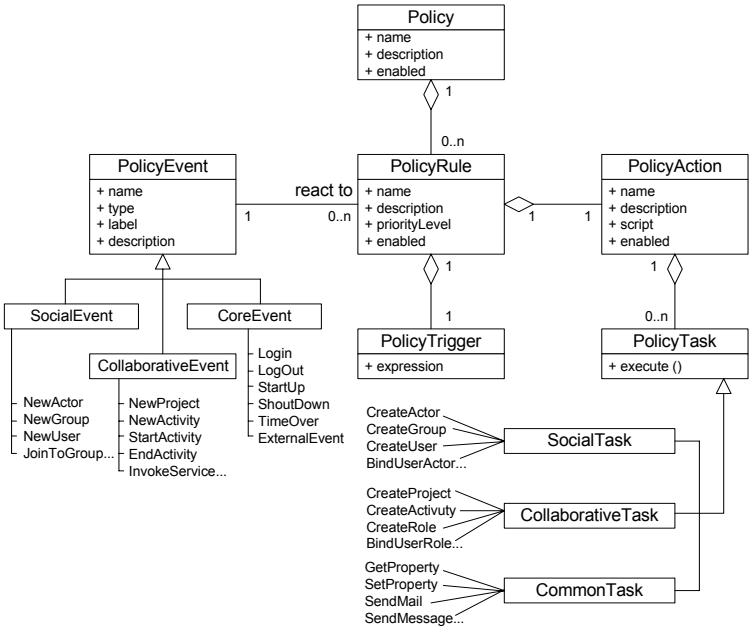

Figura 4. Modelo de intervención de Pelican

- PolicyRule. Una regla permite planificar la ejecución de un script adaptativo en un determinado momento dentro del flujo de instrucción colaborativo asociado al escenario. Dicho momento se caracteriza asociando a la regla un evento y un disparador que indica las condiciones ambientales que deben alcanzarse en el entorno para que se dispare la regla.
- PolicyEvent. Los eventos caracterizan cada una de las situaciones bajo las cuales es posible evaluar una regla para su potencial ejecución. Estos eventos responden a una taxonomía que se corresponde con cada una de las acciones que los usuarios pueden hacer sobre la plataforma (entrar en un grupo, salir de él, entrar en una actividad, etc.) o en alguna de las herramientas externas (por ejemplo, en una herramienta de votación, presentar candidato, votar, finalizar votación, etc.).
- PolicyTrigger. El disparador de una regla es una expresión lógica codificada en P# que expresa las condiciones bajo las cuales una regla debe lanzarse a ejecución.
- PolicyAction. La acción política consiste en el script P# que debe lanzarse a ejecución cada vez que se dispara un evento del tipo indicado en la regla y se verifican las condiciones del disparador. Este script combina sentencias de control de flujo iterativo y condicional (foreach e if-else) con invocaciones a operaciones atómicas de transformación llamadas tareas que deben aplicarse para adaptar la plataforma a las necesidades puntuales del escenario.
- PolicyTask. Las tareas constituyen operaciones elementales de transformación que pueden ser invocadas desde la sintaxis de los scripts de P#. Como puede apreciarse en la figura 4, existe una amplia taxonomía de tareas programadas que se encuentran disponibles a los administradores (crear proyecto, asignar usuario a role, etc.).

2.3.1. Ejecución de reglas políticas

Las construcciones sintácticas codificadas en P# (tanto los disparadores como los scripts asociados a las reglas) se expresan en términos de una colección de variables que hacen referencia a diferentes objetos de datos de alguno de los modelos de información anteriores. Por ejemplo *$currentUser* es una variable reservada que representa al objeto User asociado al usuario en curso (aquél cuya acción provocó el disparo del evento en la plataforma). Este mecanismo es el que permite que las reglas de intervención se expresen con un nivel de abstracción adecuado. No obstante, ello exige que previamente a interpretar cualquier construcción en P# sea necesario contextualizarla.

El proceso de contextualización consiste en traducir la construcción haciendo uso de un contexto asociado al usuario en curso que contiene un valor para cada una de las variables allí referenciadas. Este contexto es mantenido y actualizado por la plataforma para cada usuario registrado. En concreto, mediante la contextualización, las referencias a variables se sustituyen por su valor, las sentencias de control de flujo condicional se resuelven en favor del bloque de sentencias asociado al ramal que debe ejecutarse una vez evaluada la expresión condicional y las sentencias de control de flujo iterativo se resuelven desarrollando el bloque iterado para generar la secuencia ordenada de tareas equivalente.

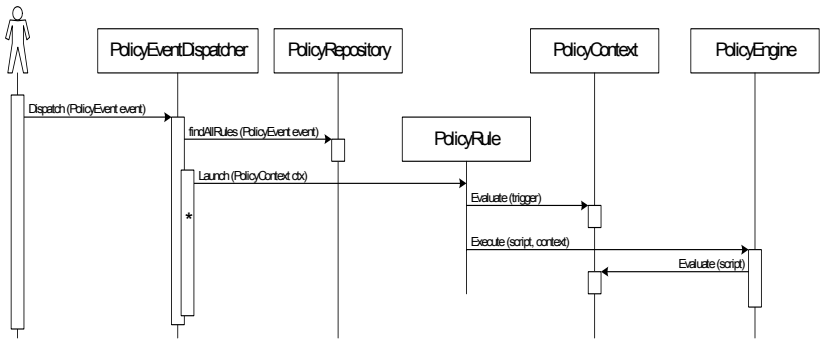

Figura 5. Ejecución de una política en Pelican

En la figura 5 se ilustra la colaboración que se produce entre las entidades principales del modelo de de intervención para llevar a cabo la ejecución planificada de una colección de scripts en respuesta a un evento del sistema. Como puede apreciarse, el proceso arranca con el lanzamiento de un evento por parte de la plataforma. Este evento es atendido por la entidad PolicyEventDispatcher quien se encarga, en primer lugar, de consultar el repositorio de reglas políticas para seleccionar todas aquéllas que fueron asociadas a un evento de este tipo. A continuación, evalúa los disparadores de cada una de ellas y envía a la entidad PolicyEngine los scripts que deban lanzarse a ejecución.

2.4. El sistema de integración

El sistema de integración permite incorporar una colección de herramientas Web externas, potencialmente heterogéneas y desarrolladas por terceras partes dentro de los espacios de trabajo compartidos de la plataforma. El objetivo es conseguir que estas herramientas alcancen un nivel de interoperabilidad entre ellas suficiente como para ofrecer al usuario una sensación de continuidad en el uso de las mismas. A este respecto, la responsabilidad de la plataforma es orquestar su funcionamiento para cohesionar los servicios proporcionados por ellas. Los principios en los que se apoya el sistema de integración para ello se resumen en los tres siguientes puntos:

- Integración orientada a la interacción. En el modelo de integración los procesos de interacción colaborativa que se producen en el seno del desarrollo de las actividades de aprendizaje son los que disparan los mecanismos de interoperabilidad entre las herramientas. En efecto, se pretende que la interacción que realice un usuario en una herramienta provoque cambios adaptativos potenciales en las otras herramientas y / o en los modelos de la plataforma.
- Integración basada en roles. El rol representa en Pelican un conjunto de responsabilidades colaborativas de cara a una determinada actividad. Para dar soporte a esta idea, las herramientas identifican el rol que está desempeñando un usuario y muestran un comportamiento, aspecto y funcionamiento adaptado al mismo.
- Integración dirigida por reglas y servicios Web. Para dar soporte a la integración e interoperabilidad entre las herramientas se utilizan, por un lado, reglas políticas para planificar la ejecución de acciones de orquestación puntuales que es necesario realizar a lo largo del flujo de instrucción. Por otro, los servicios Web permiten a las herramientas externas descubrir información de estado mantenida por los modelos social y de colaboración de la plataforma.

Recordemos, no obstante, que uno de los objetivos primordiales que distinguen este sistema de otras propuestas de integración que serán discutidas más adelante es el de alcanzar una buena integración de las herramientas sin forzar el uso de una solución tecnológica especifica. Por ello, Pelican organiza tal integración en cuatro niveles de interoperabilidad a los que las herramientas pueden adscribirse. Cada uno de ellos resulta progresivamente más acoplado a los modelos de Pelican pero también ofrecen mayores prestaciones:

- Nivel A. Integración ligera. Éste constituye el mecanismo más débil de integración. Para integrar una herramienta en Pelican a este nivel únicamente es necesario informar a la plataforma de cuál es la dirección Web donde se encuentra ubicada la herramienta y de cuáles son los parámetros de entrada que deben ser proporcionados a la misma para su funcionamiento, es decir su esquema de invocación. Para obtener un mayor nivel de

abstracción en la descripción del esquema de invocación, el valor de cada parámetro puede ser expresado en P# y contextualizado dinámicamente en tiempo de invocación.

- Nivel B. Integración unidireccional. Este nivel de integración permite que las herramientas de terceras partes consulten los modelos de información de Pelican a través de una colección de servicios Web. De esta manera, la gestión social y colaborativa se centraliza en la plataforma mientras que las herramientas pueden centrarse en ofrecer sus servicios de colaboración y se puede adaptar su comportamiento al contexto de explotación específico. Así, por ejemplo, las herramientas externas pueden consultar los permisos concedidos a un usuario o el rol que éste desempeña dentro de una actividad de Pelican para ofrecerle una interfaz adaptada.
- Nivel C. Integración bidireccional. En el nivel de integración C las herramientas de terceras partes ofrecen una serie de funcionalidades a otras herramientas cliente que quedan expuestas mediante una colección de servicios Web. Se plantean así escenarios de integración basados en el uso de modelos de interacción bidireccional entre las herramientas. En el caso particular de integración que nos ocupa, tales escenarios lo forman, de una parte, la plataforma Pelican, que expone los servicios Web tal y como se discutió en el nivel anterior, y, de otra, la herramienta externa en cuestión. El proceso de integración consiste, comúnmente, en llevar a cabo un protocolo de interacción consensuado entre Pelican y las herramientas.
- Nivel D. Interacción basada en componentes. Este nivel constituye el modelo más fuerte de integración propuesto. Para que una herramienta se integre con Pelican a este nivel, ésta debe implementar una colección de servicios Web que quedan prescritos por la plataforma. Estos servicios permiten controlar el ciclo de vida de las herramientas.

El sistema de integración, cuyo modelo UMS está representado en la figura 6, ilustra varios aspectos clave. En primer lugar que los 4 niveles anteriores están contemplados a través de diferentes tipos de servicios dentro del modelo. En segundo lugar, que los niveles B, C y D permiten a las herramientas mostrar un comportamiento adaptativo en función de los roles que juegan los usuarios de éstas. Esto es posible gracias a que éstas pueden interpretar la colección de propiedades de cada role. Es decir, las propiedades son, en este caso, un mecanismo de soporte en la plataforma mediante el cual se expresan directrices de configuración para las herramientas. Además, en los niveles C y D, Pelican puede hacer uso de funcionalidades proporcionadas por las herramientas como servicios Web, generalmente invocados mediante reglas políticas.

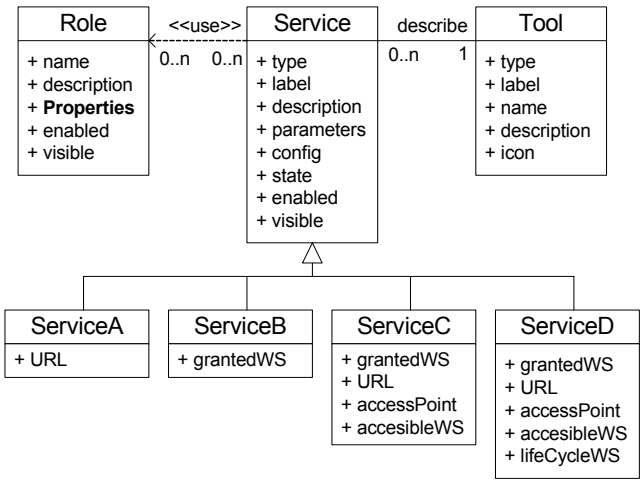

Figura 6. Modelo de integración en Pelican

3. Implantación de un escenario en Pelican

Una técnica de aprendizaje colaborativo que suele ser utilizada comúnmente en multitud de escenarios es la división colaborativa del trabajo. Mediante esta técnica un grupo de alumnos estudia cómo se puede dividir la actividad en tareas más elementales. Una vez hecha esta descomposición se crean equipos de trabajo formados por subconjuntos de los miembros del grupo y se les asigna a cada uno una de las tareas para que la desarrollen. Cuando se han desarrollado todas las tareas, los miembros del grupo vuelven a reunirse para integrar los resultados y generar un producto final. Con el fin de ilustrar las capacidades de la plataforma, a lo largo de los siguientes apartados describiremos en detalle cómo puede ser implantado este escenario en Pelican

3.1. Diseño social

Desde una perspectiva social, el problema se centra en realizar la asignación de tareas. Tal asignación se articula definiendo un nuevo grupo para cada tarea definida donde se inscriben los usuarios involucrados. Naturalmente, cada uno de estos grupos no tiene por qué ser disjuntos entre sí, ya que un usuario participará, potencialmente, en varias tareas. El reto tecnológico que supone la implantación de esta técnica es que la decisión acerca de qué tareas y, por ende, qué grupos se formarán para dar soporte a la descomposición de una actividad es una información no conocida a priori ya que corresponde al resultado de un proceso de negociación llevado a cabo por todos los miembros del grupo. Por tanto, la formación de los nuevos grupos es una operación que deberá realizarse dinámicamente solamente cuando se conozca el esquema de descomposición seleccionado. Tal y como discutiremos más adelante para ello utilizaremos las capacidades del sistema de intervención. No obstante si es posible definir a priori los dos tipos de plantillas de grupo que aparecen en este escenario: grupos y equipos de trabajo. Esencialmente un grupo estará formado por 6 estudiantes, un profesor y una colección de equipos de trabajo. Por su parte, los equipos de trabajo estarán formados por 3 estudiantes (figura 7).

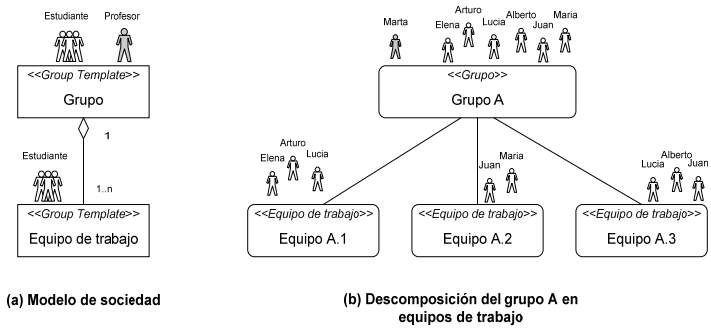

Figura 7. Organización social del escenario

3.2. Diseño de actividades

Describamos ahora cómo debe ser el diseño de las actividades que es necesario llevar a cabo para implantar la técnica de división del trabajo colaborativo en Pelican. De acuerdo al modelo de colaboración de Pelican, la organización de actividades de aprendizaje se organiza en torno a proyectos que luego son desplegados sobre algunos espacios de trabajo compartidos asociados a los grupos mantenidos por el modelo social. Como puede apreciarse en la figura 8, podemos distinguir aquí dos tipos de proyectos. Por un lado, el proyecto de grupo, que involucra a todos los miembros del grupo y está formado por la actividad de preparación, división del trabajo y consolidación. Por otro, los proyectos de equipo, que representan cada una de las tareas fruto de la descomposición, cuyo desarrollo se intercala dentro del proyecto de grupo a nivel de equipo y que a su vez están formadas por una secuencia ordenada de actividades, a priori desconocidas. Esta configuración también será generada dinámicamente mediante una regla cuando se conozca el esquema de división del trabajo.

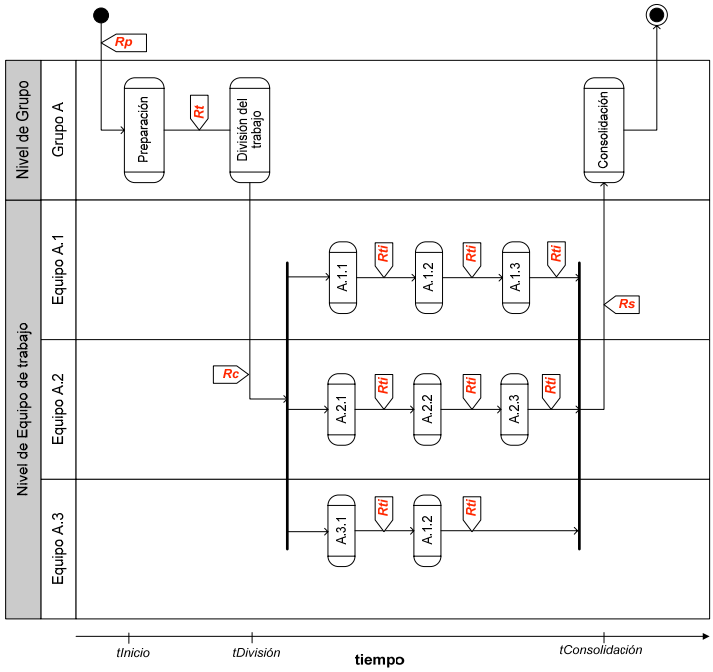

Figura 8. Desarrollo de la experiencia colaborativa a lo largo del tiempo

3.3. Integración de herramientas

En cuanto a la integración que requiere este escenario destacaremos dos herramientas que se requieren en la actividad de división del trabajo. Se podrían incluir otras herramientas, en esta u otras actividades pero su discusión carece de interés a este respecto. En concreto, para alcanzar un consenso acerca del plan de división del trabajo que se seguirá, los estudiantes

harán uso de una herramienta de discusión y votación que genera un informe, en formato XML, con la descripción precisa de la división acordada (número y composición de tareas, número y composición de equipos y asignación de los primeros a los segundos). Además, este resultado se almacena en una herramienta repositorio como un objeto de aprendizaje, convenientemente tipificado.

3.4. Aspectos dinámicos

Para capturar los aspectos dinámicos de este escenario definiremos una familia de reglas que han sido señaladas con etiquetas *Rx* en la figura 8 para indicar su momento de aplicación dentro del flujo instrucción. A continuación discutimos cada una de ellas y presentamos su codificación en P#.

- Rp. Regla de preparación. Esta regla tiene por objeto preparar la ejecución secuencial de las actividades del proyecto de grupo. Como puede verse en el evento asociado, la regla se ejecuta durante el despliegue del proyecto y habilita la actividad de preparación para que sólo ésta pueda ser accedida desde el espacio de trabajo. Esencialmente eso se hace recorriendo la lista de actividades del proyecto de grupo y para cada una se invoca a la tarea set para cambiar la propiedad *enabled* a false en todas las actividades menos en la primera (❶).

```
Event: << deployProjectEvent >>
Trigger: true
Action:
#set ($acts = ['Preparación', ...])
#set ($locks = ['true', 'false',
'false'])
#set ($crtAct = 0)
#foreach ($act in $acts)
  <set type = 'Activity' ❶
       name = '$act'
       property = 'enabled'
       value = '$locks.get($step)'/>
#end
```

Listado 1. Regla de preparación

- Rt / Rti. Regla de transición. Esta regla se encarga de mantener el secuenciamiento de actividades de los proyectos. Se dispara ante el evento de cambio de estado de una actividad cuando ésta ha finalizado (estado *Finished,* ❶). La acción localiza primero la actividad siguiente a aquella que está en curso (❷) e invoca la operación[2] *set* para activarla (❸). Además, durante el recorrido de búsqueda se invoca otra operación set para que se asegure de que el resto de las actividades se mantienen inaccesibles (❹).

```
Event: << changeActivityStateEvent >>
Trigger: $event.newState == 'Finished'
❶
Action:
#foreach ($act in $acts)
  #if ($step == $crtAct + 1) ❷
     <set type = 'Activity' ❸
          name = '$act'
          property = 'enabled'
          value = 'true'/>
      #set ($crtAct = $step)
  #else
    <set type = 'Activity' ❹
         name = '$act'
         property = 'enabled'
         value = 'false'/>
   #end
#end
```

Listado 2. Regla de transición

- Rc. Regla de creación. Esta regla es la encargada de acceder al repositorio de objetos, obtener el resultado de la votación, interpretar su contenido XML y, a raíz de ello, crear los equipos de trabajo, los proyectos correspondientes a cada tarea y desplegar estos últimos sobre el espacio de trabajo de los primeros. Como en el caso anterior la regla se dispara cuando la actividad de creación alcanza su estado de finalización (❶). En primer lugar se invoca *call-service* para llamar al servicio Web del repositorio que permite extraer el objeto que contiene el resultado de la votación (❷). Después se interpreta dicho XML para poder procesarlo desde P# (❸). De este modo es posible recorrer todos los grupos del documento y crear nuevos grupos de acuerdo a la plantilla de equipo invocando la operación *create-group* (❹), así como recorrer todas las tareas para crear los proyectos invocando las operaciones *create-proyect-template* y *create-activity-template* (❺) y desplegar dichos

[2] *Utilizaremos el término operación para referirnos a las tareas políticas con el ánimo de no confundirlas con las tareas colaborativas resultantes del proceso división del trabajo*

proyectos en los espacios de trabajo apropiados (❻).

```
Event: << changeActivityStateEvent >> ❶
Trigger: $event.newState == 'Finish' &&
         $event.activity ==
'preparacíon'
Action:
<call-service ❷
   url = '...'
   operation = 'getVotingSession'
   property = 'xmlResult'/>
<read-xml-stream ❸
   inputProperty = 'xmlResult'
   outputProperty = 'result'/>
#set ($groups = $result. groups) ❹
#foreach ($group in $groups)
  <create-group template = 'Equipo'/>
  #foreach ($user in $group.users)
    <bind-user-actor
       user = '$user'
       actor = 'Estudiante'/>
  #end
#end
#set ($tasks = $result.tasks) ❺
#foreach ($task in $tasks)
  <create-project-template
     name = '$task.name'
     description = '$task.name'/>
  #foreach ($activity in
$task.activities)
    <create-activity-template
      name = '$activity.name'
      description = '$activity.name'/>
  #end
#end
#foreach ($task in $tasks) ❻
  <create-project
     name = '$task.name'
     template = '$task.name'
     group = '$task.groups.get
($step)'/>
  #foreach ($activity in
$task.activities)
     <add-activity
       name = '$activity.name'
       project = '$task.name'/>
  #end
#end
```

Listado 3. Regla de creación

- Rs. Regla de sincronización. Finalmente, la regla de sincronización se encarga de efectuar la transición a la actividad de consolidación del proyecto de grupo solamente cuando todos los proyectos de equipo han finalizado. Esta regla se dispara cada vez que termina un proyecto (❶) y se encarga de contabilizar el número de proyectos terminados. Cuando ese número alcanza el de aquellos que fueron creados en la regla de creación (❷) significa que todos los proyectos de equipo han finalizado con lo que invoca una operación *set* encargada de activar la actividad de finalización del proyecto de grupo.

```
Event: << changeActivityStateEvent >>
Trigger: $event.project == 'Finished' ❶
Action:
#set ($nProjects = $nProjects + 1)
#if ($nProjects == $tasks.size ()) ❷
  <set type = 'Activity'
       name = '$acts.get ($crtAct)'
       property = 'enabled'
       value = 'true'/>
#end
```

Listado 4. Regla de sincronización

4. Otras propuestas relacionadas

Existen, dentro de la literatura del CSCL, algunas otras propuestas de diseño instruccional e integración que merecen la pena ser reseñadas aquí para compararlas con la que nosotros hacemos en este artículo. A continuación destacamos las más relevantes.

4.1. IMS – LD

La especificación [IMS – LD] constituye un estándar para realizar diseños instruccionales independientes de paradigma que pueden ser utilizados para definir escenarios colaborativos tal y como se describe en [Caeiro et al., 2003] [Caeiro et al., 2006]. Sin embargo, el problema surge a la hora de tratar de definir script de colaboración para que sean interpretados por los entornos de aprendizaje. [Miao et al., 2005] destacan 5 problemas a este respecto. En primer lugar, IMS no representa explícitamente el concepto de grupo ni de role. En Pelican estos dos elementos se incluyen para proporcionar funcionalidades de modelado social. En segundo lugar, en IMS los artefactos no pueden ser compartidos por un colectivo de estudiantes. En

Pelican esto sí es posible ya que el modelo de colaboración basado en espacios de trabajo compartidos permite que distintas herramientas de soporte (típicamente repositorios de objetos de aprendizaje) se integren para soportar la construcción colaborativa de los artefactos involucrando a usuarios tanto de un mismo grupo como de grupos diferentes. En tercer lugar, los autores destacan la ausencia de mecanismos de especificación declarativa para capturar todos los aspectos dinámicos de un escenario colaborativo. En Pelican este requerimiento queda soportado por el modelo de intervención. En cuarto lugar, se echan de menos capacidades para realizar el control de flujo de trabajo. En Pelican el control de flujo puede ser establecido mediante reglas políticas que dictaminen la lógica de transición entre actividades organizadas de acuerdo a diferentes rutas instruccionales tal y como se discutió en el apartado 3.4. Y finamente, los autores reclaman la necesidad de cambar el enfoque de IMS hacia un modelo centrado en la actividad colaborativa destacando la importancia de la configuración adaptativa para que la implantación de un mismo escenario pueda ser diferente en cada grupo. En efecto, la actividad y el proyecto en Pelican son los dos elementos claves del modelo de colaboración.

4.2. IMS – TI

La especificación de IMS para la interoperabilidad entre herramientas [IMS – TI], propone un mecanismo para integrar sistemas desarrollados por terceras partes dentro de plataformas de aprendizaje. La arquitectura general para alcanzar tales objetivos se expresa en términos de 3 elementos fundamentales: 1) las herramientas externas que prestan una colección de servicios de soporte a la plataforma; 2) los proxies, artefactos asociados a cada herramienta que sirven de adaptador con la plataforma y 3) el entorno de ejecución de interoperabilidad, que se añade a la plataforma para poder comunicarse con las herramientas a través de las operaciones definidas en los proxies. La comunicación entre este último elemento y los proxies de las herramientas se lleva a cabo haciendo uso de la tecnología de servicios Web.

La solución planteada consiste en una arquitectura de componentes que son gobernados de forma centralizada a través de un ciclo de vida. 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 se expresa la lógica de integración y, por tanto, puede considerarse que es 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.

4.3. Arquitectura OGSA con servicios en Grid

Para dar respuesta a los problemas apuntados en la especificación de IMS, en [Hernández – Leo, 2003] se propone una extensión de la misma que incorpora el concepto de servicio de grupo. Un servicio de grupo permite describir el uso colaborativo de una herramienta especificando la información de contexto que es necesario proporcionar a los estudiantes para que la utilicen, las políticas de uso, la naturaleza de la comunicación que soportan, los roles que intervienen y los tipos de interacción.

A partir de esta extensión [Bote – Lorenzo et al., 2004] proponen una arquitectura de integración basada en el uso de servicios articulados de acuerdo al estándar arquitectónico OGSA (Open Grid Service Architecture) [Foster et al., 2003]. Esta aproximación propone construir un entorno de aprendizaje conjugando una colección de servicios en Grid desarrollados por terceras partes que dan soporte a ciertas interacciones colaborativas. Los servicios 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 dos ideas, 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 que han sido resueltos en nuestra propuesta. En primer lugar los servicios Grid deben ser desarrollados de acuerdo a un estándar específico para conseguir la integración. En Pelican por el contrario existen 4 niveles de integración con diferente grado de acoplamiento. En segundo lugar, si bien la propuesta arquitectónica de Grid también utiliza un middleware para soportar la integración, éste no es utilizado para descargar el trabajo de coordinación necesario. En su lugar, tal responsabilidad es delegada a las herramientas. En Pelican por el contrario, las tareas de orquestación y diálogo entre herramientas son especificadas declarativamente mediante el lenguaje de dominio P# de la plataforma. Y finalmente, el middleware propuesto no parece ofrecer ningún mecanismo centralizado para soportar las sesiones de trabajo de los estudiantes. Nuevamente esta labor se delega a las herramientas con todos los problemas de integración que ello conlleva. En Pelican, sin embargo las sesiones de trabajo son utilizadas para mantener en todo momento los modelos de usuario y el estado de la colaboración, información que, como vimos, resulta útil para contextualizar el uso de las herramientas y de las interacciones en la plataforma.

4.4. OSGi

El marco de trabajo OSGi (Open Services Gateway iniciative) [OSGi], constituye una especificación de un modelo de componentes completo para la plataforma J2SE / J2EE. 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 java 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, estos 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, PDAs, 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 un modelo arquitectónico.

5. Conclusiones

La plataforma que hemos presentado en este artículo es una propuesta que ofrece algunas ventajas importantes con respecto a otras ya existentes

[Moodle] [Web - CT]. En concreto, el modelo social y colaborativo conjugan elementos de modelado y meta-modelado para ofrecer potencia y flexibilidad en la definición de escenarios pedagógicos. No obstante el foco de interés de la plataforma Pelican reside en el sistema de integración y el sistema de intervención.

La integración es una cuestión que cada vez preocupa más a investigadores, docentes y desarrolladores. Una de las demandas que hacen los usuarios, a este respecto, es encontrar una forma de incorporar en un mismo entorno diferentes herramientas heterogéneas. Esta unión trae consigo un efecto sinérgico en el que las herramientas se benefician de un entorno potente en los aspectos organizativos y éste, a su vez, se ve enriquecido por la incorporación de nuevos servicios específicos de soporte a la colaboración. El sistema de integración de Pelican proporciona una solución en este sentido más ligera y flexible que otras propuestas [IMS – LD] [Bote – Lorenzo et al., 2004] ya que no fuerza a los desarrolladores de las herramientas externas a acogerse a ciertos estándares y protocolos de desarrollo. En concreto se pueden destacar tres principios fundamentales de este sistema: 1) garantizar una integración organizada en niveles de interoperabilidad y orientada a la interacción, 2) basar la misma en el uso de roles como artefacto conceptual y tecnológico para soportar la adaptación de las herramientas externas al entorno y 3) utilizar un mecanismo declarativo basado en reglas y servicios Web para orquestar la funcionalidad de los servicios proporcionados por las herramientas.

Por su parte, el modelo de intervención ofrece la posibilidad de capturar todos los aspectos dinámicos de la especificación de escenarios colaborativos. Mediante un mecanismo de planificación de la ejecución de scripts adaptativos basado en reglas y dirigido por eventos es posible indicar el momento exacto, dentro del flujo de instrucción, en el que las transformaciones de configuración del entorno deben producirse. Las posibilidades que ofrece este sistema permiten cubrir gran parte de las necesidades de los procesos que intervienen en el aprendizaje colaborativo. Así, es posible realizar modificaciones sobre la estructuración social y colaborativa de un escenario, definir reglas para la monitorización y el control de los estudiantes dentro de cada actividad y automatizar tareas de evaluación basada en la supervisión de productos generados. Sin embargo, esta especificación mediante reglas es aún una ardua labor para usuarios con un perfil no técnico, problema que trataremos de abordar en futuros trabajos.

Agradecimientos

Proyecto ENLACE. Un Entorno Basado En Agentes Para Comunidades De Aprendizaje (escuela y naturaleza, lugares para aprender, colaborar y experimentar). TIN 2004 – 04232.

Referencias

Blackboard. http://www.blackboard.com

Bollen, L. Harrea, A. Hoppe, U. Joolinger, W.R. 2007. “A broker architecture for integration of heterogeneous applications for inquiry learning”. 7th IEEE International Conference on Advanced Learning Technology (ICALT) 2007

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

Caeiro et al. 2006. Caeiro, M. Llamas, M. Anido-Rifón, L.E. “The PoEML Proposal to Model Services in Educational Modeling Languages”. CRIWG 2006, 187-202.

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

Colab. http://www.co-lab.nl/

Dillenbourg & Baker, 1996. Dillenbourg, P. Baker, M. J. 1996. “Negotiation spaces in human – computer collaboration.” En las actas de COOP 1996, Second International Conference on Design

of Cooperative Systems, pp. 187-206, INRIA, Juan les pins, Junio 1996

Dimitrakopoulou, A. 2005. "State of the art of interaction analysis for metacognitive support and diagnosis." D.31.1.1 IA Project. Prepared for the European Commission, DG INFSO, under contract ITS 507838 as a deliverable from WP 31

Foster, I., Kesselman, C., Nick, J. M., Tuecke, S. 2003. "The Physiology of the Grid". In Berman, F., Fox, G. Hey, A. (eds.): Grid Computing: Making the Global Infrastructure a Reality. John Wiley & Sons, Chichester, UK pp. 217-249

Hernández-Leo, D. 2003. "From IMS-LD to MDA: Approaches to the modelling of CSCL applications based on components (in Spanish)", MSc Thesis, Valladolid, Spain: University of Valladolid.

IMS - LD. www.imsglobal.org/learningdesign

Johnson et al., 1998. Jhonson, D.W. Jhonson, R.T. Smith K.A. 1998. "Active learning: Cooperation in the college classroom". Edina, MN: Interaction Book Company

Joolinger, W.R. De Jong, T. Manlove, S. 2007. "CIEL: Setting the state for integrated inquiry learning". AERA annual meeting. Chicago, Abril 9-13 2007.

Koschman, J. 1996. "CSCL: Theory and Practice of an Emerging Paradigm." Malwah, N.J.: Lawrence Erlbaum Assoc.

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.

Moodle. http://moodle.org/

MSpace. http://modellingspace.net

Sharan, S. 1994. "Handbook of Cooperative Learning Methods." Westport, CT. USA.

Suchman, 1987. Suchman, L. 1987. "Plans and situated actions: The problem of human machine communication". Cambridge University Press.

Vélez & Verdejo, 2007. 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

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, 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

Verdejo, M. F. Celorrio, C. Lorenzo, E. J. Sastre, T. 2006. "An educational networking infrastructure supporting ubiquitous Learning for School Students". 6th IEEE International Conference on Advanced Learning Technologies. pp. 174-178. Kerkrade. The Netherlands, July 2006.

Web-CT. http://www.webct.com