Primera Aproximación al Diseño e Implementación de los DHD

Alejandro Sartorio, Maximiliano Cristiá

[1] CIFASIS-CONICET
Bv. 27 de Febrero 210 Bis Rosario, 2000 (República Argentina)
[2] FCEIA, Universidad Nacional de Rosario
{sartorio,cristia}@cifasis-conicet.gov.ar

Resumen En este trabajo[3] se presenta una propuesta de diseño e implementación de espacios Web e-learning con propiedades de reconfiguración de servicios en tiempo de ejecución, brindando un nuevo mecanismo de adaptación y personalización con características de sensibilidad al contexto.
Se describe un modelo de integración para la incorporación de "coordinación de contratos sensibles al contexto" a un framework e-learning. La implementación fue resuelta utilizando patrones de diseño que permiten superposición de reglas, a través de contratos, entre las interacciones de los objetos que implementan servicios (ej., edición de mensajes en un Foro) y los objetos que lo invocan.

1. Introducción

A medida que el avance en la investigación y desarrollo de plataformas e-learning brindan mejoras e innovaciones en herramientas (videoconferencias, porfolios, wikis, workshops, etc.) y sus respectivos servicios, crece la cantidad de posibles configuraciones de los espacios e-learning. Estas configuraciones abarcan diferentes tipos de requerimientos pertenecientes a las etapas de diseño, desarrollo e incluso exigen que el espacio e-learning se adapte en tiempo de ejecución. A partir de estos requerimientos se definen los procesos e-learning (que nosotros denominamos Pe-lrn) [4] de manera semejante a procesos de negocio en otro dominio de aplicación. Al igual que los procesos de negocios en una Aplicación Web convencional, los Pe-lrn están compuestos por transacciones Web [4]. En este contexto, una transacción (o transacción e-learning) es definida como una secuencia de actividades que un usuario ejecuta a través de una Aplicación e-learning con el propósito de efectuar una tarea o concretar un objetivo, donde el conjunto de actividades, sus propiedades y las reglas que controlan sus ejecuciones dependen del Pe-lrn que la Aplicación debe brindar. Un ejemplo de

[3] Este trabajo pertenece a la tesis doctoral en curso sobre "Contratos para Context-Aware" (UNLP- Dir. Gustavo Rossi, Codir. Dra. Patricia San Martín (CIFASIS) del becario Lic. Alejandro Sartorio en el marco del Proyecto "Técnicas de Ingeniería de software Aplicadas al Dispositivo Hipermedial Dinámico" (Dir. Maximiliano Cristiá)

estrategia didáctica es la posibilidad de que un alumno acceda a determinado tipo de material (vídeos, archivos, etc.) dependiendo de sus intervenciones en los Foros. Estos tipos de requerimientos resultan difíciles de implementar con las actuales aplicaciones e-learning de extendido uso a nivel global.

En el marco de los análisis efectuados podemos sostener que el proyecto Sakai wwww.sakaiprojet.org brinda una de las propuestas más consolidadas de diseño y desarrollo de entornos colaborativos e-learning para educación, orientado a herramientas que se implementan a través de servicios comunes (servicios bases). Por ejemplo, el servicio de edición de mensajes es utilizado en las herramientas Foro, Anuncio, Blog, PorFolio, etc. Más aun, otras de las características salientes de Sakai es la versatilidad para su extensión y/o configuración. En efecto, Sakai permite alterar ciertas configuraciones en tiempo de ejecución, por ejemplo, instrumentar una nueva funcionalidad en un servicio base de Sakai.

Sin embargo, estas soluciones no pueden resolver aquellos Pe-lrn que involucren cambios en el comportamiento de la relaciones entre un componente (cliente) que ocasiona, a través de un pedido, la ejecución de un componente servidor (proveedor). Estos tipos de cambios refieren a la capacidad de adaptación dinámica del sistema [6] extendiendo, personalizando y mejorando los servicios sin la necesidad de recompilar y/o reiniciar el sistema. En este trabajo se presenta un propuesta para la incorporación (agregado) de propiedades de adaptación dinámicas a los servicios bases del framework Sakai, especialmente diseñadas para implementar Pe-lrn donde se requieren nuevos aspectos de adaptación [1] en la perspectiva de los Dispositivos Hipermediales Dinámicos en el campo del e-learning con característica *context-aware* [1]. Se entiende por DHD *"a una red social mediada por las TIC en un contexto físico-virtual-presencial, donde los sujetos investigan, enseñan, aprenden, dialogan, confrontan, evalúan, producen y realizan responsablemente procesos de transformación sobre objetos, regulados según el caso, por una coordinación de contratos integrados a la modalidad participativa del Taller".*(San Martín et al, 2008)

2. Contractos con características sensibles al contexto

Nuestra propuesta de solución a los requerimientos mencionados sobre adaptación dinámica comienza con la construcción de un modelo de contrato orientado a la implementación de servicios sensibles al contexto. El uso de contratos parte de la noción de Programación por Contrato ("Programming by Contract") de Meyer [5] basada en la metáfora de que un elemento de un sistema de software colabora con otro, manteniendo obligaciones y beneficios mutuos. En nuestro dominio de aplicación consideraremos que un objeto cliente y un objeto servidor "acuerdan" a través de un contrato (representado con un nuevo objeto) que el objeto servidor satisfaga el pedido del cliente, y al mismo tiempo el cliente cumpla con las condiciones impuestas por el proveedor. Como ejemplo de la aplicación de la idea de Meyer en nuestro dominio de sistemas e-learning planteamos la situación en que un usuario (cliente) utiliza un servicio de edición de mensajes (servidor) a través de un contrato que garantizará las siguiente condiciones:

el usuario debe poder editar aquellos mensajes que tiene autorización según su perfil (obligación del proveedor y beneficio del cliente); el proveedor debe tener acceso a la información del perfil del usuario (obligación del cliente y beneficio del proveedor).

A partir de la conceptualización de contratos según Meyer se propone una extensión por medio del agregado de nuevas componentes para instrumentar mecanismos que permitan ejecutar acciones dependiendo del contexto.

En aplicaciones sensibles al contexto [3], el contexto (o información de contexto) es definido como la información que puede ser usada para caracterizar la situación de una entidad más allá de los atributos que la definen. En nuestro caso, una entidad es un usuario (alumno, docente, etc.), lugar (aula, biblioteca, sala de consulta, etc.), recurso (impresora, fax, etc), u objeto (examen, trabajo práctico, etc.) que se comunica con otra entidad a través del contrato. En [1] se propone una especificación del concepto de contexto partiendo de las consideraciones de Dourish [9] y adaptadas al dominio e-learning, que será la que consideraremos en este trabajo. Contexto es todo tipo de información que pueda ser censada y procesada, a través de la aplicación e-learning, que caracterizan a un usuario o entorno, por ejemplo: intervenciones en los foros, promedios de notas, habilidades, niveles de conocimientos, máquinas (direcciones ip) conectadas, nivel de intervención en los foros, cantidad de usuarios conectados, fechas y horarios, estadísticas sobre cursos, etc.

En términos generales, la coordinación de contratos es una conexión establecida entre un grupo de objetos aunque en este trabajo se consideran sólo dos objetos: un cliente y un servidor. Cuando un objeto cliente efectúa una llamada a un objeto servidor (ej., el servicio de edición de la herramienta Foro), el contrato "intercepta" la llamada y establece una nueva relación teniendo en cuenta el contexto del objeto cliente, el del objeto servidor, e información relevante adquirida y representada como contexto del entorno [9].

A continuación se brindarán detalles sobre algunas de los componentes y relaciones esenciales para la integración de este modelo con el framework Sakai y con los módulos que instrumentan la coordinación de contratos.

2.1. Elementos de los contratos sensibles al contexto

Un contrato que siga las ideas de Meyer contiene toda la información sobre los servicios que utilizarán los clientes. Para incorporar sensibilidad al contexto nuestros contratos deberán tener referencias sobre algún tipo de información de contexto para su utilización.

En el diagrama de relaciones entre entidades mostrado en la Figura 1 se describen los elementos que componen el concepto de contrato sensible al contexto, especialmente para Pe-lrn.

La figura comienza con la representación de un contrato según Meyer donde se caracterizan los principales elementos que lo componen (pre-condiciones, acciones, pos-condiciones). La flechas salientes de la zona gris indican los dos tipos de relaciones (acción-servicio e invariante-contexto) que se debe instrumentar para incorporar un mecanismo que provea a los contratos la característica de

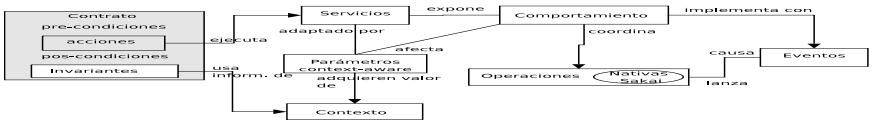

Figura 1. *Modelo de un contrato context-aware*

sensibilidad al contexto. En la porción derecha de la Figura 1 aparecen las entidades necesarias para obtener contratos sensibles al contexto. Las explicamos a continuación.

- *Servicios*: En esta componente se representan los elementos necesarios para la identificación de un servicios y clasificación de los servicios que pueden formar parte de las acciones de los contratos. Por ejemplo, nombre del servicio, identificadores, alcance, propósito, etc.

- *Comportamiento*: El comportamiento de un servicio se logra a partir de combinar operaciones y eventos que son representadas con el las componente *Operaciones* y *Eventos*. De la misma manera el servicio puede ser implementado a través del uso de eventos, representados con el componente *Eventos*, que puede lanzado operaciones del componente *Operaciones*. Por ejemplo, de acuerdo con los roles (ej., alumno, instructor, docente, etc.) asignados a un usuario de una herramienta involucrado en un Pe-lrn, en un determinado contexto del entorno (ej., si está en un espacio Foro) y del usuario (ej., si tiene permiso de moderador), la componente *servicio* brindan distintas funcionalidades (ej., editar un mensaje), que son instrumentadas por medio de operaciones concretas (ej., guardar un mensaje en una tabla) y/o a través de la publicación o subscripción de eventos.

- *Parámetros Context-Aware*: Se denomina *parámetros context-aware* a la representación de la información de contexto que forma parte de los parámetros de entrada de las funciones y métodos exportados por los servicios, estableciendo de esta manera una relación entre el componente *Servicios* y el componente *Parámetros contex-aware*.

- *Contexto*: Para nuestro modelo este tipo de información es utilizada de dos maneras diferentes: en primer lugar para la asignación de los valores que toman los *Parámetros context-aware*; en segundo lugar esta información puede ser utilizada para definir los invariantes que se representan en los contratos.

3. Modelo de integración de Sakai con contratos

En esta sección se muestra un esbozo de la propuesta para incorporar a Sakai un mecanismo de coordinación de contratos sensibles al contexto, a través de la presentación de una diseño que describe la comunicación entre módulos [7] y sus dependencias. En la figura 2 los módulos son representados con paquetes UML y las relaciones entre ellos por medio de flechas que indican comunicación y dependencias. A su vez, en un segundo plano se muestra cuáles son las clases específicas de cada módulo y cómo se implementa la integración a través de estas.

El framework Sakai, representado en la figura 2 por el módulo Sakai, está diseñado según una arquitectura de cuatro capas: La capa de **agregación** se en-

carga completamente de la implementación de la interfaz con el usuario al estilo de las implementaciones de portales Web. La segunda capa denominada **presentación** tiene la responsabilidad de permitir la reutilización de los componentes Web (ej., "widget"que proveen calendarios, editores "WSYWIG", etc.). La tercera corresponde a la capa de **herramientas** donde reside la lógica de negocio de las herramientas Sakai que interactúan con el usuario (ej., Foro, Wiki, etc.). Por último la capa de **servicio** implementa los servicios Sakai bajo una Arquitectura Orientada a Servicios que serán utilizados por varias herramientas a través de API.

Para nuestra propuesta de integración tendremos en cuenta únicamente servicios que componen la capa de **servicios** pertenecientes a los servicios bases Sakai. Los servicios del núcleo Sakai serán envueltos mediante la coordinación de contratos, lo que se representa en la Figura 2 con las referencias del **Módulo Sakai** hacia el **Módulo de Coordinación de Contractos**.

De esta forma, se logra controlar las invocaciones a los servicios del núcleo Sakai a través del **módulo Coordinación de Contratos**, quedando establecida una primer relación de integración. Esta misma relación es implementada a nivel de clases como una relación de agregación entre la clase *Servicios Sakai* (correspondiente al módulo Sakai) y la clase *Conector* correspondiente al framework de coordinación que será tratado en la sección 4 a través del uso de patrones de diseño.

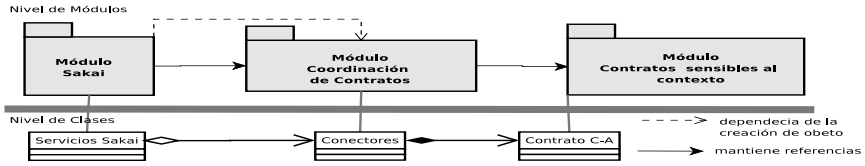

Figura 2. *Diseño de la integración de Sakai con Contratos*

La última relación de integración se produce entre el módulo de **Coordination de Contratos** y el módulo de **Contratos Sensible al contexto** el cual fue desarrollado en la sección 2. En este caso, la figura 2 muestra cómo desde el framework de coordinación se estable una relación de composición entre la clase *Conector* y la clase *Contrato C-A* que implementa el elemento Contrato representado en la Figura 1 coloreada en gris.

De esta forma, instanciando dinámicamente diferentes contratos, Sakai presentará una flexibilidad en tiempo de ejecución que actualmente no posee y que es la que requieren los DHD.

4. Implementación de contratos en Sakai

En las secciones anteriores se mencionó al contrato como un agente activo que se encarga de la coordinación de las componentes que lo integran. En es-

ta sección se describe cómo este mecanismo fue implementado a través de una herramienta que permite modificar implementaciones de clases Java de los servicios Sakai para incorporarle mecanismos de coordinación de contratos sensibles al contexto, según el diseño mostrado en la Figura 2. Sin embargo creemos que antes es conveniente resumir brevemente otros intentos de proveer la flexibilidad requerida por los DHD.

4.1. Niveles de flexibilidad de Sakai

Diferentes estándares y frameworks de desarrollo de software basados en componentes e inyección de servicios fueron propuestos para mejorar la flexibilidad y adaptabilidad [4] de los sistemas e-learning Web, los más importantes son: CORBA, JavaBeans, Hibernate, Spring, JSF, RSF, etc. Sin embargo, ninguno de estos estándares proveen un mecanismo convenientemente implementativo y abstracto en donde se permita representar a las relaciones entre usuarios y servicios como un objeto de primera clase [5]. Esta idea se relaciona con las soluciones informáticas en la que se refuerza la pontencialidad de un lenguaje en la implementación de un servicio perteneciente a la capa de servicios en vez de la capa presentación. Los primeros antecedentes de este tipo de soluciones fueron utilizados para sistemas de comunicación aplicadas en la capa de los datagramas para el cambio de protocolos a través de reglas [8]. En este sentido, proponemos una solución utilizando patrones de diseño que implemente características deseadas de la TCCs-c sobre la capa de servicios base de framework Sakai.

Desde un punto de vista implementativo, esta propuesta cambia la filosofía Java2EE-Sakai[5] donde la incorporación de nuevas prestaciones de versatilidad para la configuración e implementación se logran a través de extensiones de clases, relaciones a liberías, técnicas de reflexión ("reflection"), etc. En el siguiente diagrama de clases UML se muestra el diseño propuesto para incorporar coordinación de contratos sensibles al contexto en Sakai.

4.2. Patrones de diseño para la coordinación de contratos sensible al contexto

- *MétodoHerramineta_Interfaz*: Como su nombre lo indica, esta es una clase abstracta que define una interfaz común de los servicios de una Herramienta provisto por *MétodoHerramienta_Proxy* y *MétodoToProxy_Adaptador*.

- *MétodoToProxy_Adaptador*: Esta es una clase concreta que implementa a un intermediario manteniendo la referencia a la clase *Método_Proxy* para encargarse de los pedidos recibidos. En tiempo de ejecución, esta entidad puede indicar a *MétodoServiceComponente* que no hay contratos involucrados o instrumentar una conexión entre

[4] La adaptabilidad en el sentido que se proponen en los sistemas hipermediales que permiten personalizarse (adaptarse) en función de los usuarios individuales (Henze, 2000).

[5] Sakai mantiene la filosofía de desarrollo de Java2, de aplicaciones orientadas a servicios ("service-oriented") que pueden ser escalables, fiable, interoperable y extensible.

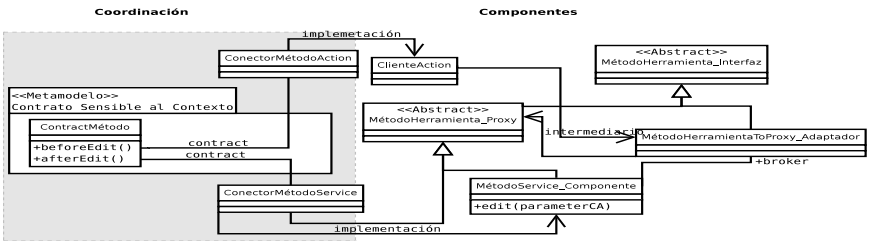

Figura 3. *Diseño del modelo de integración*

ConectorMétodosService para que conecte la clase real *MétodoServiceComponente* con el contrato *ContractMétodo* en el que se encuentran las reglas de coordinación.

- *MétodoHerramienta_Proxy*: Es una clase abstracta que define la interfaz que tienen en común *MétodoHerramientaToProxy_Adaptador* y *ConectorMétodoService*. La interfaz es heredada de *MétodoHerramienta_Interfaz* con el propósito de garantizar que ofrezca la misma interfaz de *MétodoHerramientaToProxy_Adaptador* con la cual un cliente real debe interactuar (ej., un objeto que invoque el servicio de edición de un Foro).

- *EditServiceComponente*: Es la clase concreta donde se encuentra la lógica de la edición del Foro y la implementación concreta de los servicios originales de Sakai.

- *ConectorMétodoService*: Efectúa la conexión entre el contrato y los objetos reales (en este caso *MétodoServiceComponente*) involucrados como partes del contrato. De esta manera, no se requiere la creación o instanciación de nuevos objetos para coordinar el mismo objeto real agregando o sacando otros contratos. Esto es posible solamente con una nueva asociación a un nuevo contrato y con la instanciación de una conexión con una instancia existente de *ConectorMétodoService*. Esto significa que hay una sola instancia de esta clase asociada con una instancia de *MétodoServiceComponente*.

- *ContractMétodo*: Es la clase que cuya instancia genera un objeto de coordinación que al ser notificado toma decisiones siempre que un pedido es invocado a través de un objeto real. Para los casos en que no haya contratos coordinando un objeto real, el diseño de la figura 3 puede ser simplificado y sólo las clases y relaciones que no pasen por la zona del rectángulo gris son necesarias. La clase que implementa al contrato, llamada *ContractMétodo*, se encuentra representada dentro de la figura de un paquete UML, llamado *Contratos Sensible al Contexto*, indicando su pertenencia a un modelo subyacente descripto en la sección 2.

Por lo expuesto, es válido reforzar la argumentación sobre las ventajas de esta propuesta en comparación con las actuales soluciones tecnológicas basadas en componentes y frameworks para la inyección y representación de servicios. A su vez, se observa que el camino de su instrumentación es complejo debido a que se deben generar (automáticamente) porciones de código y creación de nuevos archivos código Java, a partir de los originales de la plataforma Sakai, que deberán ser compilados y puestos en servicio para su operatividad. Cabe destacar que los procesos de compilación y puesta en servicio de Sakai es efectuada una sola vez, cualquier tipos de cambios y configuración serán impuestas a través los contratos sensibles al contexto.

5. Caso de estudio para la implementación de contratos

En esta sección se presentará un caso de uso para ejemplificar las diferentes etapas que se deben cumplimentar para lograr la inclusión de contratos en Sakai bajo la perspectiva de los DHD, retomando la idea de incluir un contrato en los servicios de edición de la herramienta Foro de Sakai.

Siguiendo con el caso de estudio de las etapas anteriores, a continuación se muestra una porción de código Java correspondiente al código fuente original de Sakai, en donde se propone un método llamado **editMenssage** correspondiente a la implementación del servicio edición de la herramienta Foro. El siguiente fragmento de código fuente Java implementa una herencia de la clase *Base-DiscussionService* donde se encuentran las interfaces de los servicios de edición correspondiente al núcleo del framework Sakai.

```
import org.sakaiproject.discussion.api.DiscussionMessage;
public class DiscussionService extends BaseDiscussionService{
public MessageEdit editMessage(MessageChannel channel, String id){
return (MessageEdit) super.editResource(channel, id);}
```

Luego, a través de la herramienta SwC se crean archivos XML (similar a la herramienta CED) para especificar las reglas del contrato que involucrarán servicios implicados en el método *editMensage* (Paso1). A continuación, se ejecuta la generación automática de código para crear dos tipos de archivos Java diferentes (Paso2). El primero (archivo 1) implementa las funcionalidades que permitan las conexión con el Proxy (ej., *MétodoHerramienta_Proxy* para nuestro caso de estudio); el segundo (archivo 2) implementa el conector *ConectorEditService* representado en la figura 3. A continuación se muestran fragmentos de código que permiten una mejor ilustración de las características adquiridas a partir de las modificaciones del código fuente de la aplicación Sakai a través de la herramienta.

Frag. 1 del archivo 1. Se importan los paquetes correspondiente al framework de coordinación de contratos (figura 3) y el framework context-aware (figura 1)

```
import cde.runtime.*; import obab.ca.*; // Framework context
public abstract class DiscussionService extends
BaseMessageService implements
DiscussionService,ContextObserver,EntityTransferrer,ForoInterface
```

Frag. 2 del archivo 1. Métodos agregados por la herramienta para identificar las clases que van a ser interceptadas por el contrato. SetProxy se encarga del la instanciación de las componentes de Proxy.

```
public static Class GetClassId() {return _classId;}
public CrdIProxy GetProxy() { return _proxy; }
public void SetProxy(Object p){if(p instanceof CrdIProxy && p instanceof
DiscussionInterface) _proxy = (CrdIProxy)p; }
public void SetProxy(CrdIProxy p)  { _proxy = p; }
AccountInterface  GetProxy_Account(){if ( _proxy == null ) return null;
return (DiscussionInterface) _proxy.GetProxy(_classId);}
```

Frag. 3 del archivo 1. Método agregado que implementa la llamada del objeto cliente al Proxy.

```
public long _getNumber(){new ComponentOperationEvent(this , "getNumber")_
.fireEvent(); return number;}
public messageEdit editMessage(MessageChannel channel,String id){
new ComponentOperationEvent(this,"Edit").fireEvent();
return (MessageEdit) super.editResource(channel, id);}
```

Frag. 1 del archivo 2. Porción de código donde se importan las componentes de los frameworks, se heredan las clases abstracta del los conectores y proxys. La clase del objeto real *MétodoService_Componente*, según la figura 3, se representa a través del atributo subject.

```
import cde.runtime.*; import obab.ca.*;
public abstract class IDiscussionPartner extends
CrdContractPartner implements CrdIProxy, DiscussiontInterface {
protected  Discussion subject;
```

Frag. 2 del archivo 2. Definición de los métodos abstractos para la conexión (*ConnectorMétodoService*, representada en la figura 3), del contrato con el servicio.

```
public void SetProxy(Object p) {subject.SetProxy(p);}
protected Object GetSubject_Object() {return subject;}
public void ResetProxy() { subject.SetProxy(null);}
```

Frag. 3 del archivo 2. Métodos que permiten el acceso a los métodos que definen los servicios (ej., métodos de la clase *MétodoService_Componente*).

```
protected Discussion GetSubjectDiscussion(){return (Discussion) subject;}
protected IDiscussionPartner GetNextPartner_Discussion(){
return (IDiscussionPartner)GetNextPartner(Discussion.GetClassId());}
```

Frag. 4 del archivo 2. Implementación por defecto de métodos definidos en la interfaces de los servicios. A través del métodos *GetSubjectDiscussion()*, detallado en el punto anterior, se accede a los métodos creados por la herramienta en el primer archivo.

```
public void messageEdit (double amount,Customer c)_
throws DiscussionException { IDiscussionPartner
next = GetNextPartner_Discussion()
if (next != null) next.editMessage(amount,c);
else GetSubjectDiscussion()._editMessage(amount,c);}
```

Frag. 5 del archivo 2. Implementaciones de los condicionales de la reglas de los contratos que se encuentran en los archivos XML para configurarlos.

```
public CrdPartnerRules  messageEdit_rules(string texto,Student c) throws
DiscussionException, CrdExFailure { return new CrdPartnerRules (this);}
```

La generación de código automático a través de la herramienta presupone tener ciertos niveles de conocimientos sobre: el lenguaje Java, aspectos de la implementación del framework base Sakai y los frameworks que los componen; a igual que experiencias en la compilación a través de Maven [6] de Sakai y las clases modificadas-agregadas para el uso de contratos. Este aspecto denota una posible limitación en la implementación de la idea propuesta en este trabajo.

[6] Proyecto maven: http://maven.apache.org/ref/2.0.4/maven-project/

6. Conclusión

Fundamentados en el marco conceptual de los Dispositivos Hipermediales Dinámicos para educación e investigación y en las actuales limitaciones observadas en las plataformas e-learning sobre el grado de flexibilidad adaptativa en el sentido de [10] y [6]), que pueda ser inducida por usuarios expertos del dominio (ej., docentes) para el desarrollo de estrategias didácticas efectivas para el logro de objetivos pedagógicos o investigativos planteados y, constando que dicha adaptabilidad no pueda ser enteramente resulta por sistemas adaptativos inteligentes (ej., agentes, hipermedia adaptativa, sistemas expertos, etc.), es que en este trabajo propusimos una implementación de la teoría de coordinación de contrato en un framework específico sobre sistemas colaborativos Web e-learning (correspondiente al proyecto Sakai) brindando un modelo evolutivo conceptual y tecnológico.

El fin último de esta propuesta en desarrollo, se inscribe en brindar respuestas tecnológicas efectivas a la necesidad de promover procesos educativos e investigativos de interacción responsable entre los sujetos intervinientes en la red sociotécnica del DHD, implementando la solución en los servicios cuya prestaciones dependen -fuertemente- de reglas con estructuras volátiles cuyos condicionales son influidos por el contexto (en el sentido de "Identificación del contexto"desarrollado en el capítulo 5 [1] y manteniendo los lineamientos sobre contexto fundados por P. Dourish [9]).

Referencias

1. San Martín, P., Sartorio, A., Guarnieri, G., Rodriguez, G.: Hacia un dispositivo hipermedial dinámico. Educación e Investigación para el campo audiovisual interactivo. Universidad Nacional de Quilmes (UNQ). ISBN: 978-987-558-134-0. (2008)
2. Sartorio, A.:Un modelo comprensivo para el diseño de procesos en una Aplicación E-Learning. CACIC 2007. ISBN 978-950-656-109-3. (2007)
3. Dey, A.K., Salber, D., Abowd, G.: A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. Human-Computer Interaction (HCI) Journal, Vol. 16 (2-4), pp. 97-166. (2001)
4. Brambilla, M., Ceri, S., Fraternali, P., Manolescu I.: Process modeling in web applications. ACM (TOSEM). (2006)
5. Meyer, B.:, Applying Design by Contract, IEEE Computer, 40-51. (1992)
6. Dowling J, Cahill, V.: Dynamic software evolution and the k-component model. In: Proc. of the Workshop on Software Evolution, OOPSLA. (2001)
7. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.: Pattern- Oriented Software Architecture. John Wiley. (1996)
8. Koutsoukos, G., Gouveia, Andrade, Fiadeiro, L.: Managing evolution in Telecommunications Systems, IFIP Working Conference on Distributed Applications and Interoperable Systems, Kluwer. (2001)
9. Dourish, P.: What we talk about when we talk about context. Personal and Ubiquitous Computing, vol. 8, N° 1, Roma, 2004, pp. 19-30. (2004)
10. Houben, G.: Adaptation Control in Adaptive Hypermedia Systems. en Adaptive Hypermedia Conference. AH2000. Trento, Italia. Agosto. LNCS, vol. 1892. Springer-Verlag, pp. 250-259. (2000)