Experiencia Práctica de la Aplicación de Aproximaciones Orientadas por Aspectos en el Desarrollo de un Portal Temático

Practical Experience of the Application of Aspect-Oriented Approaches in the Development of a Thematic Portal

Juan B. Quintero[1], MSc., Diana M. Hernández[2], Ing. y Raquel Anaya de P.[1], PhD
1. Grupo de Investigación en Ingeniería de Software, Universidad EAFIT, Medellín, Colombia.
2. Departamento de Sistemas, ABC-Flex Ltda. Medellín, Colombia,
jquinte1@eafit.edu.co, dianahdez@abcflex.net, , ranaya@eafit.edu.co

Recibido para revisión 26 de Marzo de 2007, aceptado 15 de Junio de 2007, versión final 19 de junio de 2007

Resumen— Este trabajo muestra la experiencia de aplicación de diferentes aproximaciones orientadas por aspectos para la definición, especificación e implementación de un portal temático. En la fase de requisitos, se utiliza un desarrollo dirigido por casos de uso para las características funcionales y el *Framework* NFR para las consideraciones de calidad. En la fase de análisis y diseño, se aplicó Theme/UML para especificar dos de las funcionalidades claves del portal: la asociación de temas y las búsquedas. En la fase de implementación, se fusionó el uso de librerías orientadas por aspectos con *frameworks* de desarrollo en PHP; la implementación de una de las funcionalidades transversales se realizó utilizando el patrón de diseño *Strategy*. La experiencia de este caso de aplicación sirve de referente en la adopción de aproximaciones orientadas por aspectos en el proceso de desarrollo, ilustrando la manera de mantener la trazabilidad entre los artefactos involucrados desde los requisitos hasta la implementación.

Palabras Clave— Orientación por Aspectos, Casos de Uso, Patrones y Frameworks.

Abstract— This work shows the experience of application of different aspect-oriented approaches for the definition, specification and implementation of a thematic portal. In the requirements phase, a use case driven development has been used for functional features and the NFR framework was used for quality considerations. Theme/UML was applied to specify two key functionalities of the portal: association of topics and search. At the implementation phase, use of aspect libraries has been fused with development frameworks written in PHP; one of the traverse functionalities was implemented using the Strategy design pattern. The experience taken from this case of application serves as a starting point in the adoption of aspect-oriented approaches in the software development process, illustrating the way to maintain the traceability among the assets involved from requirements to implementation.

Key words— Aspect Oriented, Use Cases, Patterns and Frameworks.

I. INTRODUCCIÓN

TRAS el surgimiento de la programación orientada por aspectos se han realizado diversos esfuerzos para integrar sus principios a lo largo de todo el proceso de desarrollo de software. Las estrategias propuestas en la orientación por aspectos para la separación de intereses involucran nuevas unidades para el reuso en las diferentes etapas del proceso que se derivan de los intereses cruzados (*crosscuting*

concerns), y han sido llamadas de diferentes formas como temas (*Themes*) [1], casos de uso cruzados [2], hiperespacios (*Hiperspaces*) [3], filtros de composición [4], entre otros. La adecuada utilización de estas unidades de reuso, inclusive desde etapas tempranas del proceso de desarrollo [5], se constituye en el factor clave para la exitosa adopción de los aspectos en el proceso.

Cuando se sigue una aproximación orientada por aspectos, la separación de intereses no solo debe involucrar las características funcionales sino también las no funcionales, que son definidas por las consideraciones de calidad como seguridad, concurrencia, persistencia, sincronización, etc. Este conjunto de características son los primeros candidatos a ser intereses cruzados, que podrían ser implementados de forma aspectual.

Se entiende por portal temático un sitio web que presta un conjunto de servicios alrededor de la misma área de conocimiento, clasificando sus contenidos por temas. El portal que sirve como caso de aplicación de este trabajo se ocupa de la orientación por aspectos como un área específica de la ingeniería de software, el nombre de dicho portal temático es TerrACoTA, acrónimo de TERRitorio de Adopción de COnocimientos en Temas Aspectuales.

El objetivo de este artículo es ilustrar la experiencia de aplicación de aproximaciones orientadas por aspectos en el proceso de desarrollo de software, para tal propósito se organizó de la siguiente manera: Se presentan las aproximaciones que se encontraron oportunas para este trabajo en la sección 2; luego, en la sección 3, se muestra la forma en que se articularon estos enfoques para su adopción en el proceso de desarrollo; en la sección 4 se trabaja un caso práctico de aplicación en el desarrollo del portal; y finalmente se muestran las conclusiones.

Este trabajo está enmarcado en el proyecto MMEDUSA, un esfuerzo conjunto de la Universidad EAFIT y AVANSOFT S.A. con apoyo de COLCIENCIAS, el cual pretende brindar un **Marco Metodológico para Desarrollo de Aplicaciones Utilizando la Aproximación de Aspectos**.

II. MARCO DE REFERENCIA

Para la realización de este trabajo se revisaron diversos modelos que pretenden integrar aproximaciones orientadas por aspectos en diferentes flujos de trabajo del proceso de desarrollo de software, especialmente en las disciplinas de análisis de requisitos y de análisis y diseño. En al análisis de requisitos se estudiaron: La aproximación multidimensional de AORE [6], la aproximación orientada a objetivos de V-Graph [7], la propuesta de requisitos orientada a temas llamada Theme/Doc [8] y el esquema de clasificación de COSMOS [9]. Por otro lado en el análisis y diseño se exploraron: Theme/UML [10] y AOSD/UC [11].

En esta sección se presentan las propuestas de aplicación de aproximaciones orientadas por aspectos que se consideraron convenientes para el desarrollo del portal temático. La primera aproximación es una adaptación de un enfoque dirigido por casos de uso, presentada por el centro de informática de la Universidad de Pernambuco [2]; la segunda es el enfoque presentado en Theme [1].

Estas dos de aproximaciones fueron seleccionadas debido a que ambas propuestas inician desde la etapa de requisitos e intentan cubrir transversalmente el proceso de desarrollo.

El aporte de este trabajo de investigación, consiste en mostrar las heurísticas para integrar los postulados de las propuestas planteadas en [1] y [2] a lo largo del proceso de desarrollo unificado [13], ilustrando de forma concreta su aplicabilidad a través de un caso de aplicación en el que se enfatiza la trazabilidad entre los diferentes artefactos.

A. Adaptación de un Enfoque Dirigido por Casos de Uso

Esta propuesta se basa en los planteamientos del *Framework* NFR [12], y en los postulados del Proceso de Desarrollo de Software Unificado [13], que según sus principales características es "dirigido por casos de uso".

El enfoque dirigido por casos de uso fue también adaptado posteriormente, por su propio autor, para soportar aspectos [11], sin embargo la propuesta de adaptación del centro de informática de la Universidad de Pernambuco [2] se presenta concreta y es fácil de usar; esto unido a su fluida articulación con el tratamiento de las características no funcionales, constituyeron los criterios para optar por su utilización dentro de este trabajo.

1) Actividades de Requisitos: Las actividades propuestas para el flujo de trabajo de análisis de requisitos están clasificadas en 3 grupos:

− Las primeras actividades son provenientes del enfoque dirigido por casos de uso: desarrollar un modelo del dominio del problema, identificar actores y casos de uso y especificar casos de uso. Estas actividades siguen el enfoque clásico propuesto por el Proceso de Desarrollo de Software Unificado [13].

− El segundo grupo de actividades proviene del Framework NFR e incluye la descomposición de los intereses no funcionales para su operacionalización y refinamiento en unidades operacionales más finas y decisiones de diseño que sirven para alcanzar el interés no funcional. Las operaciones pueden ser especificadas con los casos de uso y las decisiones de diseño van en una sección especial del documento de requisitos. En el Framework NFR los intereses no funcionales son llamados softgoals, su operacionalización y las interdependencias entre ellos se representan gráficamente en un Grafo de Interdependencias de Softgoals.

− En el tercer grupo solo aparece la actividad de estructurar artefactos. Es en este punto en donde se introducen los casos de uso cruzados, que afectan los demás casos de uso con la relación de crosscuts; esta relación complementa las relaciones de include y extend. La heurística utilizada para determinar un caso de uso cruzado es la siguiente: el caso de uso cruzado es un curso que necesita ser aplicado en el caso de uso base, pero no persigue su principal objetivo, y adicionalmente el caso de uso cruzado puede ser aplicado a otros casos de uso.

2) Actividades de Análisis y Diseño: En cuanto a las actividades de este flujo de trabajo, se conservan las presentadas en el Proceso de Desarrollo de Software Unificado [13] como: encontrar clases de análisis, describir

la interacción entre los objetos de análisis, identificar y especificar entidades de diseño y definir relaciones entre las entidades de diseño. Cada una de estas actividades es modificada para incluir elementos propios de la orientación por aspectos, como las clases aspectuales que son estereotipadas con <<*aspect*>>.

Al igual que en las actividades de requisitos, en estas actividades se construyen tablas en las que se expresan los detalles de los elementos aspectuales. En estas tablas se destaca la columna del operador de la regla de composición, que puede tomar los valores *overlap.after, overlap.before, override y wrap* para denotar el tipo de *advice* a utilizar.

B. Theme

Theme se presenta como un enfoque y un conjunto de herramientas para identificar y manejar aspectos desde requerimientos hasta implementación: Un Theme o tema representa una característica del sistema [1].

El proceso en este enfoque consta de 4 fases: la primera encontrar temas, la segunda diseñar temas, la tercera componer temas y la cuarta verificar temas. Para encontrar y verificar temas en la fase de requisitos, se usa Theme/Doc, para diseñar y componer temas, en la fase de análisis y diseño, se usa Theme/UML.

1) Theme/Doc: se define como un conjunto de heurísticas para visualizar y analizar documentación de requisitos de software [8]. Está basada en la representación gráfica de potenciales temas llamadas *views*, en las que se muestra las relación entre los diferentes comportamientos y si éstos son posibles aspectos.

Las vistas son el resultado de la realización de algunas actividades. Las principales actividades en Theme/Doc con sus respectivas views son: identificar acciones con la *Action View,* clasificar acciones con la *Major Action View,* identificar temas cruzados con la *Clipped Action View,* revisar temas con la *Theme View* y verificar temas con la *Augmented View.*

2) Theme/UML: es un método de diseño y modelado que incluye una forma de representar aspectos en UML, propone una extensión del estándar del Lenguaje de Modelado Unificado para soportar la modularización presentada en la separación de intereses cruzados de las aproximaciones orientadas por aspectos [10]. Tiene las siguientes características:

– Reconoce dos tipos de solapamiento: el primero, los Concept Sharing cuando diferentes temas tratan un mismo concepto en el dominio; por ejemplo, los requisitos "Adicionar detalles del proyecto" y "Referenciar objetivo de un proyecto", probablemente ambos tendrán una noción de "proyecto". El otro tipo de solapamiento son los Aspect-oriented crosscutting, donde el comportamiento de un tema será activado en tanda por el comportamiento de otros temas.

– Propone tres etapas: la primera diseñar los temas separadamente, la segunda especificar relaciones entre los distintos temas, y la tercera componer los temas para propósitos de verificación.

– Involucra los Patrones de Composición (Composition Pattern): paquetes que contienen los modelos de diseño requeridos para especificar los comportamientos cruzados. Extienden la notación de plantillas de UML con dos elementos básicos: Parámetros de Plantilla (Template Parameters) para parametrizar el modelo del tema con elementos que serán reemplazados por los del modelo real, y Clases Patrón (Pattern Classes) que son clases colocadas en el paquete para cubrir el tema completo. Estos patrones de composición no deben referenciar explícitamente ningún elemento del diseño para posibilitar el comportamiento cruzado.

– Las relaciones de composición son de dos tipos: match para las relaciones concernientes a temas relativos a Conceptos Compartidos del Dominio (Concept Sharing) y bind para las relaciones de los temas cruzados (Aspect-oriented crosscutting); estas últimas pueden usar el comodín <{*},{*}> para referirse a todas las clases y todas las operaciones.

III. APLICACIÓN DE UN PROCESO DE DESARROLLO ORIENTADO POR ASPECTOS

El proceso de desarrollo utilizado en el caso de aplicación tiene las siguientes características:

– Utiliza los lineamientos del Proceso de Desarrollo de Software Unificado [13], para definir las etapas y disciplinas a seguir.

– Tiene como eje conductor los principios de aplicación de UML y patrones propuestos por Larman [14].

– Integra en los flujos de trabajo de análisis de requisitos y análisis y diseño, algunos de los planteamientos de las aproximaciones orientadas por aspectos presentadas en el marco de referencia de este artículo.

– La orientación por aspectos afectó en un alto grado las flujos de trabajo de análisis y diseño, y de ambiente; en grado medio el análisis de requisitos y la implementación; mientras los demás flujos de trabajo fueron afectados en bajo grado, conforme se esperaba [15].

Es importante aclarar que las diversas aproximaciones orientadas por aspectos muestran de forma concreta su aplicabilidad para requisitos no funcionales (como es el caso de la **seguridad** en el caso de aplicación), sin embargo su tratamiento en el frente funcional ha sido limitado, es este el motivo para profundizar en la aplicación de la orientación por aspectos en el desarrollo de los requisitos funcionales de corte transversal (como es el caso de "**asociar tema"** y las **búsquedas** en al caso de aplicación).

Para alcanzar las características del proceso de desarrollo plantado se construyeron artefactos cuyo nivel de utilidad para el portal temático fuera evidente, aprovechando la experiencia del grupo de desarrollo para hacer factible la construcción de dichos artefactos. La tabla 1 ilustra los artefactos acordados por el grupo de desarrollo para cada una de las etapas del proceso.

Tabla 1. Consideraciones para el uso de artefactos

Etapa	Artefacto	Ref.	Consideración de utilidad
Análisis de requisitos (Características funcionales)	Diagrama de subsistemas	[17]	Definición de la vista lógica del sistema e incorpora la modularización para abordar el desarrollo.

	Diagrama de casos de uso	[2]	Adaptación del enfoque dirigido por casos de uso que apoya la incorporación de las funcionalidades cruzadas (orientadas por aspectos).
Análisis de requisitos (Características no funcionales)	Grafo de interdependencias de softgoals	[2]	Tratamiento de los requisitos no funcionales para facilitar su trazabilidad en el proceso de desarrollo.
Análisis y diseño	Diagrama de diseño de temas	[10]	Consideraciones de diseño de las funcionalidades cruzadas a través de paquetes parametrizados en UML.
	Diagrama de composición de temas	[10]	Mecanismos para el reuso en las funcionalidades cruzadas, usando paquetes relacionados a través de la asociación <<bind>>.
Implementación	Diagrama de despliegue	[13]	Estrategia clásica para ilustrar la distribución de los elementos físicos en UML.

De esta forma se pretende sacar provecho de la experiencia de los desarrolladores y de las aproximaciones orientadas por aspectos estudiadas. Sin embargo, es importante tener en cuenta que la trazabilidad entre los elementos del diagrama de casos de uso propios de la adopción del enfoque dirigido por casos de uso y los temas cruzados propios de Theme/UML, se realizó de manera intuitiva, por ejemplo derivando un tema para el paquete de Búsquedas de la vista lógica.

Por razones de espacio este artículo no presenta la totalidad de los artefactos generados en el proceso de desarrollo del portal temático TerrACoTA, pero para ilustrar los diversos frentes de trabajo se muestran diagramas que trabajan diferentes funcionalidades en cada etapa del proceso.

A. Análisis de Requisitos

– Para el análisis de requisitos se utilizó la adaptación del enfoque dirigido por casos de uso [2], enfatizando la separación entre las características funcionales y no funcionales que se muestra en la propuesta. Los principales criterios para optar por esta propuesta en lugar de Theme/Doc, fueron los siguientes:

– La familiaridad con la técnica de casos de uso para el modelado de requisitos y lo sencillo que resulta aplicar las heurísticas para la determinación de los casos de uso cruzados.

– Al experimentar con Theme/Doc se obtuvieron una Action View y una Major Action View con un número considerable de nodos y de arcos, lo que hacía que el manejo de las gráficas fuera bastante engorroso.

1) Características Funcionales: En este frente se modelaron los requisitos funcionales del sistema, utilizando diagramas de casos de uso. Se involucraron las funcionalidades transversales usando casos de uso cruzados, los cuales se asociaban con los casos de uso base a través de la relación de <<*crosscuts*>>.

Los casos de uso se agruparon con base en los diferentes paquetes de la vista lógica del sistema. Cada paquete agrupa las funcionalidades referentes a cada uno de los servicios que presta el portal temático.

2) Características No Funcionales: Para estas características se tomaron los requisitos no funcionales más relevantes desde la perspectiva de los usuarios, para refinarlos y operacionalizarlos, y luego representar las relaciones entre sus operaciones y decisiones de diseño en un Grafo de Interdependencias de *Softgoals*.

Los *softgoals* asociados a operaciones, se involucraron en el modelado de los casos de uso de las características funcionales, para estructurar de esta forma el documento de requisitos definitivo.

B. Análisis y Diseño

Para el análisis y diseño se utilizó Theme/UML [10], enfatizando en el diseño de *Aspect-oriented crosscutting* presentados en la propuesta. Los principales criterios para optar en esta disciplina por esta propuesta en lugar de [2], fueron:

– La no referencia explícita a elementos del diseño por parte de los patrones de composición, facilita el paso a la implementación de las funcionalidades aspectuales.

– Las colaboraciones para el diseño de los temas se pueden realizar enfatizando las clases de negocio, sin tener que involucrar elementos de tipo boundary, entity o control, de esta forma la implementación no tiene que ser cercana a una arquitectura por capas, que por ejemplo implementa el patrón MVC[1] [16]. No ajustarse a una arquitectura específica, amplía la baraja de frameworks y librerías orientadas por aspectos que se pueden usar con PHP.

IV. CASO DE APLICACIÓN

El caso de aplicación de este trabajo investigativo es TerrACoTA, TERRitorio de Adopción de COnocimientos en Temas Aspectuales, un portal temático del Grupo de Investigación en Ingeniería de Software de la Universidad EAFIT, que presta un conjunto de servicios para el apoyo del aprendizaje y la difusión de la orientación por aspectos. Los servicios prestados por el portal se clasificaron en los siguientes módulos:

– **Portafolio de Proyectos:** gestiona información de los proyectos del grupo, tiene una opción para que los navegantes interesados en algún proyecto puedan solicitar su participación en éste.

– **Producción del Grupo:** gestiona el material producido por el grupo, como: informes de lectura, artículos, ensayos y reportes técnicos, con su historial de versiones.

1 Modelo-Vista-Controlador

– **Agenda del Grupo:** gestiona la programación de eventos relacionados con el grupo, teniendo la opción de hacer una preinscripción en aquellos que lo permitan.
– **Catálogo:** gestiona el material de referencia utilizado por el grupo, como: artículos, documentos y libros.
– **General:** gestiona información general del grupo y de otros grupos de interés, cuenta con una sección de actualidad donde se publican noticias, tiene una opción para que los navegantes tengan contacto con el grupo.

Estos cinco módulos unidos a los servicios de búsquedas y la gestión de usuarios, constituyen los paquetes de la vista lógica de la arquitectura de la aplicación, representada en el diagrama de subsistemas [17] de la figura 1.

Figura 1. Diagrama de subsistemas.

Las relaciones de los módulos de apoyo con los demás, se dan porque las Búsquedas acceden a la información de los demás módulos, mientras que el módulo de Gestión de Usuarios se encarga de construir el menú de acuerdo al rol de cada usuario e invoca servicios de los demás módulos para llamar los diferentes formularios del portal. Las relaciones de la Producción del Grupo con el Portafolio de Proyectos y con el Catálogo, se dan porque los informes de lectura, artículos, ensayos y reportes técnicos construidos por el grupo, pueden apoyar un objetivo de un proyecto o usar referencias del Catalogo.

A. Análisis de Requisitos

En lo concerniente a lo no funcional se analizó principalmente la seguridad, pues es uno de los factores críticos en un portal en Internet. Para las características funcionales, los paquetes agrupan los casos de uso de los diferentes servicios del portal.

1) Características No Funcionales: Utilizando los planteamientos del análisis de requerimientos orientado por objetivos [12], se construyó el Grafo de Interdependencias de *Softgoals* planteada en el *Framework* NFR [2].

Los *softgoals* resultantes de la operacionalización, se representaron con los bordes remarcados para su identificación en la figura 2. Se le dio prioridad al *softgoal* de confidencialidad (!), y se operacionalizó a través de la **Autenticación** y la **Autorización**. La autenticación se refiere a **Chequear Clave**, mientras que para la autorización se necesita **Cargar Menú**, a partir de **Consultar Rol** del usuario que se autenticó y **Seleccionar Opciones del Rol** a las que tiene acceso; de esta forma tenemos una alternativa de navegabilidad dinámica, basada en posibilitar la navegación de cada rol, solo a las páginas y formularios que sean permitidas al rol en el portal. Todos los *softgoals* de la operacionalización contribuyen de forma parcialmente positiva (+) a los *softgoals* a los que están relacionados, y fueron aceptados (✓).

Figura 2. Grafo de interdependencias de softgoals para la seguridad.

Para que la navegabilidad no sea restrictiva a la autenticación, se definieron un conjunto de páginas (opciones del menú) que pueden ser cargadas sin necesidad de estar autenticado, dentro de estas páginas se incluyen las búsquedas en cada uno de los módulos.

2) Características Funcionales: La estructura de la vista lógica de la arquitectura, nos plantea dos tipos de paquetes, los provenientes de los módulos ordinarios, que son: **Portafolio de Proyectos, Producción del Grupo, Agenda del Grupo, Catálogo** y **General**; y los módulos de apoyo que son: **Búsquedas** y **Gestión de Usuarios**.

Para el análisis de requisitos de los módulos ordinarios, se construyó un diagrama de casos de uso por cada módulo, usando casos de uso cruzados y la relación de <<*crosscuts*>>. Para ejemplificar la aplicación de aproximaciones orientadas por aspectos en el frente funcional, se tomó el módulo de la Agenda del Grupo, cuyo diagrama de casos de uso se muestra en la figura 3.

Figura 3. Diagrama de casos de uso del módulo de Agenda del Grupo.

Por limitantes de espacio no se ilustran todos los diagramas y tablas para el detalle de los elementos aspectuales; sin embargo, el caso de uso de **Asociar Tema** es una funcionalidad transversal que aparece en todos los módulos ordinarios, asociado con un caso de uso base a través de la relación de <<*crosscuts*>>.

En cuanto al análisis de requisitos de los módulos de apoyo, los *softgoals* de la operacionalización de la seguridad **Chequear Clave, Cargar Menú, Consultar Rol** y **Seleccionar Opciones del Rol**, fueron incluidos con los casos de uso provenientes de características no funcionales, dentro del módulo de Gestión de Usuarios.

B. Análisis y Diseño

En este flujo de trabajo se siguieron las diferentes etapas propuestas en Theme/UML [10], enfatizando en el tratamiento de los *Aspect-oriented crosscutting*; en cuanto a los temas producto de los *Concept Sharing*, se siguieron los planteamientos tradicionales para su análisis y diseño.

Teniendo en cuenta que las **Búsquedas** son transversales a los módulos de Portafolio de Proyectos, Producción del Grupo, Agenda del Grupo y Catálogo, se decidió trabajarlo como un tema *Aspect-oriented crosscutting*. En la figura 4 se ilustra la aplicación de la primera etapa de Theme/UML, correspondiente al diseño separado de los temas.

Figura 4. Diseño del tema de Búsqueda (Colaboración del patrón Strategy).

En la figura 4 se muestra la primera parte de la colaboración que realiza las búsquedas; se utiliza el patrón *strategy* [18] para el diseño de las diferentes formas de realizar las búsquedas: **Alfabética, Por Autor, Por Tema** y **Por Título**, para cada una de ellas se creó una subclase, que hereda de la clase abstracta **Busqueda**. El método **buscar( )** de la clase **Modulo** se encarga de invocar el método buscar apropiado de acuerdo al tipo de instancia que tenga el atributo **estrategia** de esta misma clase; este diseño se basa en el principio de delegación.

La composición que se da entre las clases abstractas de **Modulo** y **Contribucion**, sirve para implementar una lógica de contenedores que generaliza las relaciones de composición dadas entre las clases concretas: **Portafolio y Proyecto, Produccion y Producto, Agenda y Evento, y Catalogo y Material;** al hacer uso del principio de sustitución en estas jerarquías, se facilita en gran medida la implementación orientada por aspectos del patrón *strategy* para todos los módulos [19].

La figura 5 ilustra la segunda parte de la colaboración de las búsquedas, mostrando cuales son los mecanismos para la realización de las cuatro diferentes formas de buscar, separando en una vista de interacción [20**¡Error! No se encuentra el origen de la referencia.**] la colaboración de los métodos **buscar( )** de cada una de las clases concretas de las estrategias específicas.

Figura 5. Diseño del tema de Búsqueda (Colaboración de los diferentes métodos buscar).

En las cuatro vistas de interacción de la figura 5, se observan los métodos de las clases **Modulo, TemarioModulo y AutoriaModul***o*, que son intervenidos por los aspectos para implementar las búsquedas; este es el insumo para la definición de los *pointcuts*, los *joinpoints* y los *advices*.

Para que estas formas de búsqueda puedan ser invocadas desde cualquiera de los módulos ordinarios implicados, las colaboraciones de las búsquedas (figuras 4 y 5) no deben de referenciar de forma directa elementos del diseño; los paquetes en los que se especifican estas colaboraciones, son llamados patrones de composición, y se complementan en la esquina superior derecha con un cuadro de bordes punteados que contiene las *clases patrón*, con los métodos de las mismas que serán intervenidos.

Para el caso de las búsquedas las clases patrón son: **Modulo, TemarioModulo** y **AutoriaModulo;** mientras que las clases **Contribucion, Busqueda, Alfabética, PorAutor, PorTema** y **PorTítulo** son propias del patrón de composición.

Para la segunda etapa de Theme/UML, correspondiente a especificar relaciones entre los distintos temas, se ilustra la forma en que los temas implicados son cruzados por el tema de las búsquedas. En este punto es fundamental el uso de la relación de <<*bind*>> para definir las clases de cada tema que reemplazarán las clases patrón en la intervención del tema cruzado. La figura 6 muestra las relaciones de composición entre el tema de búsquedas, con los cuatro temas en los que se puede realizar una búsqueda.

Figura 6. Relaciones de composición entre temas.

C. Implementación y ambiente

En la fase de implementación se trabajó la vista de despliegue estática de la arquitectura [21], construyendo un diagrama de despliegue ilustrado en la figura 7.

Figura 7. Vista de despliegue estática de la aplicación.

En el diagrama de despliegue se distinguen tres elementos claves para el ambiente de desarrollo del proyecto:

– *Aspect-Oriented Programming PHP API (versión 2.0)*: una contribución de la comunidad *PHP Classes*, para la integración de la orientación a aspectos en la construcción de aplicaciones con PHP. Se seleccionó como alternativa para la implementación de las funcionalidades cruzadas de TerrACoTA.

– *framework Prado (versión 3.0.6)*: herramienta que apoya el desarrollo basado en componentes y la programación dirigida por eventos en PHP. Prado es el acrónimo de ***P**hp **R**apid **A**pplication **D**evelopment **O**bject-oriented*, y su liviana estrategia de desarrollo le hace acreedor de gran popularidad en el desarrollo Web en PHP.

– *ADODB (versión 4.53)*: librería para la implementación del acceso a datos en aplicaciones en PHP.

En este frente, el reto lo constituyó la integración de la orientación por aspectos en el *framework* Prado, pues a pesar de las diversas alternativas de *middleware* aspectuales [22], estas son escasas para plataformas basadas en PHP. Dicha integración de aspectos en Prado sirvió para implementar las funcionalidades cruzadas, como por ejemplo, la correspondiente al tema de Búsquedas. Sin embargo, la funcionalidad cruzada correspondiente al caso de uso de Asociar Temas, se implementó a través de un componente de

tipo *portlet*, que recibía como parámetros el módulo activo; dicha estrategia facilitó en gran medida la implementación de esta funcionalidad.

V. CONCLUSIONES

La integración de conceptos y estrategias presentados en las diferentes aproximaciones orientadas por aspectos [23], resulta de gran utilidad para abordar un proceso de desarrollo de software orientado por aspectos; extrayendo de cada aproximación los elementos que mas se ajusten a las condiciones específicas de un proyecto, y a las características de las personas que participan en este.

Para el caso específico de TerrACoTA, el equipo de trabajo encontró que las estrategias para el análisis de requisitos que suelen utilizar, tienen gran afinidad con la propuesta de adaptación de un enfoque dirigido por casos de uso, presentada por el centro de informática de la Universidad de Pernambuco [2]. Por otro lado, la estructuración por paquetes planteada en la vista lógica de la aplicación, facilita en gran medida el análisis y diseño con base en el enfoque presentado en Theme/UML [10].

En cuanto a la implementación, se logró hacer siguiendo fielmente los modelos planteados en el diseño. Sin embargo, se observó que en la implementación orientada por aspectos utilizando las librerías de PHP existentes, fue necesario agregar código en la funcionalidad base para indicar la intervención del aspecto, lo que significa que es una solución invasiva. Al igual que ha sucedido con plataformas Java y Microsoft [22], se espera que las alternativas en PHP, continúen su proceso de maduración como *middleware* orientada por aspectos evitando el código invasivo y facilitando la integración con algún IDE[2] para apoyar el proceso de desarrollo.

Los resultados de este trabajo investigativo se han cristalizado como un conjunto de servicios prestados por el sitio Web TerrACoTA, que se encuentra disponible para la comunidad aspectual en la dirección:
http://mmedusa.avansoft.com/terracota

REFERENCIAS

[1] Baniassad, E., Clarke, S.: Aspect-Oriented Analysis and Design: The Theme Approach. Addison-Wesley (2005)

[2] Sousa, G., Soares, S., Borda, P., Castro, J.: Separation of Crosscutting oncerns from Requirements to Design: Adapting an Use Case Driven Approach. Federal University of Pernambuco, Recife, Brazil (2004)

[3] Tarr, P., Ossher, H., Harrison, W., Sutton, S.: N-Degrees of Separation: Multi-Dimensional Separtion of Concerns. En: International Conference on Software Engineering, IEEE Computer Society Press (1999) 107–119

[4] Bergmans, L., Aksit, M.: Composing Crosscutting concerns using Composition Filters. En: Communications of the ACM. Vol. 44, No. 10 (2001) 51–57

[5] Rashid, A., Sawyer, P., Moreira, A., Araújo, J.: Early Aspects: a Model for Aspect-Oriented Requirements Engineering. En: IEEE Joint Conference on Requirements Engineering, Essen, Alemania (2002)

[6] A. Moreira, A. Rashid, and J. Araujo, "Multi-dimensional Separation of Concerns in Requirements Engineering. In International Conference on Requirements Engineering, 2005. IEEE Computer Society.

[7] Yijun Yu, J.C Sampaio, J. Mylopoulos, "From Goals to Aspects: Discovering Aspects from Requirements Goal Models", Proceedings of the Requirements Engineering Conference, 12th IEEE International (RE'04).

[8] Baniassad, E., Clarke, S.: Finding Aspects in Requirements with Theme/Doc. En Workshop on Early Aspects, Lancaster, UK (2004)

[9] S. Sutton and I. Rouvellou.: Modeling of Software Concerns in Cosmos. In Proceedings of the 1st International Conference on Aspect-Oriented Software Development (AOSD-2002), (2002) 127-133.

[10] Clarke S.: Aspect-Oriented Design with Theme/UML. En: Garcia-Molina, J., Moreira, A., Rossi, G. (eds.): Upgrade, European Journal for the Informatics Professional, Vol. V, No. 2 (2004) 14–20

[11] Jacobson, I., Ng, P.-W.: Aspect-Oriented Software Development with Use Cases. Addison-Wesley (2005)

[12] Mylopoulos, J., Chung, L., Liao, S., Wang, H., Yu, E.: Exploring Alternatives during Requirements Analysis. IEEE Software Ene/Feb (2001) 2–6

[13] Booch, G., Rumbaugh, J., Jacobson, I.: El Proceso Unificado de Desarrollo de Software. Addison-Wesley, Madrid, España (1999)

[14] Larman, C.: Uml y Patrones: Introducción al análisis y diseño orientado a objetos, 2 ed. Prentice Hall (2005) 627 p.

[15] Piveta, E., Devegili, A.: Aspects in the Rational Unified Process' Analysis and Design Workflow. AOSD, Centro Universitario Luterano de Palmas. Brasil (2002)

[16] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.: Pattern-Oriented Software Architecture: A System of Patterns. West Sussex, England: Wiley (1996)

[17] Durán, A., Bernárdez, B.: Metodología para la Elicitación de Requisitos de Sistemas Software: Versión 2.3. Departamento de Lenguajes y Sistemas Informáticos, Universidad de Sevilla, España (2002) 82 p.

[18] Gamma, E., Helm R., Johnson R., Vlissides J.: Design Patterns, Elements of Reusable Object-Oriented Software. Addison-Wesley (1995)

[19] Maiarú, L.: Switch Strategy Pattern. Universidad Argentina de la Empresa, UADE. Buenos Aires, Argentina (2003) 11p.

[20] Shekhar, C., Prabhakar, T.: An Overview of UML 2.0. En: Technical Meeting, Paris, Francia (2003)

[21] Kruchtem, P.: The 4+1 View Model of Architecture. En: IEEE Software. Vol. 12, No. 6 (1995)

[22] Loughran, N., Parlavantzas, N., Pinto, M., Fuentes, L., Sánchez, P., Webster, M., Colyer, A.: Survey of Aspect-oriented Middleware. AOSD-Europe (2005)

[23] Chitchyan, R., Rashid, A., Sawyer, P., Garcia, A., Pinto, M., Bakker, J., Tekinerdogan, B., Clarke, S., Jackson, A.: Survey of Analysis and Design Approaches. AOSD-Europe (2005)

Juan B. Quintero es Magíster en Ingeniería Informática de la Universidad EAFIT, actualmente es docente de cátedra de la Universidad de Antioquia y la Universidad EAFIT, y se desempeña como investigador en el Grupo de Investigación en Desarrollo de Software de la Universidad EAFIT y como consultor independiente. Sus principales áreas de trabajos son el desarrollo de software, la arquitectura dirigida por modelos y la ingeniería de modelos.

Diana M. Hernández C. es Ingeniera de Sistemas de la Universidad de Antioquia, actualmente es gerente general de ABC-Flex Ltda. una firma de consultoría y soluciones informáticas en Costeo ABC (Activity Based Costing). Sus principales áreas de trabajos son el desarrollo de software, la Web semántica y la gestión de proyectos de software.

Raquel Anaya de P. es Doctora en Ingeniería de la Programación e Inteligencia Artificial de la Universidad Politécnica de Valencia en España, actualmente se desempeña como Coordinadora del Área de Ingeniería de Software y jefe del Grupo de Investigación en Desarrollo de Software de la Universidad EAFIT. Sus principales áreas de trabajos son el proceso de desarrollo de software, las herramientas CASE y la arquitectura de de software.

2 Integrated Development Environment