Seventh LACCEI Latin American and Caribbean Conference for Engineering and Technology (LACCEI'2009)
"Energy and Technology for the Americas: Education, Innovation, Technology and Practice"
June 2-5, 2009, San Cristóbal, Venezuela.

Desarrollo de Líneas de Productos: un Caso de Estudio en Comercio Electrónico

Miguel A. Laguna
Universidad de Valladolid, Valladolid, España, mlaguna@infor.uva.es

RESUMEN

El desarrollo de una línea de productos software supone un reto considerable para las pequeñas organizaciones. Nuestra propuesta propugna el uso de herramientas convencionales para la mayoría de las fases de desarrollo. La trazabilidad entre los modelos de características y de diseño e implementación utiliza el mecanismo de combinación de paquetes de UML 2, de modo que la estructura de los modelos de características se refleja directamente en las relaciones entre paquetes del modelo de diseño. En el nivel de implementación la estrategia se basa en el mecanismo de clases parciales. El artículo reseña la aplicación de estas técnicas en el desarrollo de una línea de productos de comercio electrónico, elegida como caso de estudio representativo de sistemas Web. Los resultados indican que el uso de técnicas y herramientas habituales coloca este paradigma de desarrollo al alcance de la mayoría de los ingenieros de software actualmente en activo.

Keywords: línea de productos software, variabilidad, trazabilidad, comercio electrónico

ABSTRACT

Development of software product lines is a challenge for small organizations. We propose to use conventional tools for most of development phases. Traceability between the features and the UML architectural models is achieved by means of the package merge mechanism of the UML 2, as representation of the variability at design level. The structure of the feature models is directly reflected in the relationships between packages in the architectural models, so that the traceability of configuration decisions is straightforward. A similar strategy is applied at the implementation level, using packages of partial classes. This article reports a successful experience with a representative case study in the domain of web applications. The combination of these techniques and conventional IDE tools make the development of product lines accessible for most software engineers.

Keywords: software product line, variability, traceability, electronic commerce

1. INTRODUCCIÓN

El enfoque de desarrollo de software basado en líneas de producto (LP) ha alcanzado una gran aceptación en ciertos entornos industriales (Bosch, 2000). Sin embargo, el enfoque de líneas de productos es complejo y representa un gran salto para las empresas que pretenden abordarlo. Nuestro trabajo en el grupo de investigación GIRO tiene como primer objetivo simplificar el paso de un modelo de desarrollo convencional a otro que aproveche las ventajas de este paradigma. Para ello, entre otras iniciativas, propusimos una adaptación del Proceso Unificado de desarrollo para recoger las técnicas específicas de Ingeniería de Línea de Productos en un proceso paralelo al de Ingeniería de Aplicaciones (Laguna et al., 2003).

Entre las técnicas específicas de desarrollo de líneas de productos necesitamos, por un lado, definir los modelos que representen la parte común y las variaciones (y permitan configurar el conjunto óptimo de las mismas para

cada aplicación concreta) y, por otro, registrar la trazabilidad entre las variaciones del modelo y el reflejo de las mismas en la arquitectura y la implementación de la línea de productos (para facilitar la derivación de cada aplicación concreta). Existe un amplio consenso en torno al uso de modelos de features o características en alguna de sus múltiples versiones como FODA (Kang et al., 1990) o FORM (Kang et al., 1998) para expresar la variabilidad. Además nosotros hemos propuesto complementar los modelos de características con técnicas basadas en metas (goals y soft-goals) para el análisis de la variabilidad (González-Baixauli et al., 2004). Para ello se dispone de una herramienta que permite la selección del conjunto óptimo de características en función de los deseos de los usuarios, expresados como un conjunto de metas con distintas prioridades (González-Baixauli et al., 2004). Por otro lado, la trazabilidad entre el modelo de características y el diseño de la arquitectura de la línea de productos es la clave que permite automatizar la derivación de cada aplicación concreta. En trabajos previos (Laguna et al., 2007), propusimos el mecanismo de combinación de paquetes o package merge de UML 2 (Object Management Group, 2003) para representar de forma ortogonal las variaciones arquitectónicas de la línea de productos, su relación directa con las características opcionales e incluso, utilizando paquetes de clases parciales, con la estructura del código que implementa el diseño.

Este artículo describe de la aplicación práctica de estas técnicas al desarrollo de una línea de producto, elegida como caso de estudio representativo de sistemas Web, de modo que se cubren muchas de las situaciones habituales de desarrollo de sistemas de información. Una característica que distingue nuestro enfoque es el uso de herramientas convencionales en las fases de diseño (editores UML) e implementación (entornos de desarrollo o IDEs), algo que permite incorporar rápidamente a un equipo de desarrollo de líneas de producto profesionales neófitos en este campo. De hecho, las experiencias se han llevado a cabo con estudiantes recién graduados e incluso alumnos de proyecto fin de carrera, dirigidos por los autores, con buenos resultados de adaptación.

El resto del artículo se organiza como sigue: en la siguiente Sección, se repasan brevemente las técnicas específicas que utilizamos en el desarrollo de líneas de productos. La Sección 3 está dedicada a la descripción del caso de estudio abordado, los problemas surgidos y las soluciones adoptadas. La Sección 4 resume algunos trabajos relacionados y, finalmente, la Sección 5 concluye el trabajo y plantea el trabajo futuro.

2. COMBINACIÓN DE PAQUETES Y CLASES PARCIALES

Cada sistema concreto de una línea de productos se deriva de la arquitectura completa, tomando o no las partes opcionales adecuadas, según los requisitos funcionales y no funcionales seleccionados por los usuarios. Esta actividad es esencialmente un proceso de selección de características que genera un sub-modelo, que a su vez (por las relaciones de trazabilidad) compone por derivación toda o la mayor parte del código de la aplicación. La clave de este proceso reside en la trazabilidad desde las características hasta el código pasando por los modelos de diseño. Esta trazabilidad no es fácil de gestionar por varias razones. En primer lugar, una característica opcional puede originar varios elementos en un modelo de diseño (en general tenemos que asignar a la relación de trazabilidad entre elementos de distintos niveles una multiplicidad varios a varios). El segundo problema tiene que ver con el hecho de que los mecanismos básicos de modelado de la variabilidad (la especialización en los diagramas de clases o la relación <<extend>> de los casos de uso) se utilizan en muchas ocasiones para expresar dos tipos de variabilidad distinta: la existente en la arquitectura de la línea de productos (que se corresponde con requisitos opcionales) y la presente en una aplicación concreta, que sigue teniendo variaciones en tiempo de ejecución (por ejemplo, dos formas de pago alternativas).

Las soluciones habitualmente propuestas para mostrar la variabilidad de una línea de productos con UML pasan por modificar o anotar los modelos, tanto estructurales como funcionales o incluso dinámicos. En nuestro caso, una de las condiciones que impusimos fue mantener inalterado el meta-modelo de UML para permitir a los desarrolladores el uso de herramientas CASE convencionales. Las otras obligaciones entrañaban: a) localizar en un solo punto del modelo todas las variaciones que origina cada característica opcional, de forma que se facilite la trazabilidad; b) separar la variabilidad de la línea de productos de la variabilidad intrínseca de las aplicaciones concretas, eliminando imprecisiones; c) conectar los modelos de análisis, diseño e implementación para acercarnos al ideal de desarrollo sin costuras.

Para alcanzar estos objetivos, se puede expresar la variabilidad en los modelos UML utilizando el concepto de combinación de paquetes (o "package merge"), presente en el meta-modelo de infraestructura de UML 2 y utilizado de forma exhaustiva en la definición misma de UML 2. El mecanismo "package merge" consiste fundamentalmente en añadir detalles de forma incremental y se define como una relación entre dos paquetes que indica que los contenidos de ambos se combinan. Aunque la utilización en la documentación de UML se limita a diagramas de clases, cualquier meta-tipo del meta-modelo de UML se puede utilizar con esta técnica.

Este mecanismo nos permite establecer una trazabilidad clara entre los modelos de características y los artefactos UML. La aplicación a líneas de productos consiste en establecer en el modelo arquitectónico un paquete raíz que recoge la parte común de la línea de productos (incluyendo modelos estructurales, de casos de uso y de interacción). Junto a este paquete común se añade un paquete por cada característica opcional, de modo que queden localizadas en ese paquete todas las modificaciones necesarias en el modelo de diseño asociadas a esa característica. El paquete añadido se conecta mediante la relación <<merge>> con su paquete base en el punto preciso de la jerarquía de paquetes. La aplicación de la técnica se ilustra con el ejemplo trivial de la Figura 1, donde una característica opcional (ShippingOptions) implica la existencia del correspondiente paquete en el modelo de diseño. Por otro lado, mediante el uso de clases parciales organizadas en paquetes, se puede establecer una correspondencia directa entre los modelos de diseño y la implementación de la línea de productos (Laguna et al., 2007).

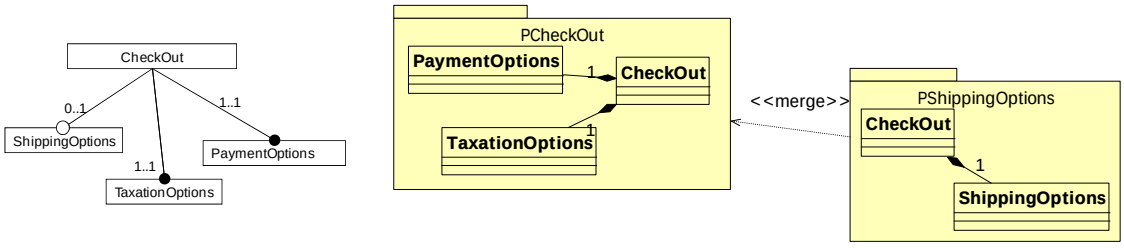

Figura 1. Modelo de características y su correspondiente modelo de diseño UML

3. CASO DE ESTUDIO: COMERCIO ELECTRÓNICO

El caso de estudio que planteamos se ha llevado a cabo utilizando como punto de partida un análisis del dominio de comercio electrónico, descrito en (Lau, 2006). En él se realiza un completo estudio del mismo, utilizando distintos modelos, pero no se llega a implementar la línea de productos, lo que nos ha proporcionado un punto de partida muy interesante para contrastar la técnica puesto que los paquetes que hemos de desarrollar dependen de un estudio ajeno.

El objetivo es mostrar que es posible desarrollar una línea de productos viable desde el punto de vista industrial, generando un conjunto amplio de prototipos de portales de ventas. Hasta el momento se ha desarrollado la parte común de la línea de productos para disponer de un sistema de comercio electrónico con una funcionalidad mínima, pero que permita un proceso de compra completo. Además, se han completado una docena de paquetes opcionales que forman una familia de cientos de productos potenciales, aunque la variabilidad irá creciendo con los sucesivos paquetes que están actualmente en desarrollo. Éste es por lo tanto el primer paso de lo que finalmente se convertirá en una familia de productos de complejidad significativa en el ámbito del comercio electrónico.

MODELO DE CARACTERÍSTICAS

El modelo utilizado como punto de partida es el original de (Lau, 2006), abordando el desarrollo por fases, dada la magnitud del mismo. En dicho modelo, se detallan las distintas características que un producto final de comercio electrónico puede tener, indicando claramente las que son

- características obligatorias: el carrito de la compra, una estructura de catálogos y productos con información básica, etc.

- características opcionales que se podrán ir añadiendo o no para aumentar las funciones de un producto, como búsqueda en catálogos
- características agrupadas en alternativas (o grupos OR): se debe elegir una opción (o más de una) entre varias. Por ejemplo, el proceso del pedido puede llevarse a cabo como usuario registrado o no, según el tipo de tienda. Y el tipo de producto vendido puede ser electrónico, físico o ambos

El número de características contempladas es de varios centenares. Dado que la representación gráfica original de los modelos de características es escasamente manejable para casos con ese número de características (e inútil para configurar los productos finales), en la práctica se utilizan herramientas como el plug-in fmp, desarrollado en la Universidad de Waterloo por Czarnecki y sus colaboradores (Antkiewicz & Czarnecki, 2004). La Figura 2 muestra parte del modelo de características original, junto con una posible configuración: se han elegido las características que representan un pedido de un usuario invitado, categorías, categorías multinivel, etc.

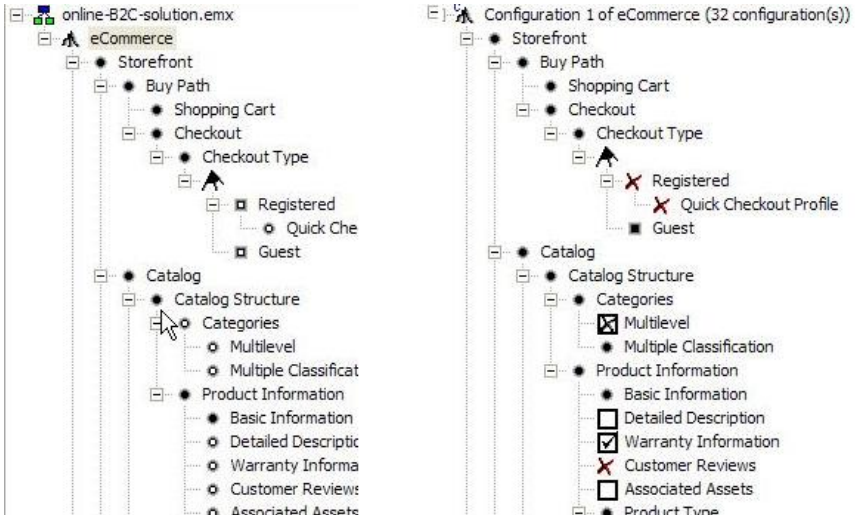

Figura 2. Modelo de características parcial de la LP y una posible configuración de la misma

ANÁLISIS Y DISEÑO DE LA LÍNEA DE PRODUCTOS

Inicialmente hemos desarrollado las características obligatorias imprescindibles en toda aplicación de comercio electrónico y algunas de las opcionales que, no siendo imprescindibles, sí son deseables para disponer de un juego mínimo de variantes. El punto de partida del diseño ha respetado las ideas básicas de la arquitectura propuesta por Lau pero reorganizándola en paquetes. Ello nos ha permitido centrar nuestros esfuerzos en resolver desde el punto de vista práctico algunos problemas secundarios: a) encontrar soluciones para manejar la variabilidad en las capas de acceso a datos y de interfaz de usuario en un entorno Web; b) diseñar el mecanismo adecuado para la configuración de los productos finales de la línea de productos.

El modelo básico de la línea de productos se muestra en la Figura 3. Los paquetes de la parte inferior son opcionales e independientes entre sí. Sin embargo los dos paquetes de la parte superior de la figura están relacionados con una estructura de características tipo OR(1..2), es decir que se debe elegir al menos un tipo de producto (de los dos posibles). Aunque esto último no se refleja en el modelo de paquetes, en realidad la configuración se lleva a cabo mediante el modelo de características y en él sí se controla de forma precisa.

Para poder disponer de una plataforma de comercio electrónico realmente útil necesitamos también al menos una forma de pago. Se ha implementado PayPal como pasarela de pago en el paquete Base por su simplicidad, dado que no es necesario el registro previo del cliente, aunque se han desarrollado otras posibilidades de pago (como tarjeta de crédito y servidor seguro). Además del paquete Base, se han desarrollado una decena de paquetes, tal como se muestra en la Figura 3. Algunos de esos paquetes son los siguientes: CategoryMultilevel, Search,

DirectDownload, ElectronicProduct y PhysicalProduct. Están también desarrollados paquetes que permiten disponer de facilidades de clientes registrados o el pago con tarjeta de crédito y facturación.

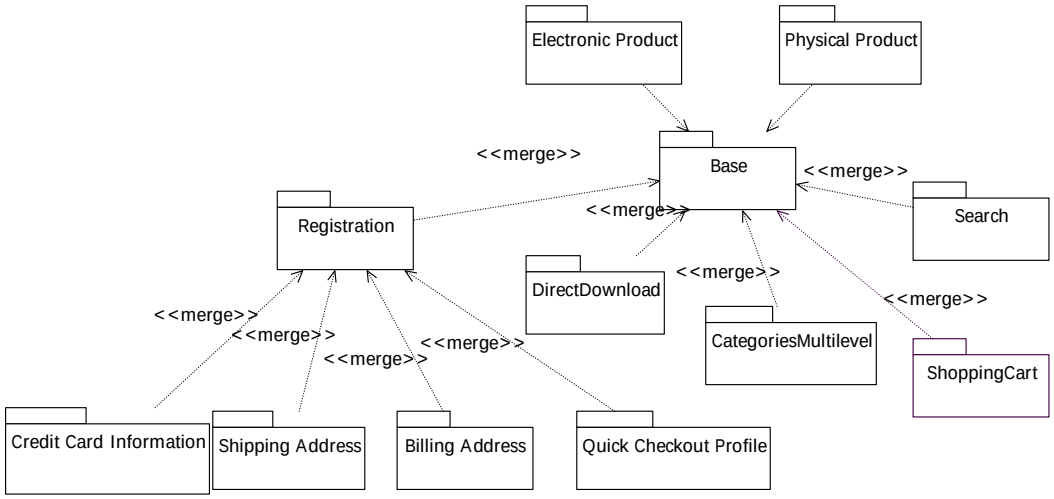

Figura 3. Paquetes de la línea de productos de comercio electrónico

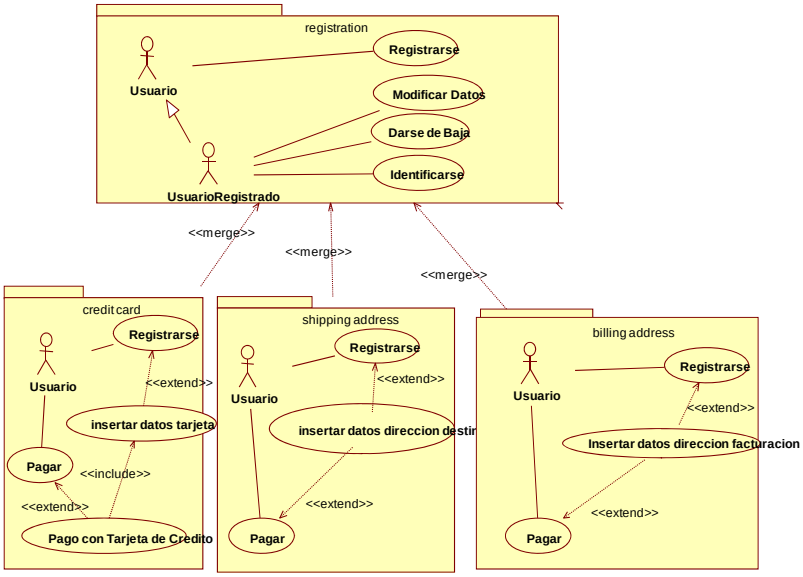

Figura 4. Contenido de algunos paquetes de la LP de comercio electrónico (vista de casos de uso)

Este esquema de combinación de paquetes se utiliza durante todas las fases de desarrollo de la línea de productos, incluyendo los modelos de análisis (casos de uso y diagramas de clases), diseño (diagramas de clases e interacción) e implementación (paquetes de clases clases parciales junto con ficheros html o XML). En el caso de la Figura 4 se muestra parcialmente el modelo de casos de uso. Si se selecciona el paquete ShippingAddress, se incluye en el modelo un caso de uso adicional como extensión que permite introducir la dirección de envío en el momento de registrarse como usuario y en el momento de cerrar una compra. En la Figura 5 se aprecia la estructura interna de algunos de los paquetes de clases de diseño que reflejan el modelo de características. Un catálogo, obligatorio en la línea de productos, tiene una estructura de categorías básica. Añadir, por ejemplo, la característica opcional CategoriesMultilevel se introduce una asociación reflexiva en la clase Category tal y como se muestra en el paquete correspondiente. Al aplicar el mecanismo package merge, el resultado de combinar estos paquetes es que aparece una clase final Category con los atributos y métodos de cada paquete y tres asociaciones con las clases Product, Catalog y con ella misma.

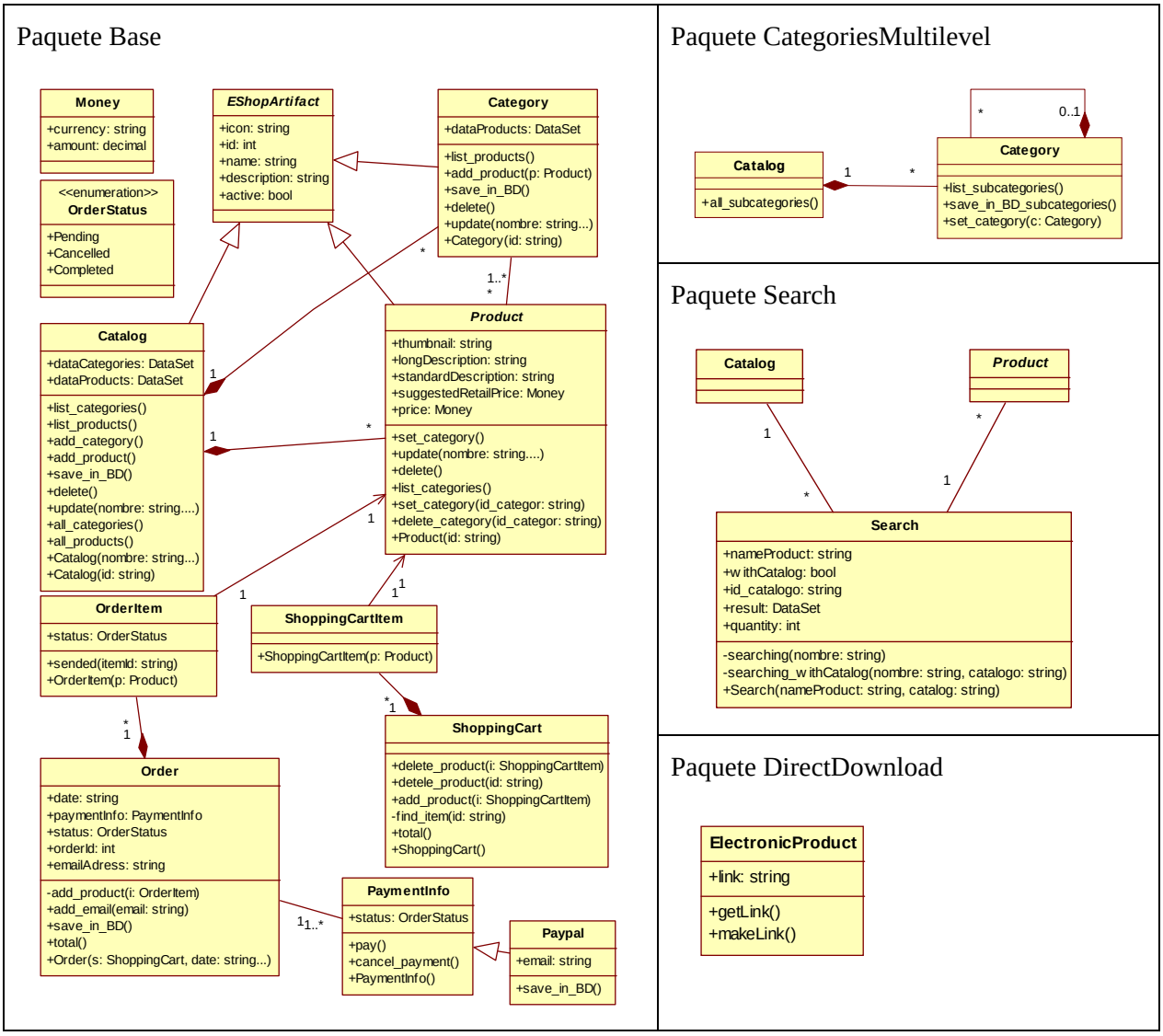

Figura 5. Paquete base de la LP de comercio electrónico, junto con tres paquetes opcionales

IMPLEMENTACIÓN DE LA LÍNEA DE PRODUCTOS

Para extender la trazabilidad hasta el nivel de implementación se utiliza el concepto de clase parcial. Aunque el nombre de las clases sea el mismo, su pertenencia a distintos paquetes las hace entidades diferentes. Si los paquetes a los que pertenecen son seleccionados, en el momento de compilación del sistema se combinan en una única clase, haciendo que este modelo reproduzca en la implementación de la línea de productos la misma estrategia utilizada en requisitos y diseño. Por tanto, para derivar una aplicación concreta basta con indicar al compilador los paquetes necesarios, que se corresponden con la configuración elegida en el modelo de características. De esta manera se cubre el objetivo de la trazabilidad uno-a-uno desde las características hasta el código. En las implementaciones llevadas a cabo hasta el momento, se ha utilizado C# como lenguaje de programación, dentro de la estructura de soluciones y proyectos de la plataforma Microsoft .NET.

Además de las clases del dominio, la implementación debe solucionar también los problemas relacionados con la interfaz de usuario y la persistencia. En el caso de la persistencia, la solución pragmática adoptada es generar un esquema de base de datos que contenga todas las tablas necesarias considerando que se utilizan todas las opciones. Sin embargo, en el detalle de los métodos que manejan la comunicación con la base de datos se utilizan

estructuras de datos a veces incompletas, de modo que muchas columnas de las tablas pueden quedar sin uso. Entre los trabajos pendientes está el establecer un mecanismo de generación automática del esquema de la base de datos a medida de cada sistema concreto.

En cuanto a la interfaz de usuario, hemos utilizado una combinación de plantillas, ficheros de estilo y contenedores dinámicos. Con ASP.NET es posible crear páginas de tipo master.page, que sirven como plantillas del sistema Web, combinadas con ficheros de estilo en cascada. Puesto que no se trata de una aplicación Web cerrada, cada producto final de la línea tendrá posiblemente una página principal distinta a ojos del usuario final. El mecanismo de variabilidad de la plantilla utiliza contenedores dinámicos (ContentPlaceHolder), que serán rellenados de forma dinámica mediante el código de cada paquete concreto. Como se puede ver en la Figura 6, cada plantilla está formada por varias partes. Las flexibles son la parte izquierda, donde se encontrarán los distintos menús y la parte central, que corresponde al contenido de la página propiamente dicho. Tanto en un caso como en otro, se ha previsto un contenedor editable de forma individual.

El siguiente paso consiste en que el sistema reconozca los paquetes instalados y los controles correspondientes a cada uno de ellos, agregándolos de forma automática. Para ello se utilizan archivos basados en XML que indican paquetes, rutas, etc. Se aprovechan los archivos de configuración que proporciona la plataforma .NET y se han añadido otros específicos en cada paquete. Se utiliza un convenio de nombres para evitar conflictos y una llamada sistemática en cada página a un método que sirve para construirla de forma personalizada.

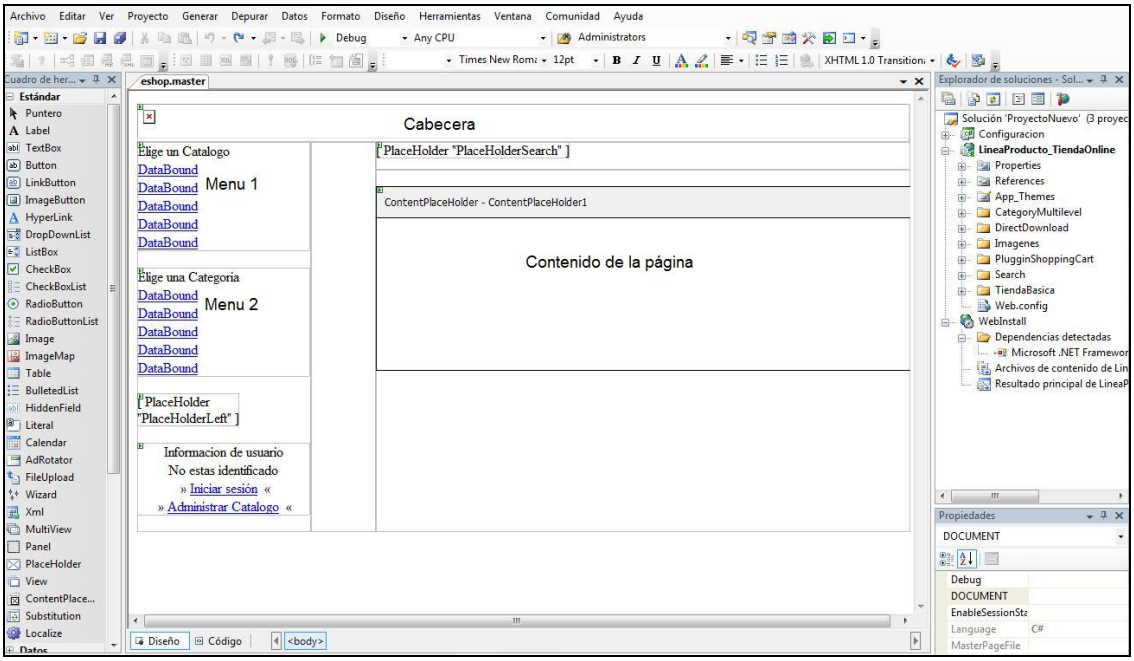

Figura 6. Vista de diseño de la plantilla con dos contenedores dinámicos

En la figura 7 se observa la forma de configurar los paquetes y el la figura 8 se pueden ver dos ejemplos de productos finales con distinto grado de complejidad. En resumen, se ha desarrollado hasta la última fase una línea de productos software en una plataforma Web, que incluye un proceso de compra totalmente real y seguro, con distintas variaciones. Al mismo tiempo, se han definido las técnicas de implementación necesarias para manejar la variabilidad a nivel de código. A día de hoy se dispone una decena de paquetes opcionales (no totalmente independientes) de modo que se pueden compilar cientos de productos totalmente funcionales, a partir de las distintas combinaciones legales de paquetes. Los detalles de cómo instalar y configurar un producto se pueden consultar en (García de Acilu & Hernández Herrera, 2008). El resultado práctico es que, una vez implementados los paquetes de clases parciales (incluyendo la interfaz de usuario y la capa de persistencia) en una herramienta IDE convencional, es posible generar, compilar y desplegar un producto concreto de comercio electrónico en pocos minutos.

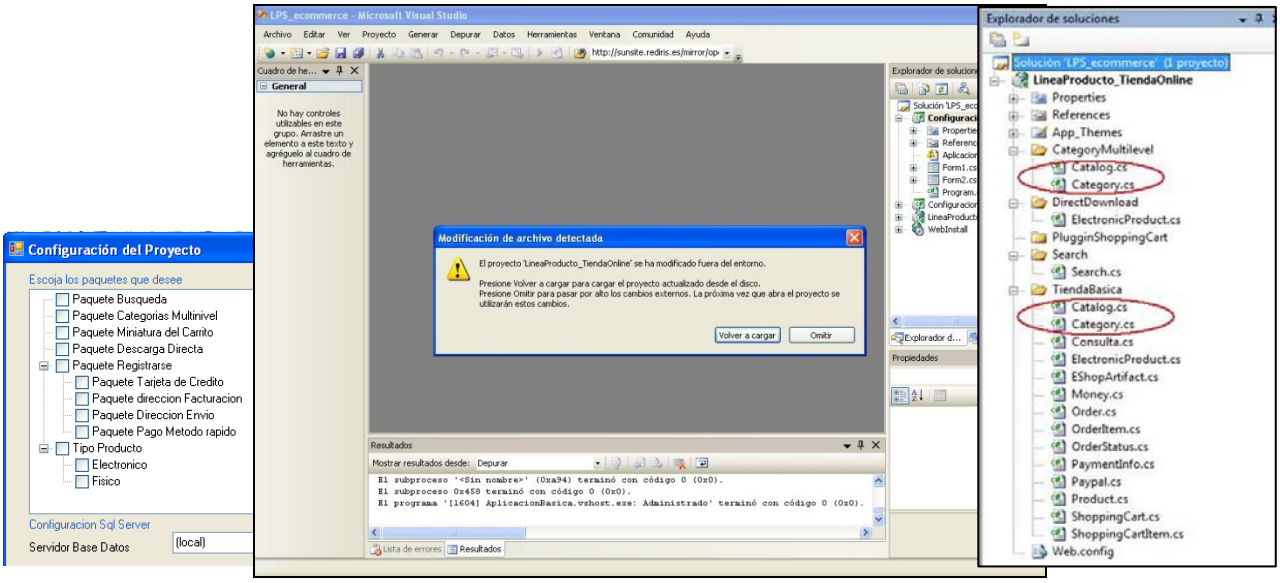

Figura 7. Proceso de configuración de la LP de comercio electrónico y detalle de los paquetes

Estos primeros pasos constituyen la base de una línea de productos de comercio electrónico que hemos utilizado como demostración del concepto por su gran cantidad de variaciones potenciales y que en el futuro desarrollará el modelo de características de referencia al completo.

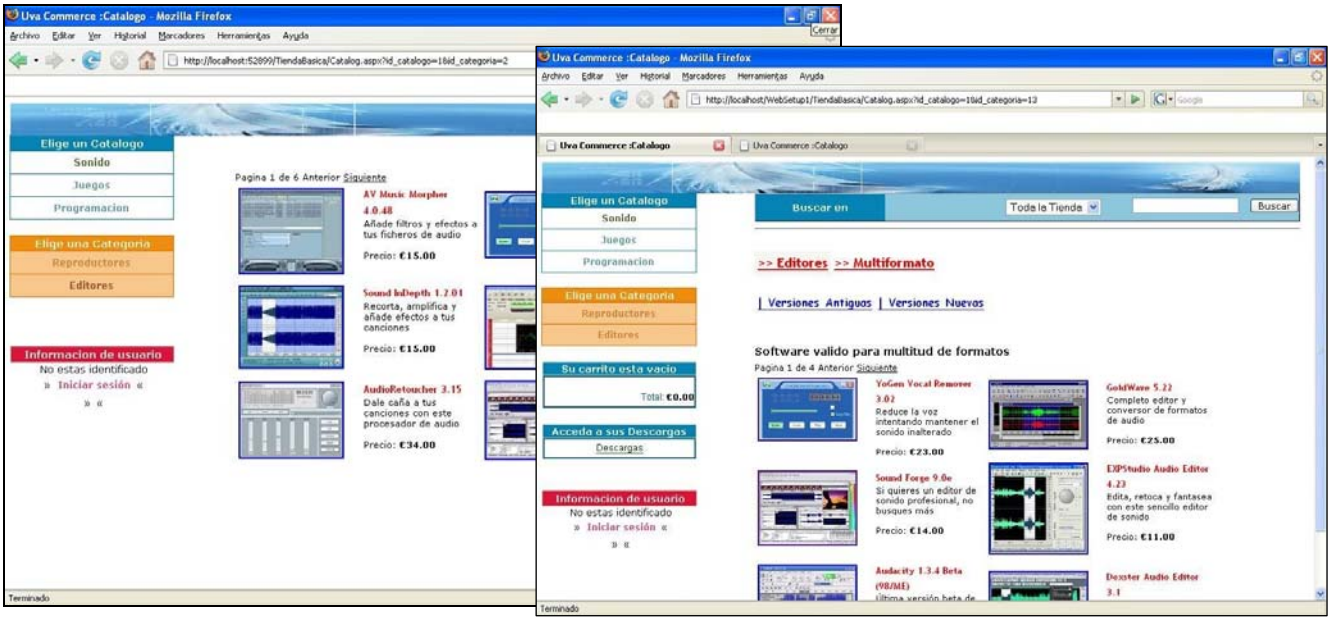

Figura 8. Vistas en el navegador de dos productos distintos de la LP de comercio electrónico

4. TRABAJOS RELACIONADOS

Aunque hay muchos trabajos que describen mecanismos de gestión de la variabilidad desde el punto de vista de los requisitos y el diseño, pocos llegan hasta los detalles de implementación. Distintos autores han propuesto representar explícitamente los puntos de variación añadiendo anotaciones o cambiando la esencia de los diagramas de UML. Por ejemplo, Von der Maßen (Massen & Lichter, 2003) propone utilizar una notación gráfica con nuevas relaciones ("option" y "alternative"), con la consiguiente extensión del meta-modelo de UML. (John

& Muthig, 2002) sugieren la aplicación de plantillas de casos de uso, utilizando estereotipos, aunque no distinguen entre variantes opcionales, alternativas u obligatorias. En cambio en (Halmans & Pohl, 2003) se defiende la modificación de los modelos de casos de uso para representar de forma gráfica los puntos de variación. En cuanto a los modelos estructurales, o bien se utilizan directamente los mecanismos de UML (mediante la relación de especialización, la multiplicidad de las asociaciones, etc.), o bien se anotan de forma explícita mediante estereotipos. El trabajo de (Gomaa, 2000) es un ejemplo de este último enfoque, ya que utiliza los estereotipos <<kernel>>, <<optional>> y <<variant>> (que corresponden a clases obligatorias, opcionales, y variantes). Propone un método de desarrollo completo hasta la implementación. De un modo similar en (Clauß, 2001) se propone un conjunto de estereotipos para expresar la variabilidad. Aunque este tipo de aproximaciones permiten rastrear la evolución de la variabilidad en los distintos niveles, no resuelven el requisito de correspondencia uno-a-uno entre los distintos modelos.

Otra solución, propuesta en (Czarnecki & Antkiewicz, 2005) consiste en anotar los diagramas de UML con condiciones de presencia expresados mediante códigos de colores, de modo que cada característica opcional se refleja en uno o varios elementos de un único diagrama (el resultado es una superposición de varios diagramas en uno). Esta técnica no limita de forma artificial la representación de una variante a un único elemento e incluso el código de colores ayuda a destacar de un solo vistazo las implicaciones de elegir una cierta opción. Sin embargo esta ayuda visual es muy poco escalable. Aunque la solución es válida, el aprendizaje de nuevas técnicas de modelado o la necesidad de herramientas CASE ad hoc representan barreras para la adopción del enfoque de líneas de productos en muchas organizaciones, por lo que creemos que la solución presentada mejora las propuestas analizadas.

Un enfoque diferente, Feature Oriented Programming (FOP) aborda el problema desde el punto de vista de la implementación (Batory et al., 2004). Las características se definen como incrementos de código (refinamientos), a partir de una clase base. El resultado final es similar al nuestro aunque carecen de un modelo de alto nivel que permita seleccionar las características opcionales sin bajar al detalle de implementación. Además, las herramientas de programación de FOP (o herramientas comerciales como Pure-Variants) representan un cambio notable que debe ser asumido por los desarrolladores que se incorporaran a este tipo de desarrollos. La utilización de IDEs convencionales, tal y como proponemos nosotros, resulta una ventaja clara en el proceso de adopción del paradigma.

5. CONCLUSIONES Y TRABAJO FUTURO

En este trabajo se ha mostrado la viabilidad de los mecanismos de combinación de paquetes y clases parciales para organizar la variabilidad de una línea de productos utilizando únicamente modelos y lenguajes de programación estándar. Esta posibilidad simplifica la adopción de este paradigma de desarrollo, evitando la necesidad de herramientas o técnicas específicas como en los trabajos de otros autores. Además, la trazabilidad entre niveles permite derivar la implementación de cada aplicación concreta a partir de la configuración de características deseada.

La técnica se ha aplicado al diseño e implementación de un caso de estudio inicial en el dominio de las aplicaciones Web, basado en un análisis previo sobre comercio electrónico publicado en la literatura. De este modo se ha evitado cualquier posible sesgo en el modelado de las características. En este primer caso se han conseguido alcanzar los objetivos propuestos: desarrollar hasta el final la línea de productos y definir las técnicas necesarias para manejar la variabilidad en el nivel de implementación.

Como trabajos en fase de realización, se están desarrollando otras líneas de productos de interés industrial. En particular, se está construyendo una segunda línea de productos basada en la Web en el dominio de asociaciones sin ánimo de lucro (culturales, deportivas, etc.), utilizando la misma filosofía básica de diseño y con similares resultados. Otras líneas de productos abordan aplicaciones sobre dispositivos móviles, que presentan problemas distintos debidos a las limitaciones de la plataforma. Se pueden ver los avances de cada una de las líneas de productos en las páginas del grupo GIRO (http://giro.infor.uva.es). En paralelo se está ampliando el primer caso de estudio, en este caso para evaluar la escalabilidad de la propuesta a medida que las características opcionales aumentan, lo que implica un aumento exponencial del número de productos. Finalmente, está pendiente de

integración en la plataforma Visual Studio una herramienta de modelado y configuración de características (desarrollada con las herramientas DSL de Microsoft). Una vez integrada, se podrá elegir como uno más de los proyectos predefinido el tipo "línea de producto" que incluirá el modelo de características como un componente más del proyecto.

AGRADECIMIENTOS

Este artículo ha sido financiado por la Junta de Castilla y León (proyecto VA-018A07).

REFERENCIAS

Antkiewicz, M., & Czarnecki, K. (2004). Feature modeling plugin for Eclipse. OOPSLA'04 Eclipse technology exchange workshop.

Batory, D., Sarvela, J., & Rauschmayer, A. (2004). Scaling Step-Wise Refinement. IEEE TSE .

Bosch, J. (2000). Design & Use of Software Architectures. Adopting and Evolving a Product-Line Approach. Addison-Wesley.

Clauß, M. (2001). Generic modeling using Uml extensions for variability. Workshop on Domain Specific Visual Languages at OOPSLA.

Czarnecki, K., & Antkiewicz, M. (2005). Mapping Features to models: a template approach based on superimposed variants. In proc. of GPCE'05, LNCS 3676, (págs. 422-437). Springer.

García de Acilu, M., & Hernández Herrera, E. (2008). Desarrollo y Configuración de una Línea de Producto Software de comercio electrónico, PFC sep. 2008. Disponible en http://giro.infor.uva.es.

Gomaa, H. (2000). Object Oriented Analysis and Modeling for Families of Systems with UML. IEEE International Conference for Software Reuse (ICSR6), (págs. 89–99).

González-Baixauli, B., Laguna, M. A., & Leite, J. C. (2004). Análisis de Variabilidad con Modelos de Objetivos. Anais do WER04, (págs. 77-87).

González-Baixauli, B., Leite, J., & Mylopoulos, J. (2004). Visual Variability Analysis with Goal Models. Proc. of the RE'2004, (págs. 198-207).

Halmans, G., & Pohl, K. (2003). Communicating the Variability of a Software-Product Family to Customers. Journal of Software and Systems Modeling , 15--36.

John, I., & Muthig, D. (2002). Tailoring Use Cases for product line Modeling. Proceedings of the International Workshop on Requirements Engineering for product lines 2002 (REPL'02). Technical Report: ALR-2002-033, AVAYA labs.

Kang, K. C., Kim, S., Lee, J., & Kim, K. (1998). FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures. Annals of Software Engineering , 143-168.

Kang, K., Cohen, S., Hess, J., Nowak, W., & Peterson, S. (1990). Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report, CMU/SEI-90-TR-21, Software Engineering Institute (Carnegie Mellon), Pittsburgh, PA 15213.

Laguna, M. A., González, B., López, O., & García, F. J. (2003). Introducing Systematic Reuse in Mainstream Software Process. IEEE Proceedings of EUROMICRO'2003, (págs. 351-358).

Laguna, M. A., González-Baixauli, B., & Corral, J. M. (2007). Seamless Development of Software Product Lines: Feature Models to UML Traceability. GPCE 07.

Lau, S. (2006). Domain Analysis of E-Commerce Systems Using Feature-Based Model Templates", . MASc Thesis, ECE Department, University of Waterloo, Canada.

Massen, T. v., & Lichter, H. (2003). RequiLine: A Requirements Engineering Tool for Software product lines. Software Product-Family Engineering, PFE, LNCS 3014 pp , (págs. 168-180).

Object Management Group. (2003). Unified modeling language specification version 2.0: Infrastructure. Technical Report ptc/03-09-15. OMG.

Authorization and Disclaimer

Authors authorize LACCEI to publish the paper in the conference proceedings. Neither LACCEI nor the editors are responsible either for the content or for the implications of what is expressed in the paper.