Documentando Arquitecturas Orientadas a Aspectos para Líneas de Productos de Software

Ocharán Hernández, Jorge Octavio[1], Cortes Verdin, Karen[1,2]

[1]Facultad de Estadística e Informática
Universidad Veracruzana
[2]Centro de Investigación en Matemáticas (CIMAT, A.C.)
jocharan@uv.mx, karen@cimat.mx
Paper received on 30/07/08, accepted on 10/09/08.

Abstract. Una línea de productos de software (LPS) es una forma de crear diferentes productos de software o aplicaciones al reutilizar un conjunto de artefactos centrales de una manera preestablecida. Actualmente las líneas de productos de software empiezan a introducir la Orientación a Aspectos en el desarrollo de nuevos productos de software. El desarrollo orientado a aspectos busca la modularización de intereses (funcionales y no funcionales) dispersos y entrelazados en diferentes partes de un sistema. La adopción de la orientación aspectos en una línea de productos de software mejorará substancialmente la modularidad obteniendo una estructura más comprensible, de fácil comprensión y evolución. Al ser una práctica nueva, se carece de una estrategia para documentar Arquitecturas Orientadas a Aspectos de Líneas de Productos de Software (AOALPS) trayendo consigo innumerables problemas en la administración y explotación de la LPS. En este artículo se presenta propuesta de una propuesta para documentar el artefacto más importante en una LPS.

1 Introducción

Las Líneas de Productos de Software (LPS) surgen de la necesidad de obtener sistemas de software complejos, de calidad, costo-efectivos y que se encuentren en el mercado en el momento adecuado. Clements y Northrop las definen como un "conjunto de sistemas de software intesivo que comparten una serie de características administradas que satisfacen las necesidades de un segmento de mercado o misión y que se desarrollan utilizando un conjunto de artefactos centrales de una manera preestablecida" [1] como artefactos activos esenciales (*core assets*) debemos atender aquellos artefactos y recursos reutilizables que forman la base de la LPS. A partir de los activos esenciales se pueden derivan diferentes productos de software que responderán a requerimientos variables.

Algunos de los beneficios de las LPS son:

- Aumento de la productividad a gran escala
- Reducción del tiempo entre que se toma la decisión de desarrollar un producto y la fecha del lanzamiento comercial (*time to market*).
- Incremento en la calidad de los productos de software

© M. G. Medina Barrera, J. F. Ramirez Cruz, J.H. Sossa Azuela, (Eds.).
Advances in Intelligent and Information Technologies
Research in Computing Science 38, 2008, pp.333-342.

- Reducción de los riesgos
- Incremento en la satisfacción del cliente
- Una más eficiente de los recursos humanos
- Permite la personalización en masa
- Permite mantener presencia en el mercado
- Crecimiento de la organización.

La Ingeniería de Líneas de Productos de Software (ILPS) se encarga de desarrollar aplicaciones de software utilizando plataformas y personalización en masa [2]. La ILPS separa el desarrollo de software en dos procesos: Ingeniería de Dominio (ID) y la Ingeniería de Aplicación (IA). La ID define la variabilidad y similitudes de la LPS estableciendo la plataforma (artefactos de software de requerimientos, diseño, arquitectura, pruebas, etc.). La IA se encarga de derivar los productos de la línea a partir de la plataforma establecida en la ID.

Uno de los artefactos principales de la ID es la arquitectura de dominio o arquitectura de referencia. Ésta determina la estructura y textura de los productos en la LPS. La estructura determina de descomposición estática y dinámica que es válida para todas las aplicaciones de la LPS. La textura es el conjunto de reglas que guían el diseño y realización de las partes y como se combinan para formar aplicaciones.

La arquitectura de software permite especificar la estructura de un sistema en términos de componentes y conectores. Bass *et al.* la definen como la estructuras o estructuras de un sistema, comprendiendo los componentes de software así como sus propiedades externamente visibles y las relaciones entre ellos [3]. La arquitectura en un sistema de software cobra singular importancia ya que facilita la comunicación entre todos los interesados en el sistema, nos permite tomar decisiones de diseño y es "crítica para la realización de los atributos de calidad de un sistema". Debido a la complejidad en la representación de la arquitectura de un sistema de software han surgido diferentes propuestas [4] [5].

Así como en sistemas aislados la arquitectura es un aspecto primordial, la arquitectura de una LPS tiene singular importancia ya que es el artefacto central de la ILPS, ésta provee la base para desarrollar e integrar los activos esenciales *(core assets)*.

Además, una arquitectura de software para LPS debe exhibir ciertos atributos de calidad específicos de las LPS. El modelo de calidad CAFÉ menciona los siguientes atributos de calidad:

1. Variabilidad (variability)
2. Facilidad de derivación (derivability)
3. Reusabilidad
4. Facilidad de estimación
5. Facilidad de integración
6. Conformidad (correctness)
7. **Capacidad de Evolución (evolvability)**
8. Facilidad de administración
9. Fácil mantenimiento (maintainability)

Por otro lado, la separación de intereses (*concerns*) es un principio clave en el desarrollo de software. El desarrollo orientado a aspectos (DOA) es el último paso que se ha tomado para obtener software que sea más fácil de evolucionar, mantener, comprender, personalizar y reutilizar. Un aspecto es definido por Chitchyan *et al.* como factores de calidad o funcionalidades del software que no pueden ser modularizadas efectivamente utilizando las técnicas de desarrollo de software existente [6]. La separación de intereses puede ser aplicada desde las primeras fases de desarrollo de software. Los intereses en dichas fases se encuentras mezclados y dispersos en los requerimientos y/o en los artefactos arquitectónicos. A dichas intereses se les conoce con el nombre de Aspectos Tempranos.

Los beneficios de la separación de aspectos en la arquitectura de un sistema deberían hacer el diseño una tarea más fácil, reduciría su costo y mejoraría la reutilización. Otros de los beneficios esperados sería la obtención de software más simple de evolucionar, mantener, comprender y reutilizar por lo que su inclusión en las LPS era un paso natural en la evolución de esta nueva forma de desarrollar software. Es necesario hacer notar que si en una LPS no son tratados eficientemente los diferentes intereses la consecuencia sería la pérdida de facilidad administración y evolución dentro de la LPS, por lo que la inclusión del desarrollo orientado a aspectos en una LPS se justifica plenamente.

2 Documentación

2.1 Documentación de arquitecturas

Es necesario que todos los interesados en un sistema puedan entender su arquitectura por medio de descripciones detalladas y de manera organizada ya que de otra manera la arquitectura sería inutilizable. Debido a la importancia de tener una arquitectura de software bien documentada surgen diferentes formas de representación en las que encontramos principalmente a los Lenguajes de Descripción de Arquitecturas y la Documentación por medio de Vistas Arquitectónicas.

En la actualidad la documentación por medio de vistas tiene gran influencia en la documentación de arquitecturas siendo el modelo 4+1 de Krutchen [4] y *Views and Beyond* de Clements *et al.* [5] los más influyentes..

2.2 Documentación de arquitecturas orientadas a aspectos

En la actualidad no existe una forma de documentación de arquitecturas orientadas a aspectos ampliamente aceptada. Aún cuando ha habido varias propuestas de modelado basadas en UML es la propuesta Krechetov *et al.* [7] la primera que busca integrar las mejores prácticas del diseño de arquitecturas orientadas a aspectos (PCS Framework, AOGA, TranSAT y CAM) en un solo lenguaje de modelado.

Este lenguaje unificado propone los siguientes conceptos de Desarrollo de Software Orientado a Aspectos (DSOA):

- Aspecto
- Componente
- Punto de corte (*Point-cut*)
- Curso de acción (*Advice*)
- *Static and dynamic crosscutting*
- Relación Aspecto-Componente
- Relación Aspecto-Aspecto

2.3 Documentación de arquitecturas de líneas de productos de software

Al igual que en la documentación de arquitecturas de software la arquitectura de una LPS no se puede representar en una sola entidad por lo que se necesitan diferentes vistas de dicha arquitectura. Además, es necesario comunicar todas las decisiones en cuanto a la variabilidad haciendo posible el uso de la arquitectura en un futuro. Una propuesta para la documentación de arquitecturas de LPS es el modelo ortogonal de variabilidad [8] el cual se encarga de documentar la variabilidad en la arquitectura.

3 Propuesta

3.1 Objetivo

Desarrollar una estrategia para la documentación de arquitecturas orientadas a aspectos para LPS que facilite su creación y evolución así como la derivación en arquitecturas de productos específicos dentro de la línea.

Además se busca facilitar la integración de la OA a las LPS, apoyar en el proceso de derivación de arquitecturas de LPS y apoyar en las actividades propias del arquitecto de software tales como la transformación de la arquitectura, análisis de atributos de calidad, elaboración de modelos y la selección de estilos arquitectónicos principalmente.

3.2 Requerimientos

Hay una necesidad de proveer una forma de documentar las arquitecturas OA de LPS. Nuestra guía para la elaboración de la propuesta se compone de los siguientes requerimientos:

R1. Utilizar las estrategias de documentación de arquitecturas, de documentación de arquitecturas orientadas a aspectos y de documentación de LPS existentes y de mayor aceptación

R2. El lenguaje de modelado para la documentación debe ser de propósito general y basarse en UML 2

R3. Debe estar completa, es decir, debe soportar todas las abstracciones de cada concepto de DSOA y representar la variabilidad de la LPS.

R4. Debe promover la simplicidad y evitar las extensiones innecesarias.

3.3 Documentación de Arquitecturas Orientadas a Aspectos para LPS

Como se menciona en secciones anteriores las prácticas modernas para la documentación de arquitecturas utilizan el concepto de vista para representar el conjunto de elementos que forman un sistema y sus relaciones. Como resultado de la investigación se optó por tomar como modelo la aproximación de Clements *et al.* llamada "*Views and Beyond*" [5].

La propuesta de documentación se compone de 3 actividades principales. En la figura 1 se presenta el diagrama de actividades de la propuesta de documentación.

La primera de ellas es la sección de las vistas relevantes a documentar ya que la mayoría de las organizaciones no pueden mantener la documentación de un número elevado de vistas. El proceso que se sugiere para escoger las vistas relevantes es el siguiente:

1. **Producir una lista de vistas candidatas:** Se construye una tabla de interesados y vistas de la arquitectura orientada a aspectos de LPS colocando al inicio de cada fila el tipo de interesado y en las columnas las vistas que apliquen. Una vez definidas las filas y columnas se rellenan las celdas con el grado de información que requiere el interesado de la vista: ninguna, general o detallada. Finalmente se obtiene la lista de las vistas candidatas tomando las vistas en las que los interesados hayan puesto mayor interés.
2. **Combinar vistas:** De la lista creada en el paso interior comúnmente se puede obtener un número impráctico de vistas a documentar. En este paso se reduce a un número manejable de vistas a documentar. Primero se buscan las vistas que sólo necesitan información general para ver si los interesados pueden obtener la información que necesitan de otra vista en la que haya más interés. De esta manera estas vistas son buenas candidatas a ser vistas combinadas. El que haya una buena correspondencia entre elementos de diferentes vistas es otro signo para realizar una vista combinada.
3. **Priorizar:** Después del segundo paso se obtienen el conjunto mínimo de vistas que los interesados necesitan. Ahora se decide qué vista debe hacerse primero de acuerdo a la prioridad de intereses.

Una vez que se han obtenido las vistas candidatas el segundo paso de la propuesta es la documentación de las vistas obtenidas en el paso 1 las cuales formarán parte del segundo volumen de los dos volúmenes que conforman el documento final. Aún cuando las vistas representan información diferente todas deberán seguir una plantilla estándar que constará de diferentes secciones. La plantilla para la documentación de cada vista deberá contener las siguientes secciones:

1. **Una presentación general (textual):** Esta deberá incluir los antecedentes y el estilo que se presenta.
2. **Un catálogo de elementos:** Todos los elementos que aparecen en la vista listando sus propiedades.
3. **Una representación gráfica:** Mostrará los elementos (incluyendo los elementos aspectuales, variables y opcionales) y relaciones entre ellos.

4. **Una guía sobre la variabilidad:** Mostrará la variabilidad propia de la LPS en la vista arquitectónica.
5. **Conflictos:** Información sobre los conflictos que existan entre los elementos y sus relaciones.
6. **Atributos de calidad:** Se listas y describen los atributos de calidad exhibidos en la vista.
7. **Información para diseño:** Se incluirá la información que se considere relevante para el diseño.

En el último paso para completar la documentación de una arquitectura orientada a aspectos de una línea de productos de software se generan los documentos que aplican a más de una vista y que aplican al paquete de documentación como un todo.

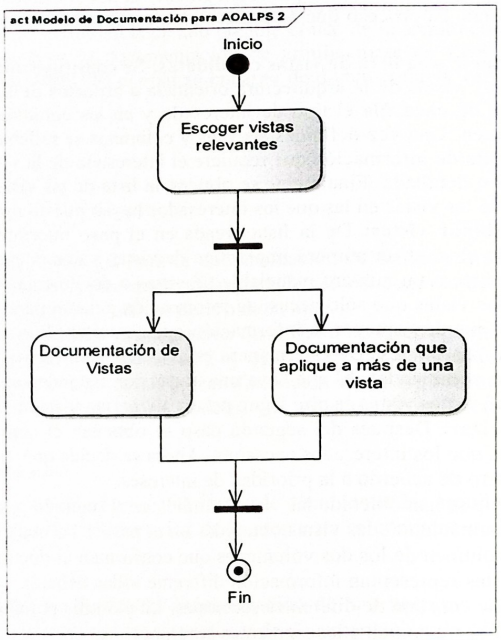

Fig. 3. Actividades de documentación

El objetivo de esta documentación es unir la documentación de las demás vistas dándoles coherencia y nos permitirá ver las diferentes vistas como partes de la arquitectura en general de un sistema y conformará el primer volumen del documento final.

La documentación que aplica a más de una vista también se propone como una plantilla la cual contendrá la siguiente información:

1. **Introducción:** Se explica al lector de la documentación la información que el arquitecto ha decido documentar. Se hace una breve explicación de las vistas, su propósito y una lista de interesados en dicha vista. También se pueden incluir posibles escenarios
2. **Plantilla para las vistas:** Se anexa la plantilla que se utilizará en las vistas que se documente. El objetivo es facilitar al lector el encontrar la sección que más le interese.
3. **Resumen de la línea de productos de software:** Se incluirá información que describa a la función de la LPS, quienes son sus usuarios, cualquier restricción que sea necesaria.
4. **Relaciones (*mapping*) entre vistas:** Información en la que se describa la correspondencia entre los elementos individuales de una vista a otra. Esta información ayudará al lector a ver las diferentes vistas como partes de una sola arquitectura.
5. **Decisiones que trastocan varias vistas:** Se describen todas las decisiones que han tomado los arquitectos al diseñar la LPS como un todo.
6. **Aspectos:** Información sobre los aspectos que tocan más de una vista así como sus restricciones y decisiones aspectuales.
7. **Atributos de calidad:** Información acerca de los atributos de calidad que debe exhibir la LPS.
8. **Glosario:** Se debe incluir una lista de términos que se encontrarán en la documentación.
9. **Versión:** Información para el control de versiones como el historial de revisiones de la documentación así como información de contacto del mantenedor.

La segunda y tercera actividad, es decir, la documentación de las vistas y la documentación que aplique a más de una vista son actividades que se pueden realizar en paralelo.

Cabe hacer notar que la propuesta cumple con todas las condiciones necesarias para estar en conformidad con el estándar IEEE 1471 [9] ya que:

1. Incluye información para la identificación de la documentación, resumen, contexto, glosario, historia de los cambios y referencias.
2. Muestra una lista de los interesados más comunes y sus intereses.
3. Se identifican las vistas arquitectónicas relevantes y se especifica el nombre de la vista; los interesados a los que está dirigida la vista; el lenguaje, técnicas de modelado y métodos analíticos utilizados; una explicación sobre la selección de las vistas e información adicional que se considere necesario
4. Cada vista arquitectónica debe pertenecer exactamente a un punto de vista arquitectónico.
5. Se incluye un análisis de las inconsistencias entre todas las vistas.
6. Se incluye información para el análisis de la arquitectura.

3.4 Documentación de las vistas usando SplaoaML

Para la documentación de las vistas se propone el uso de un perfil (*profile*) de UML creado especialmente para la documentación de Arquitecturas OA para LPS. Dicha decisión obedece a que UML es el lenguaje de modelado de mayor aceptación y esto sin lugar a dudas facilitará tanto la comprensión de las extensiones que se proponen como la adopción de la propuesta tanto en el mundo académico como en la industria.

Los perfiles UML constituyen un mecanismo que el propio UML provee para extender tanto su sintaxis como su semántica y así pode expresar cierto dominio de aplicación.

El perfil UML propuesto provee los elementos de modelo clave para especificar AOALPS. En primer lugar se propone perfiles para los elementos arquitectónicos simples en las vistas propuestas por V&B abarcando los siguientes estilos arquitectónicos:

- Tipo de vista módulos:
  - Descomposición
  - Usos
  - Generalización
  - Capas
- Tipo de vista componentes y conectores:
  - *Pipe-and-filters*
  - Datos compartidos
  - Publicación-subscripción
  - Cliente-servidor
  - De igual a igual (*peer-to-peer)*
  - Procesos en comunicación
- Tipo de vista asignación
  - De despliegue

Para la representación de los aspectos arquitectónicos se tomó como base la propuesta de Tekinerdogan *et al.* [10] y se integró en los puntos de vista de módulos y componentes y conectores. De esta manera se podrán documentar explícitamente:

- En el tipo de vista de módulos:
  - Aspectos
  - Relaciones *crosscutting*
  - Puntos de corte *(point cuts)*
  - *Advices*
- En el tipo de vista componentes y conectores
  - Conectores aspectuales
  - Roles *crosscutting*
  - Relaciones crosscutting
  - Secuencia

La información aspectual presentada en los diagramas que ocupen este perfil será de gran utilidad para la fase de diseño y construcción. En el caso del punto de

vista de asignación no se consideró necesario agregar información arquitectónica aspectual ya toda la información asociada quedará documentada en algunas de las vistas correspondientes al punto de vista de componentes y conectores.

Para hacer la representación gráfica de los elementos arquitectónicos propios de una LPS se integró el modelo ortogonal de variabilidad propuesto por Pohl *et al.* [8] y se crearon nuevos elementos producto de la combinación de las propuestas.

En el desarrollo del perfil UML (SplaoaML) se utilizaron cuatro metamodelos:

- SplaoaLM-Aspects: Metamodelo que define todos los elementos de la orientación a aspectos a nivel arquitectónico.
- SplaoaLM-LPS: Metamodelo del modelo ortogonal de variabilidad el cual brinda los elementos necesarios para representar la variabilidad en una LPS.
- SplaoaLM-Module: Metamodelo para documentar el punto de vista de módulos. Incluye elementos de SplaoaLM-Aspects y SplaoaLM-LPS además de los estilos arquitectónicos más utilizados.
- SplaoaLM-C&C: Metamodelo que documenta el punto de vista de componentes y conectores. Incluye elementos de SplaoaLM-Aspects, SplaoaLM-LPS y los estilos más comunes.
- SplaoaLM-Allocation: Metamodelo para el modelas los estilos más comunes del punto de vista de asignación.

3.5 Herramientas de soporte

Una de los aspectos principales para que la propuesta pueda ser adoptada es que pueda utilizarse alguna herramienta CASE para la representación gráfica de las vistas arquitectónicas. La herramienta que actualmente puede dar soporte a la propuesta es Enterprise Architect ya que integra fácilmente el uso de *profiles* de UML externos. En la propuesta se incluyen los archivos xml necesarios para poder utilizar el perfil SplaoaLM.

4 Evaluación

Para evaluar la propuesta presentada en el artículo se decidió utilizarla en un caso de estudio. Dicho caso de estudio es la documentación de una LPS que se busca implementar dentro de la Facultad de Estadística e Informática de la Universidad Veracruzana la cual dará soporte a la metodología de análisis de requerimientos Áncora [11] con el fin de contar con herramientas que sirva tanto a estudiantes de la facultad como a desarrolladores en general.

El caso de estudio ya se ha iniciado. encontrándose en las etapas de ingeniería de dominios. A la fecha, el desarrollo de la arquitectura de la LPS apenas comienza por lo que la aplicación de la propuesta así como la evaluación de los resultados aún se encuentra pendiente.

5 Conclusiones

La Orientación a Aspectos es útil para obtener mayor modularidad, facilitar el mantenimiento y la evolución de una LPS. El contar con una técnica para documentar una arquitectura OA para LPS nos permitirá tener un medio de comunicación entre todos los interesados en el sistema, servir como base para el análisis de un sistema y como un medio de aprendizaje sobre el mismo sistema.

A partir de las estrategias actuales en las diferentes formas de documentar arquitecturas se ha propuesto una estrategia que recoge lo mejor de cada una. También nos basamos en el uso de UML para que la propuesta sea de fácil adopción tanto en la academia como en la industria.

Finalmente podemos decir que la presente propuesta toma una de las diferentes direcciones que puede tomar una estrategia de documentación de arquitecturas orientadas a aspectos para LPS por lo que es necesario investigaciones futuras en los siguientes campos:

- Exploración del tema desde un punto de vista formal utilizando un lenguaje de descripción de arquitecturas.
- Inclusión de vistas diferentes las presentadas en la propuesta.
- Inclusión de información diferente a la propuesta.

Referencias

1. Clements, P., Northrop, L., *Software Product Lines: Practice and Patterns*, Addison-Wesley Professional, Boston, 2001.
2. R. Chitchyan, A. Rashid, P. Sawyer, A. Garcia, M. Pinto Alarcon, J. Bakker, B. Tekinerdoğan, S. Clarke, and Andrew Jackson. "Survey of Aspect-Oriented Analysis and Design Approaches". Technical Report AOSD-Europe-ULANC-9, AOSD-Europe, 2005.
3. Bass, L., Clements, P., Kazman, R., "Software Architecture in Practice". Addison-Wesley Longman Publishing Co., Boston, 1998.
4. Kruchten, P., "The 4+1 View Model of Architecture", *IEEE Software* ,vol. 12, no. 6, pp. 42-50, November, 1995.
5. Clements, Paul., Garlan, D., Little, R., Nord, R., Stafford, J., *Documenting Software Architectures: Views and Beyond.* Boston, 2003.
6. Chitchyan R., Rashid A., Sawyer P., Garcia A., Pinto Alarcón M., Bakker J., B. Tekinerdogan., Clarke S., Jackson, A., "Survey of AspectOriented Analysis and Design Approaches", University of Lancaster, Lancaster, UK, AOSDEuropeULANC9, Editor(s): R. Chitchyan, A.Rashid, 2005.
7. Krechetov, I., Tekinerdogan, B., Garcia, A., Chavez, C., and Kulesza, U., "Towards an integrated aspect-oriented modeling approach for software architecture design". 8th workshop on aspect-oriented modelling. aosd.06. *IEEE Trans. Soft. Eng*, 1995.
8. Pohl, K., Böckle, G., van der Linden, F. J. *Software Product Line Engineering: Foundations, Principles and Techniques,* Springer, 2005.
9. Maier, M. W.; Emery, D.; Hilliard, R. "Ansi/ieee 1471 and systems engineering," *Systems Engineering*, vol. 7, no. 3, pp. 257-270, 2004.
10. Tekinerdogan, Bedir, y otros. *AOSD-Europe-UT-D76.* s.l. : AOSD-Europe, 2007.
11. Sumano López, María de los Ángeles. *Ancora: Análisis de requerimientos de software conducente al reuso de artefactos.* 1a ed. Xalapa, Universidad Veracruzana, 2006