Uso del paradigma de Línea de Productos de Software (SPL) en Aplicaciones Móviles Sensibles al Contexto

Use the paradigm of Software Product Line (SPL) in Mobile Context-Aware Applications

Patricio Espinel

Departamento de Eléctrica y Electrónica, Universidad de la Fuerzas Armadas ESPE Extensión Latacunga
Correspondencia Autor: gpespinel@espe.edu.ec

Recibido: agosto 2016, **Publicado:** diciembre 2016

***Resumen*—** En este artículo exploramos los diferentes enfoques de la aplicabilidad del paradigma de la Línea de Productos Software (SPL) en el desarrollo de sistemas adaptables para dispositivos móviles. En particular, las aplicaciones móviles sensibles al contexto, las mismas que consideran la posición del usuario como un aspecto relevante para brindar servicios o información, es decir que tienen que supervisar los eventos y la información proveniente de su entorno y reaccionar en consecuencia. Al mismo tiempo, observamos que un número importante de aplicaciones móviles comparten varias características en cuanto a su arquitectura, comunicación, almacenamiento e interfaces. Esto nos lleva a considerar que los sistemas sensibles al contexto también pueden beneficiarse del paradigma de la SPL.

***Palabras Claves*—** Línea de Productos de Software, SPL, Modelo de la Variabilidad, Movilidad, Sistemas Sensibles al Contexto.

***Abstract*—** Software Product Line, SPL, Variability Modeling, Mobility, Context-aware Systems.

***Keywords*—** In this paper we explore the different approaches of applicability of the Software Product Line (SPL) paradigm into the development of adaptable systems for mobile devices. Specifically, mobile context-aware applications which consider the user position as a relevant aspect in order to give services or information, i.e. they have to monitor the events and information coming from its environment and react accordingly. At the same time, we notice that an important number of such mobile applications share several characteristics regarding its architecture, communication, storage and interfaces. This leads us to consider that context-aware systems can also benefit from the SPL paradigm.

I. INTRODUCCIÓN

Los sistemas de software han ido evolucionando independientemente de cuál sea su dominio de aplicación, actualmente nos encontramos con aplicaciones que pueden brindar servicios de acuerdo a la ubicación del usuario, la hora actual, el perfil de usuario y la conectividad [1]. Este tipo de aplicaciones que brindan estos servicios se las conoce como sistemas sensibles al contexto, las cuales predominan en los nuevos dispositivos móviles (por ejemplo, teléfonos móviles, PDAs) que están continuamente presentes en nuestras tareas diarias [2].

Los sistemas sensibles al contexto son parte de una amplia gama de sistemas dentro de la computación ubicua [3]. Estos sistemas utilizan la información del contexto para proporcionar servicios e información relevante a los usuarios. En las diferentes situaciones del contexto, los usuarios pueden acceder a otros datos y explotar los diversos aspectos de una aplicación. Por ejemplo, cuando un turista llega a un nuevo lugar y accede a una aplicación de guía turística móvil en su teléfono inteligente, espera que las visitas recomendadas se basen en sus preferencias y ubicación [4].

Las SPL tienen como objetivo administrar y construir múltiples productos de software a partir de un conjunto de assets previamente desarrollados y probados. Un assets se entiende como cualquier artefacto de software que se puede emplear en el desarrollo de una aplicación. En la ingeniería de la SPL, se identifican los aspectos comunes y variables a través de un conjunto de aplicaciones (es decir, la familia de productos), de manera que los assets puedan desarrollarse y utilizarse para crear diferentes productos [5]. Además, este paradigma ha demostrado ser una forma eficaz de hacer frente a las necesidades de los diferentes usuarios [6].

Aunque, la mayoría de los enfoques de la SPL se han centrado en el desarrollo de productos configurados estáticamente utilizando los assets con los puntos de variación [7]. Todas las variaciones se instancian antes de que un producto se entregue a los clientes. Por lo tanto, estos enfoques proporcionan una adaptación en tiempo de desarrollo en la que se han generado diferentes versiones de la aplicación de acuerdo con los clientes y las características específicas del entorno de ejecución [2].

Por ejemplo, una aplicación móvil tal como una guía turística está destinada a ejecutarse en una variedad de dispositivos móviles y adaptarse a las diferentes preferencias del usuario. Por lo tanto, si se utiliza la adaptación en tiempo de desarrollo, el número de versiones aumentará exponencialmente. Los dispositivos móviles

tienen limitada memoria, almacenamiento y procesamiento, y no siempre es posible cargar al mismo tiempo todos los componentes necesarios para todos los contextos de ejecución [2].

Este enfoque estático no es adecuado para hacer frente al dinamismo de las aplicaciones sensibles al contexto porque necesitan una adaptación en tiempo de ejecución en la que las aplicaciones ajusten su comportamiento de acuerdo con los cambios del contexto. El desarrollo de aplicaciones sensibles al contexto debería beneficiarse del concepto de la SPL en términos de reutilización y configuración. Sin embargo, presenta nuevos desafíos para la ingeniería de la SPL [8].

El principal desafío de la ingeniería de la SPL respecto a las aplicaciones sensibles al contexto es poder manejar adecuadamente la variabilidad en los sistemas adaptativos, muchos [9, 10] abogan por el uso de las técnicas de la SPL debido a los beneficios complementarios de ambos conceptos. Por lo tanto, el objetivo de este trabajo es explorar la aplicabilidad del paradigma de la SPL para el desarrollo aplicaciones móviles adaptables, a través del análisis de las diferentes contribuciones.

El resto de este documento se divide en tres secciones, además de esta introducción. En la Sección 2 la revisión literaria de las técnicas para el modelado de la variabilidad de las SPL para la construcción de sistemas adaptables es presentada. Los resultados del análisis de la revisión literaria se detallan en la Sección 3. Por último, las conclusiones son descritas en la Sección 4.

II. MATERIALES Y MÉTODOS

EL objetivo de esta sección es dar una idea de cómo se trata el modelado de la variabilidad en la SPL para el desarrollo de aplicaciones móviles sensibles al contexto y por lo tanto tener una mejor comprensión de las diferentes técnicas presentes en la literatura. En esta sección describimos aspectos relacionados al modelado de la variabilidad. Comentamos algunos conceptos a tener en cuenta en la documentación de la variabilidad, los que han sido propuestos a lo largo de las investigaciones en esta área. Por último, presentamos de forma breve algunas técnicas de modelado propuestas en los últimos años.

A. Propuesta modelos de características

Existe en la literatura un gran número de técnicas para modelar la variabilidad de la SPL. Las más conocidas son los modelos de características y se basan en el método de análisis FODA (Feature-Oriented Domain Analysis) [11], por ejemplo, Kang et al. [12] y Griss et al. [13] presentaron los métodos FORM (Feature-Oriented Reuse Method) y Feature-RSEB (Reuse-Driven Software Engineering Business) como extensiones del método FODA; Gurp et al. [14] propusieron una extensión a Feature-RSEB, en la que añaden informaciones extras a los modelos; Eriksson et al. propusieron el PLUSS (Product Line Use case modeling for System and Software engineering) como una extensión de Feature-RSEB [15], en la que combinan diagramas de características (Feature Diagram) y diagramas de casos de uso para representar la SPL en una visión de alto nivel.

B. Propuesta otras técnicas para el modelado de la variabilidad

Otras técnicas para el modelado de la variabilidad, por ejemplo, El OVM (Orthogonal Variability Modeling) es una metodología propuesta por Pohl et al. [16] para el modelo de la variabilidad en las SPL, que propone un modelo de variabilidad separado de los artefactos de la SPL; COVAMOF es un framework para el modelado de SPL que modela la variabilidad en términos de puntos de variación (variation points) y dependencias (depedencies) en diferentes niveles de abstracción [17]; Klaus Schmid e Isabel John [18] proponen una técnica para el modelado de la variabilidad en la que las características variables son capturadas en modelos de decisión (decision models); Decisionking es una herramienta para el modelado de la variabilidad propuesta por Dhungana et al. [19], la que permite crear herramientas para el modelado de la variabilidad para diferentes dominios y organizaciones; VSL (Variability Specification Language) fue presentado en [20] el autor propone un modelo general para la variabilidad, en el que considera que la variabilidad en las SPL debe ser abordada en dos niveles de abstracción: el nivel de especificación (specification level) y el nivel de realización (realization level).

C. Propuesta de Kramer

Kramer en [21], propone ayudar a integrar el modelado del contexto, las características y sus dependencias con las plataformas de hardware y firmware. Para complementar esto, un DSL (Domain Specific Languages) será definido como el método de expresión del modelo, el cual puede entonces proporcionar transformaciones de modelo a código suministrando apoyo a la derivación de productos en las SPL para móviles sensibles al contexto. Esta derivación de productos necesitará manejar diferencias de versión de la plataforma, diferencias de hardware y diferencias del contexto.

D. Propuesta CANDEL

En [4], Jaroucheh et al. presentan CANDEL, un framework genérico de representación de la información del contexto que tiene en cuenta el entorno como una línea de productos dinámica compuesta de primitivas de contexto (context primitives CPs). El framework se fundamenta en técnicas de la SPL que se utilizan junto con la ontología OWL (Ontology Web Language) para definir CPs y generar dinámicamente el modelo del contexto actual. Además, usando Petri-Nets, también se muestra cómo este

framework se utilizará para soportar las aplicaciones ubicuas adaptivas sensibles al contexto.

E. Propuesta de Medeiros et al.

Medeiros et al. en [22], presentan un enfoque sistemático para diseñar arquitecturas de la línea de productos orientadas a servicios, el cual combina los conceptos de la SPL y la SOA (Service-Oriented Architecture) centrándose en aumentar la reutilización y flexibilidad, y apoyando la personalización durante el desarrollo de sistemas orientados a servicios.

F. Propuesta de Perrouin et al.

En [23], Perrouin et al. proponen una SPL basada en el AOM (Aspect Oriented Modeling). En su propuesta, las variantes se especifican gracias a los fragmentos del modelo y la derivación de productos se realiza automáticamente fusionándolos entre sí.

G. Propuesta de Hallsteinsen et al.

En [6], Hallsteinsen et al. utilizan las técnicas de las líneas de productos para la construcción de sistemas adaptables. En su propuesta, los sistemas se implementan utilizando la arquitectura basada en componentes y el modelado de la variabilidad, y delegan la complejidad de la adecuación a una plataforma de adaptación reutilizable.

H. Propuesta de Salifu et al.

En [24], Salifu et al. proponen un enfoque de variabilidad para las familias de productos software que se ocupan de la sensibilidad del contexto. Vinculan los requisitos a la arquitectura de software utilizando los paradigmas de la familia de productos, pero no soportan la integración del contexto en la plataforma de ejecución.

I. Propuesta de Hartmann et al.

En [25], Hartmann et al. proponen el concepto de un modelo de variabilidad del contexto que contiene los principales factores de variación (por ejemplo, diferentes regiones geográficas). Este modelo restringe al modelo de características para elegir una dimensión en el espacio del contexto. Este modelo de variabilidad del contexto se mantiene como un medio estático para gestionar una derivación de productos de acuerdo con una descripción de la variabilidad ortogonal.

J. Propuesta CAPucine

Finalmente, en [26], Parra et al. presentan CAPucine una DSPL (Dynamic Software Product Line) para construir aplicaciones orientadas a servicios y su adaptación en tiempo de ejecución de acuerdo con el contexto utilizado. Su enfoque es homogéneo con dos procesos diferentes para las fases inicial e iterativa de la derivación de productos. En primer lugar, proponen un asset sensible al contexto que introduce alternativas en la SPL y que se considera en tiempo de ejecución. En segundo lugar, se basan en un conjunto de técnicas de realización de la variabilidad por el camino de las herramientas sensibles al contexto, como COSMOS y las herramientas de reconfiguración dinámica como FraSCAti.

III. RESULTADOS

En el ambiente ubicuo, es esencial para las aplicaciones informáticas ser sensibles al contexto. Sin embargo, uno de los principales retos es el establecimiento de un genérico y dinámico modelo del contexto. Existen muchos enfoques diferentes para modelar el contexto, sin embargo, una aplicación y un dominio de diagnóstico sensible al contexto, que capture los distintos tipos de información del contexto y la dependencia entre ellos, que podrían ser reutilizados y compartidos por las diferentes aplicaciones, y que puedan ser dinámicamente modificados cuando un cambio de posición se produce, no se encuentra. Por lo tanto, existe interés en la definición de una estructura para la gestión dinámica de la información del contexto.

En 1990 Kang et al. presentaron los modelos de características por primera vez. A partir de entonces han sido muchas las extensiones y mejoras que se han propuesto sobre dichos modelos que han intentado incrementar su capacidad expresiva. Actualmente estos modelos son considerados una de las más importantes contribuciones para la ingeniería de las SPL. No obstante, dichos modelos no son la única forma de representar las SPL, existen otras propuestas en la literatura con este mismo objetivo. Una de ellas es OVM, sus autores proponen un modelo específico para documentar la parte variable entre los productos. En sus recientes publicaciones presentan algunos trabajos que han sido desarrollados en el contexto del análisis automático.

Los enfoques basados en la SPL por lo general proponen un desarrollo de software sistemático basado en una familia de productos, lo que puede beneficiar al desarrollo de software sensible al contexto en términos de reutilización y capacidad de configuración. UbiFEX [4, 21] se ha presentado como una notación del modelo de características para las SPL sensibles al contexto. Esta notación permite la configuración de un producto específico para la regla del contexto, pero no termina la necesidad de modelar el sistema completo combinando características y contextos.

Se han propuesto enfoques como MDA (Model Driven Architecture) para el desarrollo de software sensible al contexto. Así como CAMEL (Context Awareness Modeling Language) [21, 27], y un diseño de DSL que ofrece un método para el modelamiento de los comportamientos dependientes del contexto para la etapa inicial de desarrollo de software. Los Autores han señalado

que actualmente el lenguaje no maneja el modelado de transformaciones a código ejecutable.

La SOA cuyos logros recientes han permitido el desarrollo de los sistemas móviles [26, 28] y las SPL son dos paradigmas arquitectónicos que reciben bastante interés por parte de los investigadores. A lo largo de las décadas pasadas ambas estrategias se aplicaron a grandes proyectos revelando muy buenos resultados en función de las características como la flexibilidad y el reúso.

Sin embargo, existieron algunas limitaciones como la dependencia de las tecnologías y las plataformas para las líneas de productos, así como la ausencia de mecanismos de personalización y reutilización de los artefactos en el caso de las arquitecturas orientadas a servicios.

La DSPL, llamada CAPucine [5] para Línea de Productos Orientada a Servicios Sensible al Contexto, propone un enfoque unificado que soporta el ciclo de vida completo de software: desde la selección de las características y la derivación inicial de productos, para la adaptación en tiempo de ejecución en respuesta a los cambios en el entorno de ejecución. Es un enfoque completo para el diseño y la implementación de la DSPL (Fig. 1), se basa en dos procesos diferentes para la derivación de productos. El primer proceso utiliza assets que representan las características de la familia de productos. Los assets, representados como modelos, se integran y se transforman con el fin de generar el producto. El segundo proceso se refiere a la adaptación dinámica. Este proceso introduce assets sensibles al contexto que operan en tiempo de ejecución. Estos assets sensibles al contexto contienen tres tipos de datos: el contexto en el que los assets pueden ser modificados, el lugar donde los assets deben ser aplicados y el cambio que debe ejecutarse. La realización de estos assets sensibles al contexto combina dos plataformas en tiempo de ejecución. Por un lado, COSMOS es un framework sensible al contexto conectado al entorno mediante el uso de sensores. Por el otro lado FraSCAti es una plataforma de la SCA (Arquitectura de Componentes de Servicios) con propiedades dinámicas que permite enlazar y desenlazar los componentes en tiempo de ejecución [26].

La SOA sigue un enfoque de reutilización y composición de servicios mientras que la SPL corresponde a un enfoque de construcción y descomposición. Pocos enfoques de investigación recientes enfatizan en una conciliación entre la SPL y la SOA. Se pretende el uso del paradigma SPL para construir sistemas sensibles al contexto basados en servicios SOA, a fin de permitir un desarrollo completo de servicios desde los requisitos hasta la implementación y una gestión del contexto a lo largo del ciclo de vida del software.

IV. CONCLUSIONES

En este trabajo de investigación se ha identificado que existen varios enfoques en la literatura que aplican el paradigma de la SPL para el desarrollo de sistemas adaptables para dispositivos móviles. Además de los conocidos modelos de características, se han estudiado otras propuestas y se ha constatado que la mayoría de ellas no poseen una representación formal y que cada una posee sus propios conceptos para modelar la variabilidad. Las publicaciones acerca de estas propuestas se han escrito desde diferentes puntos de vista, lo que dificulta su comparación.

Los modelos de expresión de la variabilidad que se generan en el desarrollo de una SPL pueden tener una gran cantidad de características y un gran número de combinaciones entre ellas. Por lo tanto, es prácticamente imposible gestionar grandes modelos sin la ayuda de herramientas. El análisis automático de los modelos de variabilidad es un gran reto en este campo de investigación. En los últimos años se han presentado algunas propuestas para el análisis automático, la mayoría de ellas orientadas hacia los modelos de características.

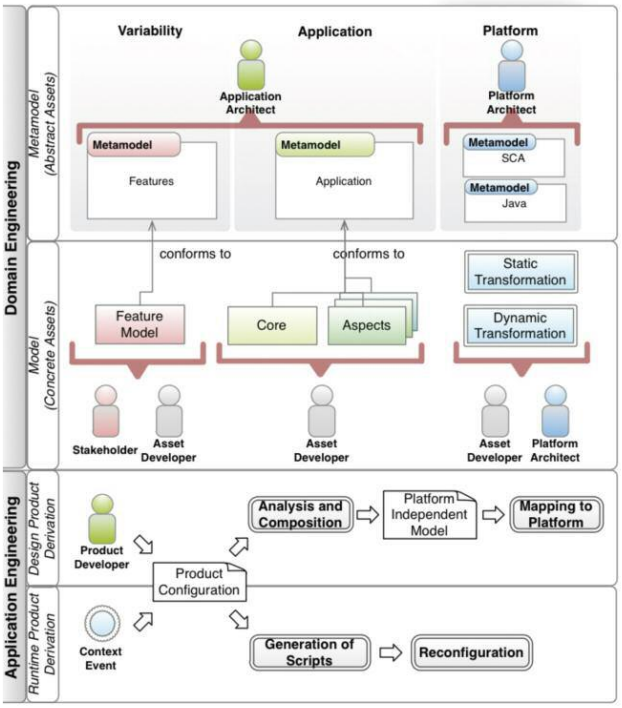

Fig. 1. CAPucine: Proceso de la Línea de Productos Software [5].

En una plataforma de servicios sensible al contexto el conocimiento del entorno debe permitir a los servicios tomar medidas automáticamente, a fin de reducir la participación directa de los usuarios mediante la disminución de la carga comunicativa del sistema y la prestación de asistencia proactiva inteligente. El conocimiento del entorno, la forma de representar el contexto y la información del mundo físico que es interpretable por parte de un sistema repercutirá en el grado de consciencia que los elementos del sistema tengan del mismo; por ello, la forma de modelar este entorno es importante.

Las SPL complementan las soluciones orientadas a servicios a través de técnicas como la gestión de la variabilidad permitiendo que las principales características

de la SOA puedan ser optimizadas a partir del reúso y la facilidad de personalización de los servicios. La mayoría de las investigaciones sobre este tema combinan las propuestas de las SPL y las SOA para diseñar las líneas de productos orientadas a servicios, pero no toman en cuenta las líneas de productos en tiempo de ejecución.

Este es un frente que todos los días recibe mayor interés planteándose una oportunidad para futuras tesis y trabajos de investigación.

REFERENCIAS

[1] Chen, G., & Kotz, D. (2000). A survey of context-aware mobile computing research (Vol. 1, No. 2.1, pp. 2-1). Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College.

[2] Jaroucheh, Z., Liu, X., & Smith, S. (2010, February). CANDEL: product line based dynamic context management for pervasive applications. In Complex, Intelligent and Software Intensive Systems (CISIS), 2010 International Conference on (pp. 209-216). IEEE.

[3] Weiser, M. 1999. The computer for the 21st century. SIGMOBILE Mob. Comput. Commun. Rev. 3, 3 (Jul. 1999), 3-11.

[4] Fernandes, P., Werner, C., & Murta, L. G. P. (2008, July). Feature Modeling for Context-Aware Software Product Lines. In SEKE (pp. 758-763).

[5] Parra, C. A., Quinton, C., & Duchien, L. (2012). CAPucine: Context-aware service-oriented product line for mobile apps. ERCIM News, 88, 38-39.

[6] Hallsteinsen, S., Stav, E., Solberg, A., & Floch, J. (2006, August). Using product line techniques to build adaptive systems. In Software Product Line Conference, 2006 10th International (pp. 10-pp). IEEE.

[7] Gomaa, H., & Hussein, M. (2003, November). Dynamic software reconfiguration in software product families. In International Workshop on Software Product-Family Engineering (pp. 435-444). Springer Berlin Heidelberg.

[8] Sugumaran, V., Park, S., and Kang, K. C. 2006. Introduction – Software product line engineering. Commun. ACM 49, 12 (Dec. 2006), 28-32.

[9] Marinho, F. G., Lima, F., Ferreira Filho, J. B., Rocha, L., Maia, M. E., de Aguiar, S. B., ... & Werner, C. (2010, September). A software product line for the mobile and context-aware applications domain. In International Conference on Software Product Lines (pp. 346-360). Springer Berlin Heidelberg.

[10] Shen, L., Peng, X., & Zhao, W. (2012, July). Software Product Line Engineering for Developing Self-Adaptive Systems: Towards the Domain Requirements. In Computer Software and Applications Conference (COMPSAC), 2012 IEEE 36th Annual (pp. 289-296). IEEE.

[11] K. Kang, S. Cohen, J. Hess, W.˜Novak, and S. Peterson. Feature–Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI- 90-TR-21, Software Engineering Institute, Carnegie Mellon University, November 1990.

[12] K. C. Kang, S. Kim, J. Lee, K. Kim, E. Shin, and M. Huh. FORM: A feature–oriented reuse method with domain–specific reference architectures. Annals of Software Engineering, 5:143–168, 1998.

[13] M. Griss, J. Favaro, and M. d'Alessandro. Integrating feature modeling with the RSEB. In Proceedings of theFifthInternational Conference on Software Reuse, pages 76–85, Vancouver, BC, Canada, 1998.

[14] J. van Gurp, J. Bosch, and M. Svahnberg. On the notion of variability in software product lines. In Software Architecture, 2001. Proceedings. Working IEEE/IFIP Conference on, pages 45–54, 2001.

[15] Eriksson, M., Börstler, J., & Borg, K. (2005, September). The PLUSS approach–domain modeling with features, use cases and use case realizations. In International Conference on Software Product Lines (pp. 33-44). Springer Berlin Heidelberg.

[16] Klaus Pohl, Günter Böckle, and Frank J. van der Linden. Software Product Line Engineering: Fundations, Principles and Techniques. Springer–Verlag, 2005.

[17] Marco Sinnema, Sybren Deelstra, Jos Nijhuis, and Jan Bosch. COVAMOF: A framework for modeling variability in software product families. In Proceedings of the Third International Software Product Lines Conference (SPLC 2004), Springer Verlag Lecture Notes in Computer Science (LNCS 3154), August 2004, pages 197–213.

[18] K. Schmid and I. John. A customizable approach to full-life cycle variability management. Science of Computer Programming, Special Issue on Variability Management, 53(3):259–284, 2004.

[19] D. Dhungana, P. Grünbacher, and R. Rabiser. Decisionking: A flexible and extensible tool for integrated variability modeling. In Proceedings of the First International Workshop on Variability Modelling of Software intensive Systems (VAMOS), pages 119–127, January 2007.

[20] M. Becker. Towards a general model of variability in product families. In 1st Workshop on Software Variability Management, Groningen, Netherlands, February 2003.

[21] Kramer, D. Using Product Lines to Manage Variability in Mobile Context-Aware Applications. In 1 st Doctoral Symposium (p. 61).

[22] Medeiros, F. M., de Almeida, E. S., & de Lemos Meira, S. R. (2010, September). Designing a set of service-oriented systems as a software product line. In Software Components, Architectures and Reuse (SBCARS), 2010 Fourth Brazilian Symposium on (pp. 70-79). IEEE.

[23] G. Perrouin, J. Klein, N. Guelfi, and J.-M. Jézéquel. Reconciling automation and flexibility in product derivation. In 12th International Software Product Line Conference (SPLC 2008), pages 339-348, Limerick, Ireland, Sept. 2008. IEEE Computer Society.

[24] M. Salifu, B. Nuseibeh, and L. Rapanotti. Towards context-aware product-family architectures. In IWSPM '06: Proceedings of the International Workshop on Software Product Management, pages 38–43, Washington, DC, USA, 2006. IEEE Computer Society.

[25] H. Hartmann and T. Trew. Using feature diagrams with context variability to model multiple product lines for software supply chains. In SPLC '08: Proceedings of the 2008 12th International Software Product Line Conference, pages 12–21, Washington, DC, USA, 2008. IEEE Computer Society.

[26] Parra, C., Blanc, X., & Duchien, L. (2009, August). Context awareness for dynamic service-oriented product lines. In Proceedings of the 13th International Software Product Line Conference (pp. 131-140). Carnegie Mellon University.

[27] Sindico, A., Grassi, V.: Model driven development of context aware software systems. In: COP '09: International Workshop on Context-Oriented Programming. pp. 1–5. ACM, New York, NY, USA (2009).

[28] N. Josuttis. SOA in Practice, The Art of Distributed System Design. O'Reuilly, August 2007.