Marco de trabajo para la aplicación de línea de producto en el desarrollo de software para gobierno digital

Mauro Cambarieri, Luis Vivas, Nicolás Garcia Martinez

Laboratorio de Informática Aplicada, Universidad Nacional de Río Negro, Argentina

{mcambarieri, lvivas, ngarciam}@unrn.edu.ar

Abstract. Este articulo presenta un marco de trabajo para el desarrollo de software en gobierno digital, utilizando un enfoque basado en el modelo extractivo, el cual permite que parte de una o varias aplicaciones (productos) ya existentes puedan ser generalizados para que sean objeto de una Línea de Producto de Software (LPS). Este paradigma se centra en gestionar aspectos comunes de los productos, y su complementario, lo variable. La reutilización en el entorno LPS es planificada, y la incorporación de nuevas variantes del producto se realiza de forma sistemática y controlada; esto agiliza el desarrollo, la puesta en marcha y también el mantenimiento. La contribución es un marco de trabajo que permitirá la adopción del paradigma de LPS para el desarrollo de software en gobierno digital bajo el proceso de diseño "bottom up" extractivo que permite identificar componentes en común y una arquitectura de referencia. El marco de trabajo propuesto se valida mediante un caso de estudio referido a sistemas de información de la Universidad Nacional de Río Negro.

Keywords: Línea de productos de software, gobierno digital, reutilización, arquitectura de software.

1 Introducción

En 1968, en el Congreso sobre Ingeniería del Software que organizó el Comité de Ciencia de la OTAN, se introdujo por primera vez la idea de la reutilización. En ese mismo congreso el ingeniero de Bell Laboratories, M.D. McIlroy, afirmó que "La industria del software se asienta sobre una base débil, y un aspecto importante de esa debilidad es la ausencia de una subindustria de componentes" [1].

En el contexto actual de la Ingeniería del Software, se entiende por componente, un fragmento reemplazable de un sistema, dentro del cual se encuentran implementadas un conjunto de funcionalidades. Un componente se puede desarrollar aislado del resto del mundo.

Uno de los desafíos centrales en el desarrollo de software, justamente es la reutilización, lo que permite utilizar nuevamente uno o más artefactos realizados como parte del desarrollo de un nuevo producto o sistema. Existen diferentes técnicas

que han facilitado de alguna manera el uso de artefactos de desarrollo de granularidad cada vez mayor. La reutilización ocurra muchas veces de manera oportunista, esto es utilizar algún artefacto ya desarrollado en el nuevo sistema, de esta manera al no disponer de contextos suficientemente amplios como para detectar los elementos reutilizables y donde puedan utilizarse se desemboca en este tipo de reutilización.

Desde la perspectiva de una institución, en el contexto de gobierno digital, es deseable lograr un enfoque de reutilización sistemática con el fin de lograr beneficios asociados a utilizar los artefactos construidos previamente en cada desarrollo nuevo que se realiza. *Para que la reutilización del software fuera sistemática, los procesos de desarrollo deberían abordar la construcción colectiva de familias de productos relacionados por un dominio*.

Este trabajo explora el paradigma LPS exponiendo las diferentes estrategias para su implementación. El marco propuesto se valida mediante un caso de estudio.

El trabajo está estructurado de la siguiente manera: La Sección 2 presenta los conceptos relacionados, incluyendo Desarrollo Basado en Componentes y Línea de Producto de Software. La Sección 3 explica a través de un caso de estudio y muestra las estrategias de aplicar el enfoque extractivo en LPS. La Sección 4 presenta otros aportes científicos en esta línea de investigación y discute la contribución de este trabajo. Por último, la Sección 5 brinda conclusiones y explica los trabajos futuros.

2 Conceptos Utilizados

Las siguientes secciones presentan los conceptos básicos utilizados en este trabajo, incluyendo: Ingeniería del Software Basada en Componentes y Reutilización (Sección 2.1) y Línea de Producto de Software (Sección 2.2).

2.1. Ingeniería del Software Basada en Componentes y Reutilización

La Ingeniería del Software basada en componentes (CBSE: component-based software engineering) se ocupa del desarrollo de sistemas a partir de componentes reutilizables, para favorecer la reutilización y, de este modo, evitar la duplicación de componentes, impidiendo que el análisis, el diseño, la implementación y la prueba de la misma solución, se desarrolle una y otra vez en muchos de los productos, a diferencia del enfoque tradicional de desarrollo de software que involucra generalmente la construcción de cada una de las piezas desde cero, cada una de ellas implementada a propósito para el producto en curso. Algunas de estas piezas o elementos provienen de productos anteriores y son reutilizados con ciertos cambios [1]. CBSE, tiende a concentrar el esfuerzo en la composición de componentes de software desarrollados (que por lo general poseen una funcionalidad concreta), trata el desarrollo de sistemas a partir de la composición e integración de componentes reutilizables. Algunos de ellos, pueden requerir una vinculación con otros componentes para proveer la funcionalidad completa. En cuanto a la comunicación como la integración de los componentes con los elementos del nuevo sistema que se encuentra desarrollando se realiza entendiendo los componentes como cajas negras,

esto permite que se utilicen tal y como están disponibles y sin considerar ninguna modificación interna. [1]

A diferencia del enfoque tradicional, CBSE incluye las siguientes actividades: 1) Selección y evaluación de componentes, 2) Adaptación, 3) Composición, y 4) Evolución [1].

Según [2] existen dos factores principales cuando el enfoque que se utiliza está basado en componentes: el primero, la reutilización – *la facilidad para reutilizar componentes existentes en la creación de sistemas más complejos* – y el segundo, la evolución – *crear un sistema altamente formado por componentes facilita su mantenimiento, ya que si el diseño es correcto los cambios estarán muy localizados y los efectos tendrán poca o ninguna propagación en el resto de los componentes del sistema*. Estos factores se aplican bajo las siguientes condiciones: en primer lugar, deben existir componentes para su reutilización. Como segundo, debe existir un conjunto de componentes bien diseñados y listos para su utilización. Por otro lado, debe existir un modelo de componentes que soporte el enlazado y la interacción de componentes, esto es, debe existir un marco de referencia estándar en el que los componentes puedan existir y comunicarse. Finalmente, es necesario un proceso y la definición y diseño de arquitecturas que den soporte al CBD.

La reutilización favorece el desarrollo de herramientas de software con un alto grado de flexibilidad, haciendo uso de componentes en diversos proyectos, ampliando las capacidades, mejoras, funcionalidades y mantenimiento del software.

Hay varias definiciones del término *Reutilización de Software*. Algunas de estas definiciones son las siguientes:

"La reutilización de software es el proceso de implementar o actualizar sistemas de software usando activos de software existentes" [3]

"Reutilización de software es el proceso de crear sistemas de software a partir de software existente, en lugar de desarrollarlo desde el comienzo" [4].

Existen varias modalidades de reutilización utilizadas, con ello beneficios y costos según el enfoque que se aplique; el oportunista, se beneficia de componentes de software construidos en productos anteriores, pero que no fueron especialmente desarrollados para ser reutilizados; en el enfoque más planificado o sistemático, los componentes se construyen pensando en que serán reutilizados, lo que significa que debe plantearse el componente con mayor generalidad, y por ello en el momento de su desarrollo, mayor inversión de tiempo y recursos.

2.2. Línea de Producto de Software

La definición más comúnmente aceptada de una LPS procede del autor Clements donde "se definen las líneas del producto de software como un conjunto de sistemas software, que comparten un conjunto común de características (features), las cuales satisfacen las necesidades específicas de un dominio o segmento particular de mercado, y que se desarrollan a partir de un sistema común de activos base (core assets) de una manera preestablecida" [5].

En este contexto, una LPS permite acotar el tiempo de producción de componentes software, con altos niveles de calidad y una alta capacidad de reutilización.

En cuanto a los beneficios y costos, se puede mostrar que existe una tendencia según diferentes casos de estudios realizados, un aumento en la productividad que duplica o triplica la producción de software con respecto a los enfoques tradicionales, por lo tanto, las LPS pueden incrementar significativamente la productividad del software entendida como una reducción del esfuerzo y el costo necesario para desarrollar, mantener y poner en marcha el conjunto de productos software similares [6].

En una LPS, los activos base se benefician también de cada producto desarrollado en la línea dado que toman ventaja del análisis, diseño, implementación, planificación, prueba, estos no son únicamente una mera reutilización de código. Las líneas de productos potencian la reutilización estratégica [2], dado que en este contexto es más planificado que oportunista, [7]: en desarrollo de software tradicional, se aprecia la posibilidad de reutilizar un componente después de haberlo desarrollado; en LPS, la reutilización es planificada, de manera que se reutilizan la mayor parte de los activos base en todos los productos de la línea [6].

Como se muestra en la siguiente figura, tomada de [6], se puede observar que existe un ahorro en cuanto al desarrollo en un entorno LPS con respecto a un entorno convencional: mayor número de productos, mayor esfuerzo. Según [6], en el enfoque tradicional es muy común realizar una copia de una aplicación ya desarrollada que se parece a la se desea desarrollar, y a partir de esa copia esta evoluciona de forma independiente. Este parecido que es en cuanto a la arquitectura, tecnologías, frameworks, requisitos de análisis y diseño, etc. sirve para agilizar el desarrollo y la puesta en marcha de la aplicación en cuestión, pero no en el caso de mantenimiento.

En este contexto ambas aplicaciones se deben mantener de manera independiente, aun cuando el mantenimiento es igual. Existen 2 tipos de mantenimiento: 1- el *correctivo*, hay que depurar el código en ambas aplicaciones; 2- el *perfectivo*, realiza mejoras en las dos aplicaciones. El resultado de realizar esta copia es que los costos de mantenimiento no se reducen, si en cuanto a la reutilización de código y otros elementos, lo que significa claramente una reutilización oportunista.

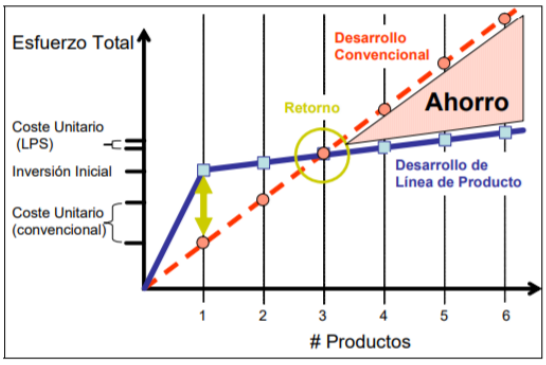

Fig. 1. Desarrollo Convencional vs. Línea de Producto [6].

Por otro lado, en un entorno de LPS, se centra en gestionar aspectos comunes de los productos, y su complementario, lo variable. La reutilización en el entorno LPS es planificada, y la incorporación de nuevas variantes del producto se realiza de forma

sistemática y controlada; esto agiliza el desarrollo, la puesta en marcha y también el mantenimiento. En cuanto al mantenimiento hay una reducción de los costos dado que son capitalizados por todos los productos, cuando se detecta un error o bug en un producto, la corrección del mismo impacta en todos los productos de la línea gracias a la existencia de este marco común que ofrece la LPS.

Podemos concluir en este punto que en un entorno convencional se centra en un único producto y cada uno tiene su mantenimiento y equipo de desarrollo, por el contrario, en LPS se aprovecha los recursos dada la semejanza entre productos, y el número de productos diferentes que podemos gestionar eficazmente.

2.2.1 Aspectos metodológicos - Estrategias

Uno de los factores para aplicar el proceso de desarrollo de la LPS depende del ámbito de la LPS. Es fundamental saber identificar y acotar la familia de productos que serán objeto de la línea, es decir, ¿Cuál es el grado de re-uso que van a soportar los activos base? Para esto se definen 3 enfoques [6]:

- **Modelo proactivo**. El enfoque a largo plazo recibe el nombre de proactivo ya que intenta adelantarse a las necesidades existentes por ejemplo dentro de cinco años, metodológicamente, es equivalente a los métodos en cascada.
- **Modelo reactivo**: Este enfoque se basa en la agilidad, va introduciendo *features* conforme las necesidades del negocio van cambiando. Metodológicamente, se equipara con los métodos ágiles.
- **Modelo extractivo**. Este modelo parte de una o varias aplicaciones ya existentes para generalizarlas hacia una LPS. Tal vez sea éste el enfoque más realista en la medida en que la mayoría de las organizaciones raramente disponen de los recursos para acometer desde cero la LPS, y, sin embargo, sí disponen de aplicaciones cuyo mantenimiento les impulsa a embarcarse en las LPS.

2.2.2 Aspectos metodológicos - Procesos

Una LPS comprende esencialmente 2 etapas:

La ingeniería de dominio- ID (core asset development, denominada por Clements [5]) es la encargada de diseñar los componentes de software que, de acuerdo a su trazabilidad, hacen parte del dominio común y que son sujetos a reutilización. En esta etapa se definen las similitudes y variantes del modelo de dominio en cada una de las fases. Como producto final en esta etapa se considera un componente altamente reusable y flexible que permita a partir de su definición, la consecución de nuevos modelos y artefactos.

La Ingeniería de Aplicación - IA (product development, denominada por Clements [6]) se encarga de articular los artefactos definidos en la etapa de dominio a partir de una plataforma específica. El objetivo de esta etapa consiste en lograr la integración de artefactos bajo uno o varios sistemas, logrando con ello un alto grado de

reusabilidad de sus componentes a partir de las especificaciones, la definición y desarrollo, documentación e interoperabilidad entre componentes.

2.2.3 Gestión en LPS

La gestión juega un papel fundamental para el establecimiento de una línea de productos. La división formal en dos etapas: Ingeniería de Dominio e Ingeniería de Aplicación es complementada con una parte dedicada a la gestión (management), por algunos autores como Clements. Esta actividad juega un papel fundamental en el éxito de la línea de producto, ya que no sólo es la encargada de asignar recursos, sino de coordinarlos y supervisarlos. Esta gestión se realiza tanto en el nivel técnico (gestión de productos concretos) como en el nivel organizativo. Los artefactos de gestión creados como planificaciones también forman parte de los elementos comunes (core assets) [6].

3 Caso de estudio

El Departamento de Soluciones Informáticas de la UNRN, se encuentra actualmente avanzando en pos de mejorar las gestiones de los agentes Docentes y No Docentes incrementando el número de procedimientos digitalizados, para ellos se propuso inicialmente desarrollar en etapas las siguientes aplicaciones:

Aplicación	Descripción
Legajos Digitales (1er etapa)	Es un sistema web que permite generar, administrar, mantener y consultar legajos digitales (legajos de empleados, alumnos, pacientes en los hospitales escuela de Odontología y Veterinaria, etc.) definiendo cualquier estructura que se disponga de acuerdo a las características que tengan los tipos de legajos a administrar. El sistema permite incorporar, buscar o consultar legajos y la documentación incluida en los mismos. Además, se implementan servicios web para consultas de información de legajos de agentes.
Estructura Organigrama Funcional (2da etapa)	Es un sistema de gestión web que ayuda a crear y definir con claridad las funciones de las diferentes unidades administrativas de la UNRN. Esta herramienta permite facilitar la coordinación institucional, dotando a la institución de un instrumento administrativo que determine la forma de la estructura orgánica básica, la misión y función de cada una de las unidades administrativas, sus relaciones de dependencia, supervisión y coordinación; así como, el detalle de las principales funciones de las unidades.
Licencias Digitales (3er etapa)	Este sistema web permite que los agentes docentes y no docentes de la UNRN puedan ingresar para generar sus licencias y avisos de ausencia a través de internet pensado como una herramienta para agilizar y tener la información disponible para la dirección de RRHH.

El desarrollo de software bajo el paradigma LPS está centrado en la reutilización o derivación de activos base. Un activo base de software puede estar compuesto por; arquitecturas, patrones de diseño, componentes, documentación, especificaciones y requisitos del sistema, planes y pruebas entre otras [5], lo que permite hacer uso de cada uno de estos aspectos para desarrollar la familia o conjunto de productos.

El concepto de línea de productos que utiliza una arquitectura base y un conjunto de elementos reutilizables es directamente aplicable a las organizaciones o instituciones, que desarrollan productos o sistemas. También es posible obtener ventajas de la puesta en marcha de una línea de productos en organizaciones más orientadas a proyectos o incluso en los departamentos de informática que dan soporte a los sistemas informáticos como es el caso de estudio presentado [2].

Según [8] el interés por las LPS surge en el campo de la reutilización del software, cuando se observa que resulta mucho más provechoso reutilizar diseños arquitectónicos completos en lugar de componentes software individuales. Es posible que pueda aplicarse un enfoque LPS e identificar los activos base, que estarán presentes en todos o en la gran mayoría de todos los productos (aplicaciones) que se implementen, a modo de ejemplo se puede mencionar algunos de los activos base comunes a los tres productos como muestra la figura 2:

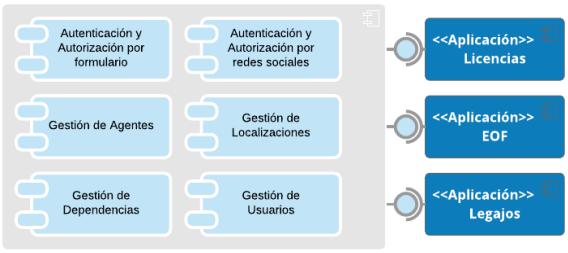

Fig.2 Activos base para los tres productos.

La etapa de Ingeniería de Dominio abarca el análisis y la identificación del ámbito de aplicación de la LPS, que incluye la captura de todo el conocimiento sobre el dominio y el negocio, a través del modelado de partes comunes y variables, que están presentes en una arquitectura de referencia, la cual es una arquitectura de software de alto nivel de abstracción, genérica y basada en un esquema instanciable o modelo de variabilidad, que dirige la derivación de productos concretos de la familia LPS.

La arquitectura de software y entornos de Trabajo (Framework) utilizados por los productos se encuentra definida en capas como muestra la figura 2. En el caso de una aplicación empresarial puede dividirse en tres capas lógicas bien definidas [9]: 1) la capa de presentación, 2) la capa de negocio y 3) la capa de persistencia. El principio para la separación en capas es que cada una esconde su lógica al resto y solo brinda puntos de acceso a dicha lógica. En la capa de presentación los objetos trabajan directamente con las interfaces de negocios, implementando el patrón arquitectónico Model-View-Controller [9]. En este, el modelo (Model) es modificable por las funciones de negocio, siendo estas solicitadas por el usuario, mediante el uso de un

conjunto de vistas (View) que solicitan dichas funciones de negocio a través de un controlador (Controller), que es quien recibe las peticiones de las vistas y las procesa. La capa de negocio está formada por servicios implementados por objetos de negocio. Estos delegan gran parte de su lógica en los modelos del dominio que se intercambian entre todas las capas. Finalmente, la capa de persistencia facilita el acceso a los datos y su almacenamiento en una base de datos.

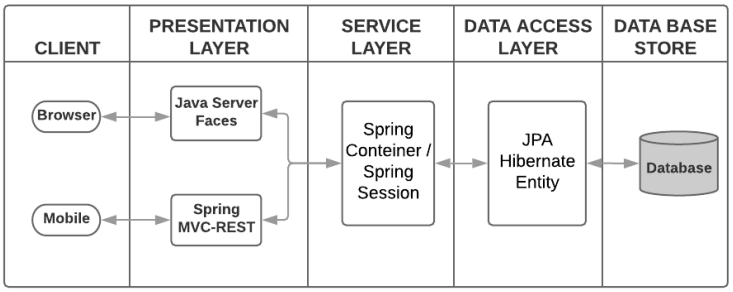

Fig.2 Arquitectura en capas.

Dada la arquitectura propuesta y los componentes comunes que se plantearon anteriormente, es posible, como estrategia, aplicar el enfoque LPS en el caso de estudio, utilizando el modelo extractivo definido en la sección 2.2.1. El enfoque extractivo comienza con un conjunto de productos o aplicaciones ya desarrolladas y de manera iterativa e incremental se refactorizan los elementos arquitectónicos. En este caso se realizó un análisis de las aplicaciones existentes, a partir de los cuales, mediante un análisis de similitud semántica de componentes, se redefinen las arquitecturas de las aplicaciones o productos a partir de la información o documentación disponible, determinándose componentes comunes y variantes.

Es importante determinar la aplicabilidad de la LPS en este contexto, bajo el proceso de diseño “bottom up” (de abajo hacia arriba), extractivo, repetible y sistemático, para construir una arquitectura LPS de referencia. [10]. La arquitectura de una LPS es una arquitectura de software genérica, representa la estructura de toda la familia de productos y no solamente la de un producto en particular. Describe los aspectos comunes y variables de una familia de productos de software.

La arquitectura LPS debe ser instanciada cada vez que se desarrolla un producto de la línea.

Una contribución importante del trabajo es definir un proceso de Ingeniería del Software hacia una LPS en el dominio del conjunto de aplicaciones en el contexto de gobierno digital con el fin de: 1- Implementar un repositorio LPS de “core assets” con componentes comunes y variables especificados a un alto nivel de abstracción; 2- diseñar la arquitectura LPS para los sub-dominios, mediante un enfoque ascendente o “bottom-up” basado en el estudio de aplicaciones existentes ya desarrolladas, y 3- definir estrategias para instanciar la arquitectura obtenida para cada sub-dominio, a partir de los “core assets” identificados como reutilizables.

4 Trabajos Relacionados

Este trabajo está basado en tareas de investigación que analizaron los aportes científicos que se encuentran en esta misma línea. En particular, se investigaron trabajos relacionados con la implementación de LPS en gobierno digital. Algunos de los trabajos más importantes se discuten a continuación.

Fajar, AN y sus colegas [11] realizaron una investigación sobre la situación del cambio dinámico en el gobierno indonesio, esta condición afecta al desarrollo de software lo que causa reemplazo, modificación y mejora. Existen algunas aspectos comunes y variables del software en el gobierno indonesio, por tanto, para gestionarlo, presentan un marco de trabajo denominado ZumaFramework (ZEF) para reducir la complejidad del mismo. Por otra parte, el autor indica que el desarrollo de sistemas en las instituciones gubernamentales es para la mejora de los procesos de negocio que pueden ayudar a hacer una buena gobernanza. Las características de similitud y variabilidad del software podrían lograrse a partir de los procesos de negocio. Esto puede relacionarse con el costo y el tiempo asignado para el desarrollo del sistema. Por esto, propusieron un framework SPLGS para desarrollar sistemas bajo el paradigma LPS en el gobierno de Indonesia [12]. En [13] la investigación muestra que las organizaciones están interesadas en la reducción de costos asociados al desarrollo de software, y es esencial utilizar herramientas automatizadas y un proceso sistemático de reutilización. El trabajo presenta un entorno de generación de CMS, en el contexto de LPS automatizada a través de frameworks, generadores de aplicaciones y repositorios de reutilización. Finalmente, [14] presenta el enfoque de Ingeniería de la Línea de Producción de Software (en inglés, Software Production Line Engineering- SPLE) en el que las organizaciones gubernamentales o universitarias sin fines de lucro y con escasos recursos, son cada vez más solicitadas para desarrollar aplicaciones personalizadas. Estas aplicaciones a menudo tienen arquitecturas y cumplen requisitos muy similares. Dadas estas características, proponen una metodología SPLE a una serie de aplicaciones.

5 Conclusiones y Trabajo Futuro

Este trabajo presentó un marco de trabajo para el desarrollo de software basado en LPS. La ventaja radica en potenciar la reutilización estratégica en el desarrollo de software en gobierno digital. La adopción de LPS implica un cambio de mentalidad en el desarrollo de software porque permite centrarnos en construir y gestionar los aspectos comunes de los productos, identificando los componentes y definiendo arquitecturas de referencia de alto nivel de abstracción. En este entorno es posible lograr una reutilización más planificada, dado que las incorporaciones de nuevas variantes de los productos se realizan de forma sistemática y controlada respecto al enfoque de desarrollo tradicional que plantea la reutilización más oportunista.

Desde la perspectiva de la Universidad que se encuentra en un proceso de transformación digital y despapelización de los procesos administrativos, es deseable contar con técnicas y metodologías de desarrollo donde pueda ser posible aplicar un enfoque de reutilización sistemática con el fin de lograr los siguientes beneficios:

utilizar los artefactos construidos previamente en cada desarrollo nuevo, aumentar la productividad de los recursos dada la semejanza entre productos y gestionar eficazmente el conjunto de productos diferentes. Por otro lado, lograr una reducción de costos en cuanto a pruebas y mantenimiento que será capitalizado por todos los productos de la línea gracias a la existencia de este marco común que ofrece la LPS.

Trabajos a futuro incluyen extender el marco de trabajo aplicando otras técnicas y metodologías de desarrollo en las que la reutilización juega un papel esencial, como la Ingeniería del Software Dirigida por Modelos (Model-Driven Engineering- MDE), este paradigma dirigido por los modelos, es decir, el sistema de modelos tiene suficiente detalle para permitir la generación de código de un sistema completo a partir de los modelos propios, esto es, construir modelos que puedan ser directamente compilados y ejecutados, "el modelo es el código" [15]. Por otro lado, explorar los Lenguajes Específicos de Dominio (Domain Specific Language- DSL) y la Programación Generativa que de manera similar al desarrollo de LPS ofrece la construcción, con un enfoque de reutilización proactivo, de familias de productos relacionados con algún dominio particular, en lugar de productos independientes.

Referencias

1. Usaola, M.: Desarrollo de software basado en reutilización. UOC. Formación de Postgrado. 2013
2. Peñalvo, F., et al.: Líneas de Productos, Componentes, Frameworks y Mecanos. Informe Técnico, Departamento de Informática y Automática - Universidad de Salamanca (2002)
3. Sodhi, J., Sodhi,P: Software reuse: Domain analysis and design process. McGraw-Hill. (1999)
4. Sametinger, J.: Software engineering with reusable components. Springer Verlag, (1997).
5. Clements,P., Northrop,L.: Software Product Lines: Practices and Patterns. Addison-Wesley (2001).
6. Díaz O., Trujillo, S.: Fábricas de Software: experiencias, tecnologías y organización" (2º edición). Editorial Ra-Ma, (2010).
7. Trujillo, S., Batory, D. y Díaz, O: Feature Oriented Model Driven Development: A Case Study of Portlets. International Conference on Software Engineering (ICSE) (2007).
8. Gomaa, H.: Designing Software Product Lines with UML. From use cases to pattern based software architectures. Addison-Wesley (2005).
9. Fowler, M. "Patterns of Enterprise Application Architecture", Addison-Wesley, (2002).
10. Losavio F., Ordaz O., Esteller V: Refactoring-Based Design of Reference Architecture, RACCIS, Vol 5, No 1, pp. 32-48, ISSN 2248-7441. (2015)
11. Fajar A.N., Shofi I.M.,"Reduced software complexity for E-Government applications with ZEF framework". TELKOMNIKA, Vol.15, No.1, pp. 415~420 ISSN: 1693-6930 (2017).
12. Fajar A.N., Shofi I.M.: Development of SPL Government System with Ontology Web Language, 4th Conference International Conference on Cyber and IT Service Management. (2016).
13. Lima V.M.A.D., Marcacini R.M., Lima M.H.P., Cagnin M.I., Turine M.A.S.: A generation environment for front-end layer in e-government content management systems. 9th Conference Latin American Web Congress. (2014).
14. Diwan P., et al: Applying software product line engineering in building Web portals for supercomputing services. Proceedings of the 28th Conference Annual ACM Symposium on Applied Computing. (2013).
15. Oquendo, F.: p-Method: A Model-Driven Formal Method for Architecture-Centric Software Engineering. ACM SIGSOFT Software Engineering Notes. Volume 31 Number 3. (2006).