LÍNEAS DE PRODUCTO SOFTWARE[1]

Óscar Díaz

oscar.diaz@ehu.es,

Facultad de Informática, Universidad del País Vasco

1. INTRODUCCIÓN

Cuando una empresa ofrece un producto software a distintos clientes, surge toda la problemática de las versiones y evolución acompasada del producto. En este escenario, no es raro encontrarse con alguna de las siguientes situaciones:

- Relaciones conflictivas entre los equipos de marketing y de ingeniería a causa de la incapacidad de los segundos para adecuarse a las solicitudes de nuevas variaciones de producto provenientes del negocio.

- Coordinación compleja y costosa de múltiples tareas de desarrollo en paralelo que comparten software común.

- Código fuente poco robusto, que resulta difícil de extender con variaciones del nuevo producto y propenso al error.

Esta problemática no es exclusiva del software. Se da también en la fabricación de otros productos como automóviles, aparatos de telefonía, electrodomésticos, y en general, cuando un mismo producto admite distintas variaciones. La producción en serie (*mass production*) —la capacidad para crear eficientemente múltiples copias del mismo producto— constituyó un gran avance en el mundo de la fabricación. Pero crear múltiples copias de un producto software es trivial. Sin embargo, la personalización en serie (*mass customization*) —la capacidad para crear eficientemente múltiples variaciones de un producto— es un importante reto tanto en la fabricación de lavadoras como en la venta de un ERP o cualquier otro producto software. La pregunta es cómo se plasma el enfoque de "personalización en serie" en el desarrollo de productos software.

---

[1] Este documento es una revisión abrevida del capítulo de Oscar Díaz y Salva Trujillo publicado en "Fábricas de Software: experiencias, tecnologías y organización" (2º edición) M. G. Piattini, J. Garzás (editores), Editorial Ra-Ma, 2010

Si miramos al sistema de producción de las lavadoras, por poner un ejemplo, la personalización en serie se consigue mediante la introducción de la cadena de montaje o línea de producto. El objetivo: sacar partido de los elementos comunes y gestionar de una manera eficaz las variaciones. En lugar de permitir la máxima flexibilidad que permite el proceso artesanal pero con grandes costes, las líneas de producto delimitan las variantes de sus productos a un conjunto preestablecido, y optimizan los procesos para estas variantes.

En la producción de software, el proceso ha venido estando centrado en el producto antes que en la línea de montaje. Las herramientas de desarrollo (IDE) y las metodologías ayudaban a agilizar y sistematizar la creación de un único producto. Sí que existe una inquietud por reutilizar pero en la mayor parte de los casos, la reutilización es oportunista, es decir, surgía la posibilidad de reutilizar a posteriori, no era algo que se supiera positivamente que se iba a poder reutilizar. Por ello, muchos esfuerzos de re-utilización no se amortizaban ya que no terminaba de surgir la oportunidad para poder reutilizarlo.

Esta situación cambia con las líneas de producto. Ahora ya no se quiere producir un único producto, sino una cadena de montaje que gestione eficiente y eficazmente las diferentes variaciones que pueden existir entre los productos. La empresa ya no se centra en un producto para un cliente (por ejemplo, construir un portal para Iberia), sino en un dominio (por ejemplo, construir portales para líneas aéreas). El reto está en delimitar el ámbito de este dominio, identificar las variaciones que se van a soportar, y dotarse de la infraestructura que permita producir el producto a bajo coste pero manteniendo altas cotas de calidad. Es decir, aplicar los principios de la producción en serie también al software. Este enfoque resulta en mejoras tanto en la eficiencia (reducción del *time-to-market*) como en la eficacia (mejora de la calidad del software). Entre los precursores de este enfoque en el mundo del software se encuentran McIllory (1968), Parnas (1976) y Neighbors (1989) que en sus trabajos ya intuían el potencial de estas ideas.

Las Líneas de Producto Software (LPS) pueden por tanto englobarse dentro de ese anhelo recurrente dentro de la Ingeniería del Software que es la reutilización. Pero nos han recordado que mejorar la reutilización no lleva necesariamente a reducir los costes globales de desarrollo debido a los costes adicionales de desarrollar (y gestionar) precisamente estos artefactos re-usables. Las LPS han vuelto a recordarnos que la reutilización eficaz no es sólo un problema técnico, sino también de procesos y organización. El proceso determina cuándo y dónde se debe realizar el esfuerzo de re-utilización. La decisión no es baladí. De hecho, muchos de los fracasos en el desarrollo basado en componentes (también orientado a la reutilización) se deben a fallos en el proceso, más que en las técnicas que se utilizaban: se invertían esfuerzos en hacer el componente reutilizable para determinadas situaciones que finalmente no se presentaban.

Así mismo, las LPS suponen una importante reorganización de los equipos humanos. De estar orientados al producto pasan a pivotar sobre la LPS. Aquí pueden surgir tensiones entre los desarrolladores de los artefactos comunes (o reutilizables) frente a aquellos encargados de desarrollar el producto con estos artefactos. ¿Qué ocurre ante nuevas peticiones del cliente no contempladas hasta entonces? Si la planificación no se cumple o se detecta algún error, ¿a qué equipo se le imputa el coste? ¿Cómo se reparten las responsabilidades/presupuestos entre los dos tipos de equipos?

Este capítulo comienza con una definición de LPS y sus beneficios potenciales, para luego centrarse en los aspectos metodológicos y técnicos que la consecución de estos beneficios conlleva. Para terminar se da la bibliografía relacionada.

2. DEFINICIÓN

La definición más comúnmente aceptada de una LPS procede de Clements (2001) 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*".

Esta definición aglutina cinco conceptos fundamentales de las LPS, a saber:

- "…un conjunto de sistemas software…": el objetivo de una LPS no es el desarrollo de un producto, sino el de un conjunto de productos. Hay que delimitar y preestablecer en la medida de lo posible el alcance de este conjunto (scoping).

- "…que comparten un conjunto común de características (features)…": este conjunto de productos se caracteriza por medio de características (features). Una característica es una peculiaridad del producto que los clientes consideran importante para describir y distinguir entre los distintos miembros de la LPS. Este enfoque es muy común en otros ámbitos. Así cuando vamos a un concesionario, el modelo de coche deseado lo identificamos por medio del color, tipo de motor, existencia de techo panorámico, tipo de tapicería, etc. En el caso de los ordenadores, estamos acostumbrados a caracterizarlos en términos del procesador, las dimensiones de la pantalla, etc. Todas ellas serían características que nos sirven para distinguir entre productos, y que son significativas para el usuario final. Se han documentado casos de LPS con más de 10.000 características. En estos casos es fundamental disponer de algún modelo que nos ayude a organizar las características, y especificar sus dependencias. Son los modelos de características.

- "…satisfacen las necesidades específicas de un dominio o segmento particular de mercado…": una LPS se desarrolla orientándose a un segmento de mercado concreto. Es decir, los productos intentan satisfacer las necesidades específicas de un segmento de mercado. De la habilidad para acotar e identificar correctamente este mercado, dependerá el éxito de la LPS.

- "…se desarrollan a partir de un sistema común de activos base (core assets)…": los productos son desarrollados a partir de un conjunto común de activos reutilizables (core assets). Este término engloba la diversidad de

elementos, tales como requisitos, planificaciones, modelo de características, arquitecturas, componentes, código fuente, descriptores, etc., que conforman la base sobre la que se construye el producto. El reto está en determinar no sólo lo común sino también lo que se va a permitir variar.

- "…de una manera preestablecida.": los productos se construyen de una forma preestablecida. Es decir, no sólo hay unos elementos comunes (core assets), sino que la estrategia para construir el producto está perfectamente establecida de antemano a través de un plan de producción.

Estos cinco conceptos forman la esencia de lo que se entiende actualmente por desarrollo basado en LPS. A continuación, se introducen los beneficios derivados del uso de LPS.

3. BENEFICIOS RELATIVOS A LA PRODUCTIVIDAD Y AL COSTE

Las LPS pueden incrementar significativamente la productividad de los ingenieros de software, entendida como una reducción en el esfuerzo y el coste necesario para desarrollar, poner en marcha y mantener un conjunto de productos software similares. En los casos de estudio se han observado mejoras en la productividad que duplican o triplican los enfoques tradicionales.

La figura 3.1 ilustra la distinta evolución en los costes entre el desarrollo convencional y las LPS. A mayor número de productos, mayor esfuerzo y coste. En un entorno convencional, un enfoque muy común es el "*copy&own*". Se realiza una copia de una aplicación previa que se parece a la actual, y a partir de aquí la copia evoluciona de forma totalmente independiente. Aquí el parecido entre las dos aplicaciones sirve para agilizar el desarrollo, pero en ningún caso el mantenimiento. Cada aplicación se mantiene de forma independiente, incluso si este mantenimiento es similar también para ambas. Si el mantenimiento es correctivo, habrá que depurar el código en las dos aplicaciones. Si el mantenimiento es perfectivo, habrá que realizar las mejoras en las dos aplicaciones. Resultado: no hemos explotado esta semejanza para reducir los costes de mantenimiento que es el auténtico agujero negro de nuestros desarrollos. Aquí la reutilización es oportunista, y la gestión de la semejanza (partes comunes y partes variables) es secundaria.

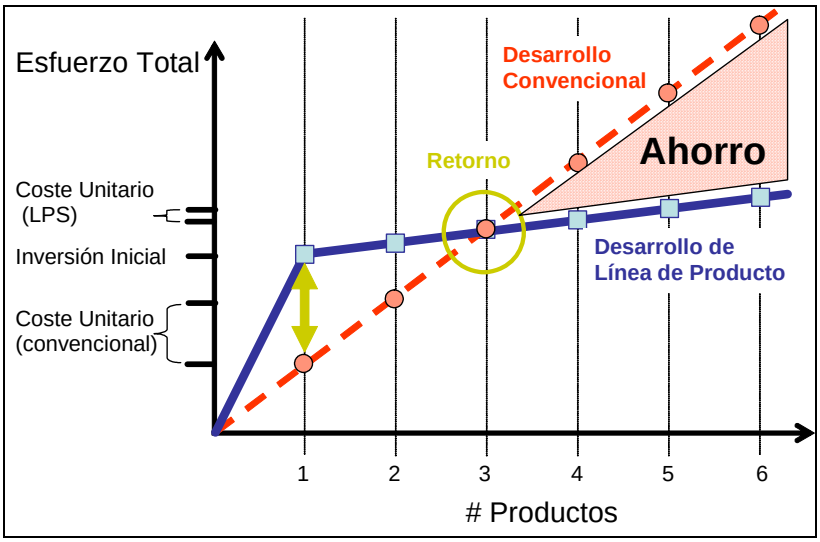

Figura 3.1. Desarrollo Convencional vs. Línea de Producto

Por tanto, el entorno tradicional tiende a centrarse en el producto: cada producto tiene su mantenimiento y los equipos humanos también tienden a fragmentarse de esta forma. No aprovechamos las potenciales sinergias que se podrían derivar de la semejanza entre productos, y el número de productos diferentes que podemos gestionar eficazmente es muy limitado.

Por el contrario, un entorno de LPS está pensado expresamente para gestionar lo común, y su complementario, lo variable. La reutilización ya no es oportunista, sino planificada, y la incorporación de nuevas variantes se realiza de forma sistemática y controlada. Esto agiliza no sólo el desarrollo del producto y su puesta en el mercado (*time-to-market*) sino también el mantenimiento. Ahora los esfuerzos de mantenimiento realizados en la LPS son capitalizados por todos los productos. Un error detectado en un producto puede ser relativamente fácil de corregir en todos los productos de la línea gracias precisamente a la existencia de este marco común que ofrece la LPS.

Esta reducción en el coste de mantenimiento de un producto permite abarcar un mayor número de productos, o de variantes dentro de la LPS. Esta capacidad para escalar a un número mayor de productos comporta ventajas estratégicas, a saber: cubrir un número mayor de mercados con productos adaptados; crear un número mayor de productos enfocados a segmentos más específicos de mercado; incrementar la agilidad para expandirse a nuevos mercados y reducir el riesgo asociado a la puesta en marcha de los productos.

La figura 3.1 fue introducida inicialmente por Weiss (1999) e ilustra el esfuerzo —y por tanto el coste— necesario para desarrollar, poner en marcha y mantener un conjunto de productos software similares. En el enfoque tradicional no se obtienen ahorros

importantes por el hecho de ser productos semejantes. El esfuerzo asociado al producto 6 es similar al esfuerzo que conlleva el producto 4. Quitando la ventaja del "*copy&own*", cada producto evoluciona de manera independiente, y es posiblemente gestionado por equipos independientes.

Por el contrario, el enfoque de LPS realiza inicialmente un estudio para analizar el dominio, determina las variaciones a soportar, y desarrolla la plataforma de LPS. Antes siquiera de realizar el primer producto, tenemos este coste de entrada (inversión inicial) cuya cuantía dependerá del modelo de proceso que utilicemos al desarrollar la plataforma de LPS (ver sección 3.5.1). La intersección entre las dos líneas de la figura 3.1 muestra el punto a partir del cual se empieza a rentabilizar esta inversión en la LPS (*return-of-investment*). A partir de aquí, la generación de un producto es menos costosa con la LPS que con el enfoque tradicional. Nótese, sin embargo, que si la familia de productos es poco numerosa tal vez no compensen los gastos iniciales de desarrollo de la plataforma. Estudios recientes de Bockle et al. (2004) sugieren que el punto de retorno puede situarse alrededor de tres productos.

4. BENEFICIOS RELATIVOS A LA CALIDAD

Los beneficios que las LPS aportan a la calidad se pueden medir de dos formas. La primera mediante el grado de precisión con que cada producto se ajusta a las necesidades de cada cliente. Esta medida depende del grado de "variabilidad" de la LPS. A mayor variabilidad, más probabilidades de adaptar el producto a los gustos del cliente. Pero, normalmente, esta variabilidad tiene un coste, y el reto es encontrar el equilibrio entre coste y variabilidad. A diferencia de los enfoques tradicionales, en las LPS la variabilidad es un concepto nuclear. Todo el proceso de desarrollo está guiado por esta noción con el objetivo de abaratar los costes de la variabilidad, y así poder conseguir mayores cotas de variabilidad y, por tanto, de satisfacción de las peculiaridades del cliente.

Otro segundo aspecto es la tasa de defectos en los productos de la LPS. Aquí los beneficios se derivan de la reutilización de los elementos comunes *(core assets).* La continuada utilización de estos elementos a lo largo del tiempo hace que finalmente estén muy depurados/probados. Además, los beneficios de encontrar y eliminar un defecto en un *core asset* no se limitan al producto donde se detecta el error, sino que se disemina entre todos los productos de la LPS.

Un ejemplo de esta situación nos lo proporciona el caso de estudio de la LPS de la empresa Salion que es descrita por Clements (2001) y que describe la figura 3.2. Esta figura ilustra dos aspectos. Primero, la tendencia descendente del número de defectos al testear un conjunto de productos A, B, C y D de forma secuencial. Aquí, los errores detectados en A son corregidos y re-alimentados en la LPS, evitando así este tipo de errores en los productos que le siguen. Otro segundo aspecto del mismo fenómeno se produce al comparar diferentes lanzamientos de los mismos productos. Aquí, el segundo lanzamiento del producto A se beneficia de los errores detectados no sólo durante el lanzamiento previo de A, sino también de todos los productos de la LPS.

Esta disminución gradual de los errores es factible gracias a la existencia de la LPS que ofrece el marco para que esta labor correctiva sea ágil y efectiva.

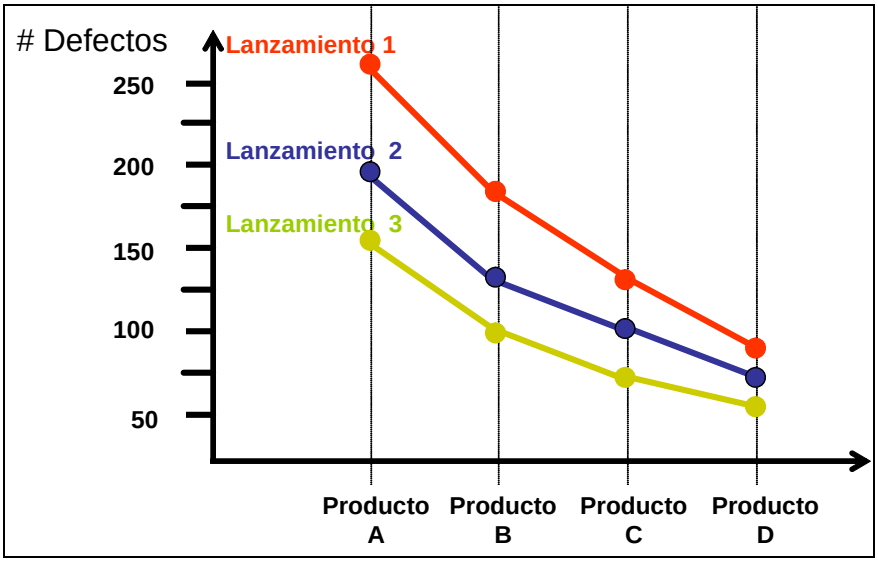

Figura 3.2. Defectos en LPS

5. ASPECTOS METODOLÓGICOS

5.1. Estrategias

El proceso de desarrollo de la LPS depende, entre otros muchos factores, del ámbito de la LPS. Es fundamental saber acotar la familia de productos que serán objeto de la línea. En general, existe una tendencia a generalizar en exceso cuando se está desarrollando software re-usable, considerando casos poco probables. Es la filosofía del "por si acaso". Sin embargo, esta excesiva generalización, si se repite con distintas "features" compatibles entre sí, puede dar lugar a una explosión combinatoria. Así cuatro features que soportaran cada una de ellas tres casos posibles, todos ellos compatibles entre sí, daría lugar a 36 posibles diferentes combinaciones (ver Benavides, 2005). Esto favorece la variabilidad, pero incurre en costes de pruebas, documentación y desarrollo adicionales, que pueden finalmente no rentabilizarse para casos poco probables.

La pregunta es entonces cuánta variabilidad soportar, qué grado de re-uso van a soportar los elementos comunes (*core assets*), es decir, delimitar el ámbito de la línea de producto.

La primera generación de líneas de producto, desarrollada en Nokia o Philips utilizando metodologías como PuLSE o STARTS, recomendaba fijarse un horizonte temporal de cinco años. Es decir, realizar un estudio de mercado y de posibles productos que no abarcara más allá de los próximos cinco años. Es el denominado modelo **proactivo**.

Este horizonte temporal puede ser adecuado para empresas grandes con una posición de mercado dominante, pero resulta un reto de planificación para empresas de tamaño medio cuya competitividad se basa principalmente en la agilidad con la que son capaces de reaccionar en un contexto mucho más cambiante. Es por ello que empresas como Salion recomiendan planificaciones a 6/12 meses vista, e ir introduciendo nuevas *features* conforme el mercado las va demandando (nunca antes). Es el denominado modelo **reactivo**.

Esta importante decisión, es decir, el ámbito inicial de la línea de producto, depende del punto de partida, de la experiencia acumulada en proyectos anteriores y de la posición en el mercado que ocupe la empresa en cuestión. El enfoque a largo plazo recibe el nombre de proactivo ya que intenta adelantarse a las necesidades existentes por ejemplo dentro de cinco años. Por otro lado, el enfoque reactivo va acometiendo los cambios conforme se van necesitando.

Un enfoque intermedio es el denominado 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 farragoso mantenimiento les impulsa a embarcarse en las LPS.

El modelo utilizado (proactivo, reactivo, extractivo) condiciona tanto el desembolso inicial de puesta en marcha de la línea de producto como la metodología de desarrollo. Respecto al primero, conforme aumenta el horizonte temporal y el número de *características* consideradas, la infraestructura de la línea de producto necesaria para producir el primer producto se vuelve más compleja. Esta situación se plasma en la figura 3.3.

Por contra, el enfoque reactivo (figura 3.3) es un enfoque a “trompicones”: restringe el número de *características* iniciales por considerar y va paulatinamente introduciendo nuevas *características* conforme las va necesitando. Aquí el coste de entrada es menor pero para un número de *características* final alto, eleva el coste total de desarrollo de la línea de producto, si bien este coste se va repartiendo en el tiempo.

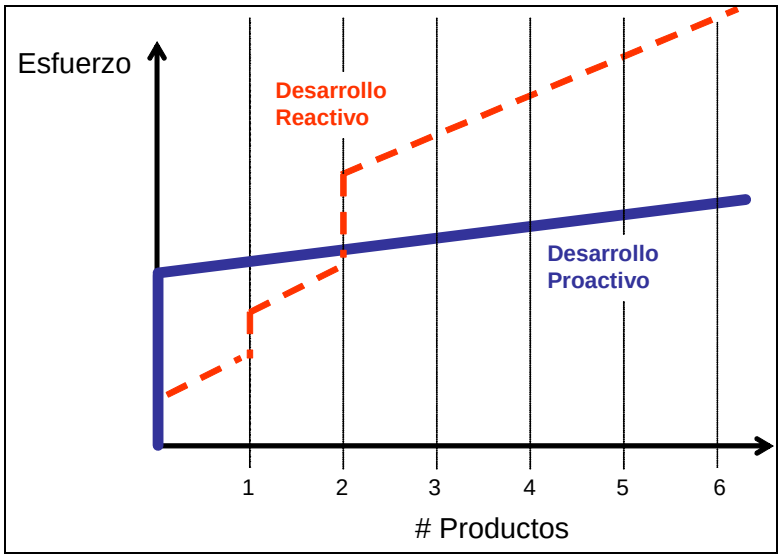

Figura 3.3. Tipos de Desarrollo

5.2. Procesos

Un aspecto central compartido por las distintas metodologías de desarrollo de LPS es la división de los procesos de ingeniería en dos equipos de trabajo (ver figura 3.4). El primer equipo se encarga de **la Ingeniería de Dominio**, el cual es definido por Clements (2001) como *core asset development*. Este equipo es responsable de desarrollar los elementos comunes al dominio: estudiar el dominio, definir su alcance (requisitos) dentro del mercado objetivo de la LPS, definir las *features*, implementar los *core assets* reutilizables y su mecanismo de variabilidad, y establecer cómo es el plan de producción.

El segundo equipo se encarga de **la Ingeniería de Producto** definido por Clements (2001) como *product development*. Sus cometidos incluyen desarrollar los productos para clientes concretos, a partir de los recursos basados no en los requisitos del dominio, sino en requisitos concretos de clientes. Para ello, este segundo equipo utiliza los recursos creados por el equipo anterior. La figura 3.4 muestra gráficamente el proceso de ingeniería de producto y su relación con la ingeniería de dominio.

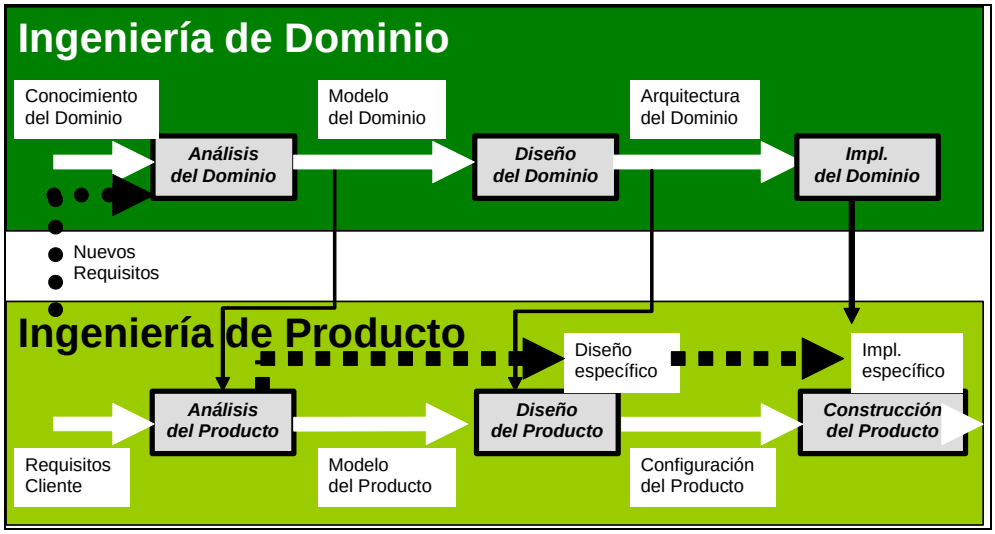

Figura 3.4. Procesos en Líneas de Producto

El objetivo de la ingeniería de dominio es establecer la capacidad de producción que posteriormente se utilizará en la producción de productos específicos para clientes. Esta actividad es iterativa, ya que se re-alimenta en el futuro de la ingeniería de producto para ir evolucionando la capacidad productiva. Es decir, en todo momento, la ingeniería de dominio se retroalimentará de la información que recoja el equipo de producto durante la producción de productos.

Esto significa que durante la ingeniería de dominio se recogen iterativamente los requisitos comunes para toda la LPS. Estos requisitos se expresan típicamente en forma de *features* (que ayudan no sólo a que los clientes puedan distinguir unos productos de otros, sino que dirigen así mismo el desarrollo de código que implemente también esas características). Este proceso incluye el estudio para conocer el dominio en cuestión al que se dirige la LPS. El conocimiento de este dominio será una parte fundamental del análisis de dominio (*domain analysis* introducido por Kang en 1990).

Un enfoque clásico de esta actividad implica no sólo analizar el dominio, sino diseñar los artefactos del dominio e implementarlos. Por lo tanto, en base a los requisitos establecidos previamente se realizará el diseño genérico de la LPS que producirá eventualmente una arquitectura del dominio. A continuación, se implementarán las partes de código que soporten esta arquitectura. Estos elementos serán reutilizados durante la construcción de productos.

Sin embargo, un enfoque más reciente implica concentrarse en tres actividades: estudiar el alcance de la línea de producto, desarrollar los elementos comunes (*core assets*) y crear un plan de producción que articule toda la capacidad productiva, y que sirva de guía durante el proceso de ingeniería de dominio (ver Clements, 2001). Mientras en el enfoque anterior ambos procesos eran similares, con este enfoque la ingeniería de dominio tiene actividades diferentes respecto de la ingeniería de producto. El proceso de

ingeniería de dominio no sólo analiza el dominio, sino que se concentra en crear la capacidad de producción para la ingeniería de producto.

La ingeniería de producto utiliza la infraestructura creada por el equipo de dominio para construir productos concretos para clientes que lo demanden. Por lo tanto, se partirá de unos requisitos concretos que deben de estar alineados de alguna forma con los requisitos de la línea de producto, aunque puede haber una pequeña parte nueva y que hay que desarrollar para este producto. El estudio de estos requisitos producirá un análisis detallado con la viabilidad de su construcción, así como la cantidad de elementos que podemos reutilizar. El siguiente paso consistirá en el diseño específico del sistema (con las partes comunes junto con las nuevas). A continuación se utilizará el configurador de producto que (en base a los requisitos seleccionados) ensamblará los elementos comunes y variables para producir el producto final que se entrega al cliente.

Durante el desarrollo del producto, las principales tareas son las de configuración y las de composición. Las de configuración "instancian" los elementos comunes, resolviendo los puntos de variación. Aquí se pueden utilizar mecanismos como los *#ifdefs*, o bien, técnicas manuales donde se rellena con código plantillas de funciones ("stubs"). Por otro lado, la composición implica la escritura de código alrededor de estos elementos comunes ("glue code") para facilitar su interconexión.

Esta división formal entre dominio y producto es complementada por algunos autores como Clements (2001) con una parte dedicada a la gestión (*management*). Esta actividad juega un papel fundamental en el éxito de la línea de producto, ya que es no sólo la encargada de asignar recursos, sino de coordinarlos y supervisarlos. Esta gestión se realiza tanto en el nivel técnico (gestión de proyectos concretos) como en el nivel organizativo (estructura organizativa adecuada a los objetivos propuestos). Los artefactos de gestión creados como planificaciones también forman parte de los elementos comunes (*core assets*).

Todas estas consideraciones generales se ven luego matizadas dependiendo del tipo de estrategia seguida (reactiva, proactiva, extractiva). Metodológicamente, el enfoque proactivo se equipararía con los métodos en cascada (*waterfall*) utilizados en los desarrollos convencionales, mientras que el enfoque reactivo encontraría su equivalente en los métodos ágiles.

En el caso del enfoque reactivo uno de los retos radica en sincronizar la planificación del desarrollo del producto con la de la propia LPS que se va construyendo de forma gradual conforme van surgiendo las necesidades. Pueden además surgir problemas asociados al mantenimiento, y la evolución de la LPS. Al convivir dos equipos diferenciados, los ingenieros del dominio y los ingenieros de la aplicación, existe el riesgo de que esta evolución se haga de forma independiente bien sólo con los elementos comunes, bien limitando la evolución a un producto concreto sin explotar la posibilidad de reutilización en otros productos.

7. CONCLUSIONES

Este capítulo introduce la noción de Línea de Producto Software como una etapa más en la búsqueda del equilibrio entre coste y calidad del software. La estructura del capítulo trata de resaltar las tres bandas en las que tiene que jugar este enfoque: la organizativa, la metodológica y la técnica. De un buen maridaje entre ellas dependerá en gran medida el éxito final que obtenga nuestra línea de producto.

Si bien las ideas aquí expuestas fueron expuestas hace ya más de 30 años, no es hasta hace poco más de diez años que estas ideas se han evaluado en casos reales de cierta envergadura. Los resultados han sido muy esperanzadores, y los trabajos para profundizar en modelos de organización, metodologías y técnicas adaptadas a las LPS son temas de interés creciente. Así lo avala la existencia de números monográficos sobre LPS en diversas revistas de divulgación científica, y la inclusión de este tema en las principales conferencias y programas de I+D.

Además de la bibliografía que acompaña a este capítulo (y que rápidamente quedará obsoleta), aconsejamos las siguientes URL:

- http://www.sei.cmu.edu/productlines/plp_hof.html. Aquí encontrarán lo que se denomina el "Product Line Hall of Fame", una iniciativa de los impulsores de la SPLC (Software Product Line Conference) para divulgar las mejores prácticas de líneas de producto en casos reales.

- http://www.softwareproductlines.com/. Este sitio está soportado por BigLever, una empresa especializada en temas de LPS. Tiene una excelente introducción al tema. Concretamente, el apartado de beneficios de este capítulo se ha elaborado a partir del material de esta Web.

- http://www.sei.cmu.edu/productlines/index.html. El SEI (Software Engineering Institute) ofrece en esta página abundante información sobre un tema donde llevan muchos años trabajando.

BIBLIOGRAFÍA

ACM. (2006). ACM Communications. *Special Issue on Software Product Lines* (Número especial sobre Líneas de Producto Software). 49(12), diciembre.

Anastasopoulos, M. y Muthig, D. (2004). An Evaluation of Aspect-Oriented Programming as a Product Line Implementation Technology. *International Conference on Software Reuse* (ICSR).

Apel, S., Leich, T. y Saake G. (2006). Aspectual Mixin Layers: Aspects and Features in Concert. *International Conference on Software Engineering* (ICSE).

Batory, D., Neal, J. y Rauschmayer, A. (2004). Scaling Step-Wise Refinement. *IEEE Transactions on Software Engineering* (TSE). 30(6), pp. 355-371.

Benavides, D., Ruiz-Cortés, A. y Trinidad, P. (2005). Automated Reasoning on Feature Models. *International Conference on Advanced Information Systems Engineering* (CAiSE).

Böckle, G., Clements, P., McGregor, J. D., Muthig, D. y Schmid, K. (2004). Calculating ROI for Software Product Lines. *IEEE Software*, 21(3), pp. 23-31, mayo/junio.

Bracha, G. y William, R. C. (1990). *Mixin-based Inheritance*. OOPSLA/ECOOP.

Capilla, R. (2006). Context-aware Architectures for Building Service-Oriented Systems. *European Conference on Software Maintenance and Reengineering* (CSMR).

Clements, P. y Northrop, L. (2001). *Software Product Lines: Practices and Patterns*. Addison-Wesley.

Czarnecki, K. y Eisenecker, U. (2000). *Generative Programming*.

Gacek, C. y Anastasopoulus, M. (2001) Implementing product line variabilities. *Symposium on Software Reusability* (SSR).

Jarzabek, S., Bassett, P., Zhang, H. y Zhang, W. (2003). *XVCL: XML-based Variant Configuration Language*. ICSE. http://xvcl.comp.nus.edu.sg/xvcl_tutorial.php

Kang, K. C., et al. (1990). Feature Oriented Domain Analysis Feasability Study. *Technical Report CMU/SEI-90-TR-21, Software Engineering Institute*.

McIlroy, M. D. (1968). Mass Produced Software Components. *NATO Science Committee*.

Parnas, D. (1976). On the Design and Development of Program Families. *IEEE Transactions on Software Engineering* (TSE), 2(1), pp. 1-9, marzo.

Pohl, K., Böckle, G. y van der Linden, F. (2005). *Software Product Line Engineering: Foundations, Principles, and Techniques*.

Trujillo, S., Batory, D. y Díaz, O. (2006). Feature Refactoring a Multi-Representation Program into a Product Line. *Generative Programming and Component Engineering Conference* (GPCE).

Trujillo, S., Batory, D. y Díaz, O. (2007). Feature Oriented Model Driven Development: A Case Study of Portlets. *International Conference on Software Engineering* (ICSE).

Weiss, D. M., y Robert Lai, C. T. (1999). *Software Product-Line Engineering: A Family-Based Software Development Process*.