UNIVERSIDAD POLITECNICA DE VALENCIA

DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Un Método para la Evaluación de la Calidad de Líneas de Productos Software basado en SQuaRE

Tesis de Máster en Ingeniería del Software,
Métodos Formales y Sistemas de Información
(ISMFSI)

Grupo de Ingeniería del Software y Sistemas de Información (ISSI)
Departamento de Sistemas Informáticos y Computación (DSIC)
Universidad Politécnica de Valencia (UPV)

Diciembre 2009

Sonia Montagud Gregori

Directora:
Dr. Silvia Mara Abrahão Gonzales

Un Método para la Evaluación de la Calidad de Líneas de Productos Software basado en SQuaRE

Impreso en Valencia, España
Diciembre 2009

A aquellas personas
que han estado a mi lado,
que han confiado en mí,
que me han apoyado.

Gracias.

Resumen

La aproximación al desarrollo de Líneas de Productos Software se basa en la reutilización sistemática y a gran escala del desarrollo de artefactos tales como la arquitectura, diseño, y componentes, entre un conjunto de productos con funcionalidades similares. Esta aproximación promete, entre otras cosas, acortar el tiempo del desarrollo de los sistemas software y reducir significativamente los costes de desarrollo y mantenimiento. Para lograr las mejoras prometidas, los componentes y artefactos destinados para la reutilización deben ser de alta calidad. Por lo tanto, el aseguramiento de la calidad en las líneas de productos es aún más importante que en el desarrollo de software tradicional debido a que un error o una decisión de diseño inadecuada podrían propagarse a varios productos de la familia.

Sin embargo, el aseguramiento de la calidad en las líneas de productos software es más complejo debido a las propiedades intrínsecas de las mismas, como la generalización de componentes software o la gestión de la variabilidad. Hasta ahora la investigación en el ámbito de las líneas de productos software se ha centrado principalmente en el análisis, diseño e implementación de las mismas. En particular, los desafíos del aseguramiento de la calidad que surgen en este contexto no han sido abordados suficientemente, y hay poca orientación para las organizaciones que utilizan líneas de productos sobre cómo asegurar de manera sistemática la calidad de sus líneas de productos y artefactos reutilizables.

Para cubrir la escasez de métodos y técnicas, se ha propuesto un modelo de calidad específico para líneas de productos software con características, subcaracterísticas, atributos de calidad y métricas que permiten evaluar la calidad de todos los artefactos obtenidos en el ciclo de vida de una línea de productos software. Además, se ha propuesto un método de evaluación acorde al estándar de calidad ISO/IEC 25000 (SQuaRE) para guiar la aplicación del modelo de calidad propuesto. A diferencia de los métodos de evaluación tradicionales, el modelo de calidad interactúa con el resto de vistas del sistema siendo éste un artefacto activo durante el desarrollo de la línea de productos.

Resum

L'aproximació de desenvolupament de Línies de Productes de Programari es basa en la reutilització sistemàtica i a gran escala de determinats artefactes com són l'arquitectura, el disseny i els components, entre un conjunt de productes amb funcionalitats similars. Aquesta aproximació és força prometedora perquè, entre altres coses, s'espera que permeta acurtar el temps de desenvolupament del sistemes de programari, així com també reduir de forma significativa el cost de desenvolupament i manteniment posterior. Per aconseguir aquestes millores, els components i artefactes destinats a la reutilització han de ser d'alta qualitat. Per tant, donat que un error o una mala decisió de disseny faria que es propagaria l'error a diversos productes d'una família, tenint un efecte amplificador de l'errada, assegurar la qualitat en les línies de productes és encara més important que en les aproximacions tradicionals al desenvolupament de programari.

No obstant això, l'assegurament de la qualitat en les línies de productes de programari és més complex com a conseqüència de les seues propietats intrínseques, com són la generalització de components de programari o la gestió de la variabilitat. Fins ara, la recerca en l'àmbit de les línies de productes de programari ha estat centrada principalment a l'anàlisi, disseny e implementació d'aquestes. En particular, els reptes d'assegurar la qualitat que sorgeixen en aquest context no han sigut abordats suficientment, i hi ha poca orientació per a les organitzacions que empren línies de productes per tal que puguen assegurar de manera sistemàtica la qualitat de les seues línies de productes i artefactes reutilitzables.

Per tal de cobrir la manca de mètodes i tècniques, es proposa en aquesta tesina de màster un model de qualitat específic per a línies de productes de programari amb característiques, subcaracterístiques, atributs de qualitat i mètriques que permeten avaluar la qualitat de tots els artefactes obtinguts al llarga del cicle de vida d'una línia de productes de programari. A més a més, s'ha proposat un mètode d'avaluació conforme a l'estàndard de qualitat ISO/IEC 25000 (SQuaRE) per tal de guiar l'aplicació de model de qualitat proposat. A diferència dels mètodes tradicionals, el model de qualitat presentat interactua amb la resta de vistes del sistema, passant aquest a ser un artefacte actiu durant el desenvolupament de la línea de productes.

Abstract

The Software Product Line development approach is based on the systematic and high-scale reuse of software artifacts including architecture, design, and components, from a set of products with similar functionality. This approach promises to significantly reduce the software system development time and maintenance costs. However, in order to be able of reaching these benefits, the reusable components and artifacts should be of high quality.

Hence, quality assurance in software product lines is even more important than in traditional software due to the fact that an error or bad design decision could propagate to other products of the family. Nevertheless, quality assurance for software product lines is more complex due to its intrinsic properties, such as the generalization of software components and variability management. Nowadays, research works in software product lines are focused on the analysis, design and implementation of the product family. In particular, challenges in quality assurance for software product lines had not been sufficiently explored, and poor guidance is offered to software organizations to improve the quality of their reusable artifacts and product lines.

In order to fulfill this lack of methods and techniques, we propose a specific quality model for software product lines which is organized in characteristics, sub-characteristics, quality attributes, and measures. This quality model allows us to evaluate the quality of every artifact obtained during the whole product line lifecycle. In addition, an evaluation method has been proposed according to the quality standard ISO/IEC 25000 (SQuaRE) which is capable to guide the use of the proposed quality model.

Contenido

Índice de Figuras

Índice de Tablas

Acrónimos

IEC	International Electrotechnical Commision
ISO	International Organization for Standardization
LPS	Líneas de Productos Software
SQuaRE	Software product Quality Requirements and Evaluation
UML	Unified Modeling Language
VCS	Vista de la Calidad del Sistema
VFS	Vista Funcional del Sistema
VVS	Vista de la Variabilidad del Sistema

Capítulo 1

Introducción

Este capítulo de introducción se centra en la presentación de las principales motivaciones que han llevado a la realización de esta Tesis de Máster y de sus objetivos. De esta forma, la estructura de este capítulo es la siguiente. En primer lugar se describe cuál es la motivación de este trabajo de investigación. En segundo lugar se presentan los objetivos del trabajo. En tercer lugar se describe el método de investigación empleado. En cuarto y último lugar, se presenta brevemente cada uno de los capítulos restantes.

1.1. Motivación del trabajo

La producción de software de calidad, en el tiempo adecuado y con unos costes razonables sigue siendo un problema abierto de la Ingeniería del Software que ha sido abordado desde distintas aproximaciones. Una aproximación industrial al problema consiste en usar Líneas de Producto Software (LPS). Una LPS es un conjunto de sistemas software que comparten un conjunto de características comunes que satisfacen las necesidades específicas de un segmento de mercado particular y que son desarrollados a partir de un conjunto de activos comunes de un modo preestablecido [15]. El desarrollo de LPS consta de dos procesos básicos: la *Ingeniería del Dominio*, donde se establece cuales son las partes comunes y las variables y se construye un conjunto de activos como partes de los sistemas, y la *Ingeniería de la Aplicación*, donde los activos son reutilizados sistemáticamente para derivar productos específicos. De este modo se reducen costes y tiempo de desarrollo.

Las LPS requieren conocer en fases tempranas de su desarrollo los requisitos que deben cumplir los diferentes productos que se construirán a partir de la LPS (requisitos funcionales y de calidad). Una de las tareas más complicadas durante el proceso de derivación es conocer los atributos de calidad requeridos. Un atributo de calidad es *una propiedad física o abstracta de un artefacto producido durante el desarrollo de la LPS*. Mientras que la calidad es un factor importante en la construcción de productos software individuales, en la ingeniería de líneas de producto software cobra aún más importancia debido a que la calidad de todos los productos de la línea debe ser asegurada.

El desarrollo de LPS tiene características que lo distinguen del desarrollo de productos individuales. En concreto, la variabilidad, la reusabilidad, la comunalidad, o la composicionabilidad son conceptos específicos de este enfoque para la producción de software. Incluso, el modelo de ciclo de vida de un producto desarrollado mediante LPS es diferente al desarrollo de un producto único. De ahí que no sea posible reutilizar con facilidad los métodos y técnicas de evaluación de la calidad propuestos para productos individuales.

En los últimos años se han propuesto muchos métodos para el desarrollo de LPS, pero muy pocos incluyen actividades de evaluación de calidad: ATAM [45], QADA [53], QUASAR [66], QASAR [11], PuLSE-DSSA [6] y HoPLAA [57]. La revisión de estos métodos pone de manifiesto la necesidad de métodos de evaluación y aseguramiento de la calidad que permitan capturar de manera precisa todos los atributos de calidad relevantes del dominio así como los atributos relevantes para ciertos productos de la familia (ya que en una línea de productos los atributos de calidad también pueden variar debido a que no todos los

productos de la familia requieren el mismo nivel de usabilidad, mantenibilidad, seguridad, etc.

Además, la mayoría de los métodos de evaluación existentes no soportan las características específicas de las líneas de producto software (variabilidad, composicionalidad, etc.), se centran en atributos de calidad individuales (por ejemplo QADA sólo tiene en cuenta la integridad y la evolución) y no dan un soporte adecuado para evaluar los atributos del dominio.

Aparte de los métodos de evaluación, en la literatura también podemos encontrar distintos enfoques para tratar la variabilidad de los atributos de calidad: Kuusela [51] propone utilizar un árbol AND lógico; Capilla [14] extiende FODA para introducir parámetros de calidad de servicios (QoS); Zhang [72] plantea utilizar una red bayesiana para modelar la variabilidad en los atributos de calidad; González-Baixauli [33] sugiere adaptar el análisis orientado a metas a las LPS; Sinnema [63] presenta la posibilidad de utilizar el Framework COVAMOF de modelado de la variabilidad en LPS e introducir en él los aspectos relativos a la calidad; Benavides [8] extiende el modelo de características para trabajar con características extra-funcionales; Jarzabek [43] sugiere el modelo F-SIG (Feature-softgoal interdependency graph) construido a partir del modelo de características y un grafo de interdependencias orientado al cumplimiento de ciertas metas; Etxeberria et al. [22] utilizan un modelo de características extendido que mezcla requisitos funcionales con atributos de calidad, tomando en consideración la variabilidad en los atributos de calidad; finalmente, Gallina [28] plantea un análisis semántico de las partes variables y comunes de las propiedades ACID (Atomicity, Isolation, Durability and Consistency).

El análisis de estos trabajos apunta que aunque existan varias propuestas para tratar la variabilidad de los atributos de calidad, muchas no han sido integradas en un método de evaluación para ser aplicadas durante todo el ciclo de vida de una LPS. Por lo tanto, existe la necesidad de métodos sistemáticos de aseguramiento de calidad para líneas de producto software que por un lado (1) sean capaces de capturar de manera precisa todos los atributos de calidad relevantes del dominio, así como los atributos relevantes para ciertos productos de la familia, que (2) proporcionen mecanismos que permitan medir los atributos de calidad capturados y que (3) guíen el evaluador en la evaluación de los distintos artefactos obtenidos durante el ciclo de vida de la LPS (activos, arquitectura, producto final).

Las métricas de software se consideran un instrumento apropiado para medir los atributos de calidad ya que permiten entender, monitorizar, controlar, predecir y testar el desarrollo y el mantenimiento de proyectos de software. En los últimos años, se han propuesto una gran variedad de métricas para medir atributos concretos de las líneas de producto software

pero desafortunadamente la mayoría de éstas no se utilizan en la práctica y el estado actual de la industria es todavía inmaduro. Esto puede ser debido a diferentes causas, como la ausencia de información acerca de la idoneidad y limitaciones de estas métricas para asegurar la calidad de las LPS o que estas métricas no han sido propuestas en el marco de un método de medición sistemático.

El estándar de calidad ISO/IEC 25000 Software product Quality Requirementes and Evaluation (SQuaRE) [40] argumenta la relevancia de los atributos y métricas en la evaluación y aseguramiento de la calidad del software y propone un modelo de calidad constituido básicamente de características, atributos y métricas. Un modelo de calidad es una formalización del concepto de calidad y se basa en la descomposición jerárquica de este concepto. Sin embargo, el modelo es genérico para cualquier producto software y no recoge las características propias de las LPS.

1.2. Objetivos e hipótesis

Este trabajo tiene como objetivo la definición de un método de evaluación de calidad específico para LPS que capture de manera precisa todos los atributos de calidad (relevantes del dominio y específicos para ciertos productos de la familia), proporcione mecanismos para medir dichos atributos y guíe el evaluador en la evaluación de la calidad de los distintos *productos* (activos, arquitectura, producto final) obtenidos durante el ciclo de vida de una línea de productos software.

Además, esta tesis tiene como objetivo adicional realizar un estudio que recoja el conocimiento actual sobre los métodos y técnicas existentes para evaluar la calidad en LPS de manera que un investigador o profesional de la industria pueda conocer el estado actual y seleccionar la aproximación que más se ajuste a sus necesidades. Los resultados de ese estudio también proporcionarán información relevante para guiar la definición del método de evaluación propuesto.

Para lograr los objetivos generales de esta tesis, es necesario satisfacer los siguientes objetivos específicos:

- Analizar el estado actual de los métodos y técnicas propuestos para la evaluación de la calidad en el ámbito de las líneas de productos software.
- Analizar los atributos de calidad que los investigadores y profesionales de la industria consideran relevantes para asegurar la calidad en las líneas de productos software.
- Analizar las métricas que han sido propuestas para evaluar la calidad de las líneas de productos software.
- Definir un Modelo de Calidad específico para las líneas de productos software que recoja el estado actual, cubra las

carencias encontradas en el estado del arte, y que siga los estándares de calidad más actuales.

- Definir un método para evaluar la calidad de líneas de productos software que utilice el Modelo de Calidad propuesto.

Aunque la calidad puede describirse desde distintas perspectivas (la calidad del producto en sí, la calidad del proceso utilizado para obtenerlo y la calidad del producto percibida por los usuarios), en esta tesis, nos centramos en la calidad del producto debido a que la industria se ha centrado básicamente en la aplicación de los modelos de calidad del proceso dejando en un segundo plano a los modelos de producto. Además, esa perspectiva es de suma importancia debido a que las evaluaciones de calidad pasan a estar basadas en evidencias extraídas directamente de los atributos del producto y no en evidencias circunstanciales deducidas a partir del proceso [67].

1.3. Método de investigación

El primer paso antes de iniciar la investigación es establecer el método de investigación que se va a seguir. En este trabajo se ha seguido el método mostrado en la Figura 1.1.

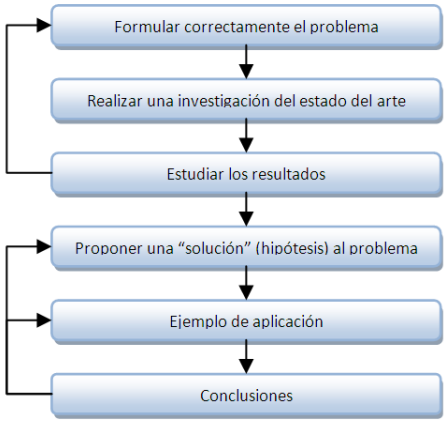

Figura 1.1. Método de investigación.

Como se puede observar, la primera actividad es formular correctamente el problema. A continuación se realiza una investigación del estado del arte. Esta investigación se ha llevado a cabo mediante una *Revisión Sistemática* por ser un método de investigación para obtener, evaluar e interpretar toda la información disponible que está relacionada con una pregunta de investigación o área de interés. Su propósito es proveer un aseguramiento objetivo en un área de investigación de forma fiable, rigurosa y metodológica.

El método de revisión sistemática apareció en el área de la medicina, y su adaptación a la Ingeniería del Software ha sido realizada por Kitchenham [46]. En contraste con una revisión literaria tradicional, una revisión sistemática sigue una secuencia estricta y bien definida de pasos metodológicos, que garantizan el alto valor científico de los resultados obtenidos. La principal razón para llevar a cabo una revisión sistemática es incrementar la probabilidad de detectar más resultados reales en el área de interés que los obtenidos con una revisión informal. Una revisión sistemática requiere un esfuerzo considerablemente mayor en comparación con una revisión tradicional, pero proporciona una revisión profunda y completa de un área de interés determinada.

Figura 1.2. Actividades en el proceso de Revisión Sistemática de la Literatura.

Una revisión sistemática consta de varias actividades, (ver Figura 1.2) las cuales son explicadas brevemente a continuación:

1. **Planificación de la revisión**: se identifica la necesidad de la revisión, las preguntas de investigación son establecidas, y el protocolo de revisión es definido.
2. **Conducción de la revisión**: los estudios primarios son seleccionados, se definen los criterios de aseguramiento de calidad de los estudios primarios incluidos y se extrae y sintetiza la información.
3. **Presentación de resultados**: los resultados de la revisión son interpretados y presentados.

Una vez realizado el estudio del estado del arte y analizados los resultados, se puede pasar a la primera actividad o bien realizar una propuesta en base a los resultados obtenidos. La propuesta es aplicada y en función de los resultados se obtienen unas conclusiones que pueden llevar a replantear la propuesta o repetir la aplicación del método en otras condiciones.

1.4. Organización del documento

El presente documento está organizado en 8 capítulos (ver Figura 1.3). El Capítulo 1 introduce al lector en el contexto y en el alcance del trabajo, además de indicar el método de investigación empleado y la organización del documento.

El Capítulo 2 recoge brevemente los conceptos más importantes sobre las líneas de productos software, su historia, el estado actual, los modelos más utilizados para su representación, etc.

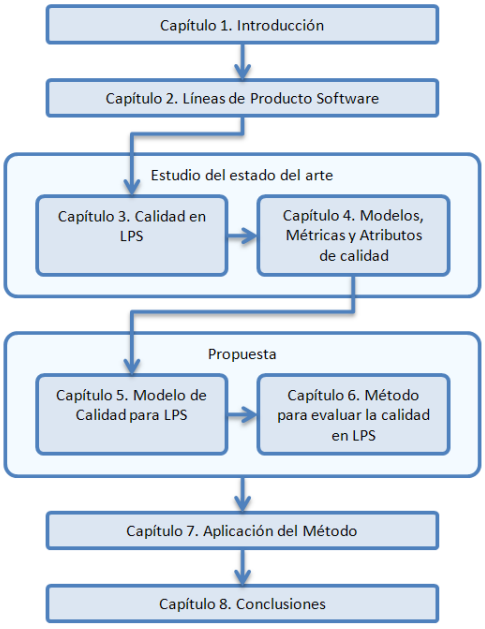

Figura 1.3. Organización de los capítulos.

Los Capítulos 3 y 4 constituyen un estudio del estado del arte. Para estos estudios se ha utilizado la metodología de revisión sistemática por ser un método claro, repetible y objetivo.

En concreto, en el Capítulo 3 se evidencia la importancia de la calidad en la ingeniería del software, y en particular para las líneas de productos software. Con el fin de conocer los métodos y técnicas utilizados por los investigadores durante los últimos 10 años para evaluar las líneas de productos software se ha llevado a cabo una revisión sistemática, cuyos pasos y resultados son discutidos en detalle en dicho capítulo.

En el Capítulo 4 se introducen los modelos de calidad más importantes utilizados durante los últimos años y se explican los estándares de calidad propuestos por la ISO/IEC (9126 y SQuaRE). Estos estándares proponen un modelo de calidad basado en la descomposición jerárquica de Características, Subcaracterísticas y Atributos, para proponer métricas que

evalúen los atributos. Con el fin de conocer los atributos de calidad que han sido considerados importantes por los investigadores para líneas de productos software, además de las métricas que han sido empleadas, se ha realizado una revisión sistemática para identificar y clasificar dicha información. Los resultados obtenidos constituyen un primer paso para la propuesta de un modelo de calidad para líneas de productos software.

Los capítulos 5 y 6 constituyen las contribuciones del trabajo. El Capítulo 5 describe un modelo de calidad para líneas de productos software compuesto por un conjunto de características, subcaracterísticas, atributos de calidad y métricas para medir dichos atributos. Además, el modelo ha sido propuesto siguiendo un enfoque de desarrollo de software dirigido por modelos por lo que las métricas se pueden aplicar tanto a nivel de modelos independientes de la plataforma como artefactos software finales. Para ello, se presenta una propuesta de un metamodelo para describir los elementos del modelo de calidad propuesto.

El Capítulo 6 describe el método de evaluación de calidad para líneas de productos software. El método es genérico, por lo que puede ser aplicado a cualquier ciclo de vida de desarrollo de LPS. No obstante, el objetivo final es aplicarlo a un enfoque dirigido por múltiples modelos, el cual se describe en el capítulo. En el método, se integra el uso del Modelo de Calidad propuesto en el Capítulo 5 al ciclo de vida de desarrollo de la Línea de Productos Software. El modelo de calidad constituye uno de los modelos a tener en cuenta para crear la línea de productos software, es decir, es un artefacto activo durante el ciclo de vida del desarrollo de la línea de productos.

El Capítulo 7 presenta un ejemplo de aplicación del modelo de calidad y método propuestos para ayudar al lector en el entendimiento de las propuestas.

Finalmente, el Capítulo 8 describe las conclusiones del trabajo, las publicaciones generadas y los trabajos futuros.

Capítulo 2

Líneas de Productos Software

La esencia de las Líneas de Productos Software radica en abastecer de recursos, modelos y técnicas para la creación de un conjunto de productos software que aúnan características comunes y que además poseen cualidades únicas de cada producto. De este modo, el desarrollo de un producto nuevo se ve facilitado por la existencia de una arquitectura común a la que hay que añadir elementos distintivos y específicos, promoviendo la industrialización del desarrollo de software. Este paradigma de desarrollo permite mejorar la calidad del software así como reducir los costes y tiempos de lanzamiento.

2.1. Historia de las Líneas de Producto Software

Hasta finales del siglo XVIII, la economía de los países se basaba en la elaboración artesanal. Este método elevaba considerablemente el coste de una unidad de producción debido al tiempo invertido en su obtención. En 1769 James Watt patentó la primera solución práctica de la máquina de vapor y se inició lo que denominamos la 1ª Revolución Industrial. Con los nuevos modelos de maquinaria, fabricar un solo producto era costoso debido a la inversión que suponía, pero fabricar grandes cantidades del mismo era más fructífero. Con la producción masiva y en serie, el problema que se planteó fue que el producto final perdía el modelo de identificación respecto al comprador y a su estilo de vida, motivado por el nuevo concepto de prestigio social. La reutilización de piezas mediante la combinación, para conseguir acabados diferentes de un mismo producto inicial, se presentó como la solución idónea al problema planteado. El cambio supuso pasar de una masiva producción a una masiva personalización. Este procedimiento es el utilizado actualmente para la elaboración de la mayoría de productos, tan comunes como coches o aviones.

Al igual que ocurre en la mayoría de procesos de creación de bienes y servicios, la industria dedicada a la construcción de software tampoco es ajena a la necesidad de crear productos altamente personalizados que escapen de la producción en serie. Esta es la idea subyacente de las Líneas de Productos Software (LPS), dotar al usuario de las herramientas necesarias para automatizar al máximo el proceso de producción. El software resultante será único, en función de la adición de los elementos propios y característicos a una base común.

Las Líneas de Productos Software son empleadas por la eficiencia y producción sistemática del producto software. Pero sus beneficios no pueden ser conseguidos si la reutilización de software no es controlada y planeada de forma sistemática. Un producto que es parte de una línea de productos software es creado basándose en la reutilización de activos ya existentes. Alguno de estos activos es común a otros productos en la misma línea de productos y algunos otros son específicos para productos individuales. Por lo tanto, un aspecto fundamental en este contexto es capturar y expresar variabilidades y similitudes entre los diferentes productos. Una cuestión técnica fundamental que preocupa a los ingenieros de software cuando hacen frente a una línea de productos software es como especificar la propia línea de producto software en si misma, para así expresar similitudes y variabilidades.

El concepto de Líneas de Producto no es nuevo en otros sectores productivos. Líneas de producto de vehículos, de aplicaciones, e incluso de hamburguesas son comúnmente puestas en práctica. En todos los casos, la manera en que un producto es especificado es similar: usando características del producto. Así pues, en las Líneas de Productos Software,

las características son ampliamente empleadas para especificar productos individuales y los *Modelos de Características* se usan para especificar la Línea de Productos Software en sí misma, donde la *característica* es considerada como un aumento en la funcionalidad del producto.

Para comprender la necesidad de un modelo que permita disponer de la flexibilidad de un artesano con los costes de la producción en cadena, es necesario comparar la producción de software a la de cualquier otro bien.

Por todos es conocido el mundialmente famoso FORD T, primer producto fabricado bajo las bondades de la producción en cadena. El Ford Modelo T (coloquialmente conocido como el Tin Lizzie y el Flivver) era un automóvil producido por Ford Motor Company de Henry Ford desde 1908 a 1927. Con el mismo se introdujo la producción en cadena, popularizando la adquisición de los coches. El modelo T incluía novedades que otros vehículos de la competencia no ofrecían, como el volante situado en el lado izquierdo, de gran utilidad para la entrada y salida de los ocupantes, o la incorporación de grandes adelantos técnicos como el conjunto bloque del motor, cárter y cigüeñal en una sola unidad, utilizando para ello una aleación ligera y resistente de acero de vanadio. Este modelo se caracterizó por ser muy espartano, características propias de los vehículos de Henry Ford, y por su política de producción: la cadena de montaje. Gracias a ésta pudo rebajar su precio inicial de USD850 hasta un precio irresistible de USD260 que convirtió a este modelo en el favorito de una sociedad trabajadora industrial, si bien, debía ajustarse a las características de este vehículo, pues la producción en cadena impedía que los consumidores pudieran elegir algo tan común en nuestros días como es el color de la carrocería o cualquier otra cualidad.

Figura 2.1. Ford T:Primer producto fabricado bajo la producción en cadena.

El Ford T (Figura 2.1) representa a la perfección el paradigma de la personalización de los productos: la dificultad de ofrecer un bien único en tiempo record y con costes ajustados. La personalización en serie (mass customization), la capacidad para crear de forma eficiente múltiples variaciones de un producto, es un reto importante, tanto para la industria del automóvil, como para la del software.

2.2. Líneas de Productos para la Ingeniería del Software

Del mismo modo que se requieren buenos procesos de producción para la elaboración de productos tangibles, en el software también se precisan buenos procesos de producción. La idea de reutilizar partes comunes y seleccionar partes variables de un producto es conocida en la ingeniería del software como Líneas de Productos Software (SPL).

2.2.1. Una corta historia de la Ingeniería de Líneas de Productos Software

El sueño de la reutilización masiva del software es tan viejo como la propia ingeniería del software. Numerosos intentos e iniciativas para la reutilización del software han sido realizadas, pero normalmente con poco éxito. Estas iniciativas para la reutilización fueron normalmente basadas en una aproximación enfocada en pequeña escala, reutilización ad hoc (por ejemplo: típicamente a nivel de código, o al menos en la fase de desarrollo, además el desarrollo de nuevos activos era raramente basado en el análisis sistemático de la futura variabilidad).

A finales de la década de 1970 Parnas [59] había propuesto el concepto de familias de productos. Mientras su objetivo era inicialmente la variabilidad en las características no funcionales, el concepto de líneas de productos fue el origen de este trabajo.

El concepto de líneas de productos fue introducido a principios de la década de 1990. Una de las primeras contribuciones fue la descripción de método FODA (Feature-Oriented Domain Analysis)[44]. Simultáneamente, distintas compañías empezaron a tratar el problema de manera más sistemática. Por ejemplo, Philips introdujo el método de Construcción mediante Bloques (Building-Block) a principios de la década de 1990 [70]. Estas primeras aproximaciones fueron el apalancamiento de inversiones masivas en Europa en el área de la ingeniería de líneas de productos software, dentro de compañías y en partes de proyectos científicos. Las siguientes son algunas de ellas:

- ARES (1995-1998): Architectural Reasoning for Embedded Systems.
- Praise (1998-2000): Product-line Realisation and Assessment in Industrial Settings.
- ESAPS (1999-2001): Engineering Software Architectures, Process and Platforms [21].
- CAFÉ (2001-2003): from Concepts to Application in system-Family Engineering [13].
- FAMILIES (2003-2005): Fact-based Maturity through Institutionalisation, Lessons-learned and Involved Exploration of System-family engineering [25].

Estos proyectos soportaron la construcción sistemática de una comunidad de investigadores y profesionales de ingeniería de líneas de productos software en Europa. Durante el mismo tiempo el SEI (Software Engineering Institute) soportó el desarrollo de la ingeniería de LPS en EEUU, más notablemente en el contexto de la organización gubernamental.

2.2.2. Fundamentos de la Aproximación de la Ingeniería de Líneas de Productos Software

La clave diferenciadora entre el desarrollo tradicional de productos únicos y la ingeniería de líneas de productos es un cambio fundamental de enfoque: desde el sistema y el proyecto individual, a la línea de productos. Este cambio implica especialmente un cambio de estrategia: desde la visión de construir un producto a propósito para un dominio, a una vista estratégica del campo de negocio.

La ingeniería de LPS cuenta con la distinción fundamental del desarrollo para la reutilización y el desarrollo con reutilización como se muestra en la Figura 2.2. En la **ingeniería del dominio** (desarrollo para la reutilización) una base es provista para el desarrollo actual de productos individuales. En contra de muchas aproximaciones tradicionales para la reutilización que se enfocan en activos de código, la infraestructura de LP abarca todos los activos que son relevantes a lo largo del todo el ciclo de vida del desarrollo del software. Varios activos cubren el rango completo desde el estado de los requisitos, pasando por la arquitectura y la implementación, hasta las pruebas. Este rango de activos juntos define la infraestructura de la línea de productos. Una distinción clave de la ingeniería de LPS desde otras aproximaciones de reutilización es que los propios activos contienen explícitamente la variabilidad. Por ejemplo, una representación de los requisitos podría contener una descripción explícita de los requisitos específicos que aplicar sólo para un cierto subconjunto de los productos.

Los activos individuales en la infraestructura de LP son enlazados entre sí como los activos en el desarrollo del software. Por ejemplo, la trazabilidad es definida entre activos individuales, de forma que permite tomar en cuenta requisitos e identificarlos con las implementaciones de código y los casos de prueba.

En la **ingeniería de la aplicación** (desarrollo con reutilización) se construye los productos finales con la infraestructura de la LPS. La ingeniería de la aplicación está fuertemente dirigida por la infraestructura de la LPS, la cual normalmente contiene mucha de la funcionalidad requerida para un nuevo producto. La variabilidad explícitamente modelada en ella proviene de la base para derivar los productos individuales. Básicamente, cuando un nuevo producto es desarrollado, un proyecto acompañado es levantado. Entonces los requisitos son unidos y

directamente categorizados como parte de la LP (es decir, una comunalidad o variabilidad) o la especificación del producto. Entonces varios activos (por ejemplo: arquitectura, implementación, etc.) pueden ser instanciados de forma correcta, siguiendo una versión inicial del producto. En este estado del desarrollo, más del 90% del producto puede estar disponible desde la reutilización; sólo el restante 10% debería ser desarrollado en siguientes pasos.

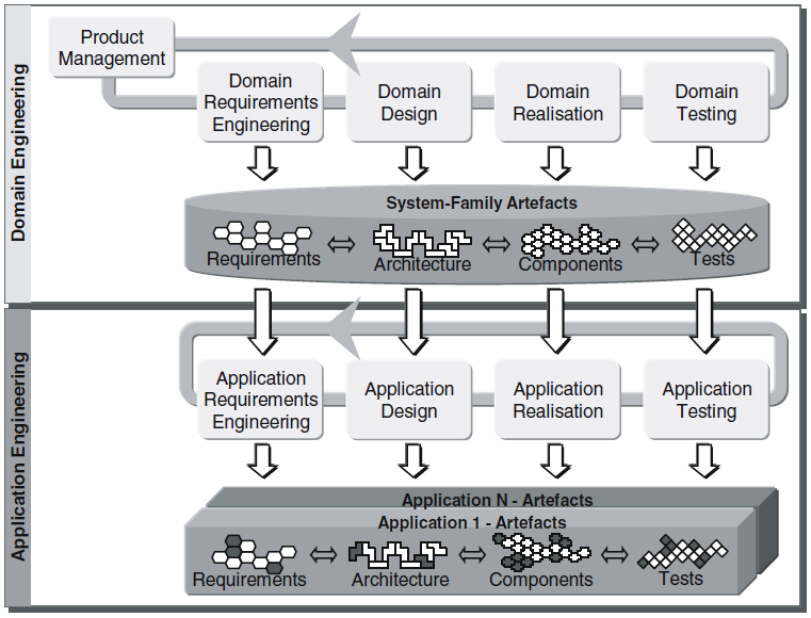

Figura 2.2. Ciclo de vida de las Líneas de Productos Software [71].

Distintos principios son fundamentales para el éxito de la ingeniería de LPS. Estos pueden ser descritos como sigue:

- Manejo de la variabilidad: sistemas individuales son considerados como variantes de una temática común. Esta variabilidad se hace implícita y debe ser manejada sistemáticamente.
- Centrado en el negocio: la ingeniería de LPS apunta a conectar la ingeniería de LP con la estrategia de negocio a largo plazo.
- Centrado en la arquitectura: el lado técnico del software debe ser desarrollado de forma que permita tomar ventajas de las similitudes entre sistemas individuales.
- Aproximación en dos ciclos de vida: los sistemas individuales son desarrollados basados en una plataforma software. Estos productos, al igual que la plataforma, deben ser "engineered" y tener sus ciclos de vida individuales.

2.2.3. Ciclo de vida de las líneas de productos software

Como se ha comentado en secciones anteriores, el ciclo de vida de desarrollo de las líneas de productos software difiere del ciclo de vida de desarrollo tradicional del software. Básicamente, el ciclo de vida de las LPS se divide en dos fases generales: la Ingeniería del dominio y la Ingeniería de la Aplicación. En la Ingeniería del dominio se construye la LPS completa, es decir, la arquitectura base, los activos reutilizables, etc. En la Ingeniería de la aplicación, básicamente se toman los activos de la fase de la Ingeniería del dominio y se combinan para dar lugar a un producto final.

Cada una de estas fases es dividida en 4 actividades: Requisitos, Diseño, Realización y Pruebas. A continuación se detalla cada una de las actividades del ciclo de vida.

2.2.3.1. Ingeniería del dominio

En la fase de la ingeniería del dominio se obtiene la arquitectura de la línea de productos y los activos con los que se podrán construir los productos software. Es la mayor máxima responsable del alcance de la línea de productos, y debe asegurar que la plataforma tiene la variabilidad necesaria para soportar el alcance necesario de los productos.

2.2.3.1.1. Requisitos en la ingeniería del dominio

Es el proceso de crear y manejar los requisitos que deberá cumplir la arquitectura y la implementación. El resultado de los requisitos debe cubrir todos los requisitos de las aplicaciones junto con el alcance de la línea de productos. Esto incluye requisitos comunes y los puntos de variación que la arquitectura de la línea de productos debe soportar.

La ingeniería de requisitos tiene 5 fases básicas [60]:

- *Elicitación*: el análisis de las necesidades de los usuarios y los stakeholders.
- *Documentación*: escribir las necesidades de los productos de forma precisa.
- *Negociación*: los stakeholders intentan investigar y adecuar el nivel de consenso de los requisitos con los requisitos documentados. La forma en la cual esto es hecho depende fuertemente de la cultura de la empresa.
- *Validación y verificación*: el resultado es un conjunto de requisitos que son claros, completos, correctos y entendibles.
- *Gestión*: acuerdos con el mantenimiento de los requisitos a través del desarrollo y el resto del ciclo de vida de la plataforma. A menudo, las fases previas serán revisadas durante la fase de gestión, cuando nuevo requisitos se incorporen y existan nuevos cambios.

2.2.3.1.2. Diseño en la ingeniería del dominio

Esta fase toma como entrada los requisitos y crea la arquitectura de referencia de la línea de productos. Esta arquitectura es la base para el diseño de productos durante el diseño en la ingeniería de la aplicación.

2.2.3.1.3. Realización en la ingeniería del dominio

Los activos comunes son diseñados y creados. En ocasiones puede ser ventajoso reutilizar implementaciones ya existentes, en lugar de crearlas de nuevo. Para cada activo, debe decidirse cuál es la mejor opción para su construcción (crearlo en la propia empresa, comprarlo, reutilizarlo de otros proyectos, encargar su realización a otra empresa).

2.2.3.1.4. Pruebas en la ingeniería del dominio

Una plataforma de líneas de productos termina en diferentes productos finales. Los errores en la plataforma pueden propagarse a estos productos. Por lo tanto, es muy importante asegurar que la plataforma tiene un cierto nivel de calidad. De esto modo, una fase de pruebas para evaluar la línea de productos es crucial para un correcto funcionamiento.

El mayor problema que se presenta durante la fase de pruebas es el tratamiento de la variabilidad. Los mecanismos de variación hacen posible que a partir de una simple línea de productos con un modesto número de puntos de variación, de lugar a un inmenso número de posibles configuraciones. Además, es posible que no puedan predecirse con seguridad todos los posibles productos que serán creados en un futuro. Por lo tanto, no es trivial asegurar la calidad de todos los posibles productos que pudieran ser alcanzados por la línea de productos en futuro.

2.2.3.2. Ingeniería de la Aplicación

En la Ingeniería de la Aplicación se toman los activos de la línea de productos y se utilizan para crear productos. Las variabilidades se combinan con diferentes activos que dan lugar a instancias de productos. Además, es posible combinar estas estancias por activos específicos para una aplicación, es decir, activos que han sido desarrollados específicamente para una aplicación y que no forman parte de la línea de productos.

2.2.3.2.1. Requisitos en la Ingeniería de la Aplicación

Los requisitos en la Ingeniería de la Aplicación deben especificar completamente un producto en particular. Los requisitos son establecidos por los stakeholders del producto bajo desarrollo, por ejemplo los usuarios finales, los gestores del producto, clientes o ingenieros.

Los requisitos de lo activos producidos en la Ingeniería de Dominio pueden ser un punto de partida, pero éstos pueden no satisfacer todos los requisitos de los stakeholders. La diferencia entre los requisitos disponibles

y los requeridos debe ser analizada, además de realizar un análisis trade-off para tomar las decisiones pertinentes para cada requisito no satisfecho. Los stakeholders pueden tratar de adaptar sus requisitos a los activos disponibles en la plataforma, o, si es necesario, crear activos que sean específicos para el producto en particular.

Si los nuevos requisitos son compartidos por diferentes aplicaciones, puede realizarse una actualización en la ingeniería del dominio para que la nueva versión de la plataforma satisfaga estos requisitos.

2.2.3.2.2. Diseño en la Ingeniería de la Aplicación

En la fase de Diseño de la Ingeniería de la Aplicación, la arquitectura de un producto es derivada desde la arquitectura de referencia. Sus puntos de variación son instanciados para crear una arquitectura que cumple todos los requisitos de un cierto producto. La arquitectura resultante es extendida con nuevos componentes e interfaces, o componentes particulares son reemplazados, con el fin de satisfacer todos los requisitos específicos de la aplicación que no son cubiertos por la arquitectura de referencia.

2.2.3.2.3. Realización en la Ingeniería de la Aplicación

El objetivo del proceso de realización en la ingeniería de la aplicación es implementar los productos. Los activos comunes desarrollados en el proceso de realización de la ingeniería del dominio son utilizados para minimizar los esfuerzos y el tiempo.

Las interfaces del dominio pueden ser reutilizadas sin cambios, pero los componentes variables tienen puntos de variación internos que deben ser enlazados. Para este fin, los componentes ofrecen mecanismos tales como enlaces de parámetros y archivos de configuración. A menudo, un número de componentes del dominio necesitan ser configurados de forma similar en una aplicación única. Repetir la misma configuración manual para cada uno de ellos tiene un mayor coste y pude introducir errores. Componentes para configuraciones especiales o herramientas de soporte son dos opciones para realizar esta tarea de manera más eficiente.

En la realización de la aplicación, los activos son complementados por componentes específicos de la aplicación. En muchos casos, estos componentes provendrán de interfaces que son específicas para la arquitectura del dominio.

2.2.3.2.4. Pruebas en la Ingeniería de la Aplicación

Las pruebas en la Ingeniería de la Aplicación son llevadas a cabo para asegurar que el producto final tiene el nivel de calidad suficiente. Aunque los componentes comunes fueron probados en la ingeniería del dominio, esto no significa que no deban ser probados otra vez. Los componentes en la ingeniería del dominio tienen una variabilidad interna que fue enlazada para hacerlos usables en la ingeniería de la aplicación. Es muy complicado

y costoso, por no decir imposible, predecir en la ingeniería del dominio todas las posibles combinaciones de configuraciones que se pueden realizar en un futuro. Por lo tanto, los componentes deben ser probados otra vez en el contexto de la aplicación específica creada para eliminar posibles errores.

Para los requisitos de la aplicación que son idénticos a los del dominio, los mismos casos de prueba pueden ser repetidos. Además, se realizan otros casos de test que evalúan los nuevos requisitos, y comprueban que los nuevos requisitos no han influido sobre los activos creados en la ingeniería del dominio.

2.3. Modelado de LPS

La esencia de las Líneas de Productos Software es la sistemática y eficiente creación de diferentes productos para satisfacer las necesidades de los consumidores. Una pregunta clave que enfrenta los ingenieros del software es cómo especificar un producto particular en una Línea de Producto Software.

Los Modelos de Características son reconocidos en la literatura como una de las más importantes contribuciones a la ingeniería de Líneas de Productos Software. Una labor primordial en la ingeniería de Líneas de Productos es la captura de las similitudes y variaciones entre los productos, y los Modelos de Características son usados para este fin. La Figura 2.3 muestra la representación gráfica de un modelo de características clásico.

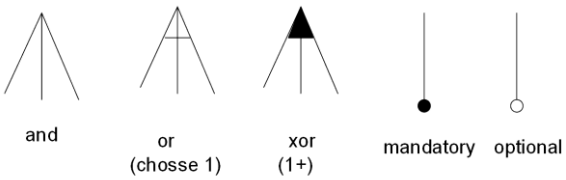

Figura 2.3. Notación gráfica utilizada en el modelo de características clásico.

Un Modelo de Características representa todos los posibles productos de una Línea de Productos Software. Es un conjunto de características jerárquicamente ordenadas compuestas por:

- Relaciones entre una característica padre (o componente) y su hijo (o subcaracterística).
- Restricciones en forma de árbol que son básicamente la inclusión o exclusión de sentencias de la forma: si la característica F está incluida, entonces las características A y B deben estar también incluidas (o excluidas).

Desde que los modelos de características fueron presentados por primera vez en 1990, se han sucedido numerosas publicaciones y

propuestas para extender, mejorar y modificar el modelo de características original. No obstante, a pesar de los años de investigación, no hay consenso sobre la notación del Modelo de Características. Hay dos tipos de notaciones: indicando las cardinalidades y los modelos sin cardinalidades. La principal diferencia radica en que el primero permite relaciones más complejas en la estructura de árbol que el segundo.

Desde la introducción de los modelos de características existe un desafío que todavía hoy no ha podido ser resuelto, es el análisis automatizado de los Modelos de Características. El análisis de los modelos de características consiste en la observación de sus peculiaridades como si el modelo en si estuviera vacío (no representara productos).

Los Modelos de Características son considerados por algunos autores como uno de los más importantes en la ingeniería de Líneas de Productos Software, ya que, es más natural e intuitivo, tanto para clientes como para desarrolladores, expresa similitudes y variabilidades de una línea de productos software, en tanto en cuanto las características son entendidas por todos los usuarios.

Por lo tanto, los Modelos de Características constituyen una notación utilizada y valorada dentro de la ingeniería del software. Ya que la especificación se realiza mediante modelos, es posible introducir esta notación dentro de un método de producción de software dirigido por modelos.

2.4. Conclusiones

Una línea de productos software es un conjunto de sistemas intensivos de software que comparten un conjunto de características comunes que satisfacen las necesidades específicas de un segmento de mercado particular y que son desarrollados a partir de un conjunto de activos núcleo comunes en un modo preestablecido.

Por su enfoque de la producción del software, se mejora en tiempo y esfuerzo ante líneas de producto que recogen muchos productos con una base común y variabilidades que pueden añadirse o eliminarse. Es por ello que el aumento de la utilización de las LPS está creciendo considerablemente y cada día es más común ver que grandes empresas utilizan esta metodología de desarrollo.

Capítulo 3

Calidad en Líneas de Productos Software

La calidad es un factor crucial en la Ingeniería del Software, y aún más en la Ingeniería de Líneas de Productos Software donde un error producido en la fase de la Ingeniería del Dominio (arquitectura base, activos, etc.) podría impactar en la calidad de muchos productos.

Aunque en los últimos años han aparecido diversos métodos y técnicas para evaluar la calidad en LPS, éstos no han sido extensamente utilizados por investigadores ni empresas. Además, el estudio de los mismos muestra que ninguno de ellos reúne todas las características deseables que debería contener un método para evaluara la calidad en LPS.

En este capítulo se explican las bases de la calidad en la ingeniería del software y las características específicas que se requieren para LPS, así como un estudio de los métodos y técnicas actuales para evaluar la calidad en LPS.

3.1. Calidad en Líneas de Productos Software

La calidad es un factor relevante para la construcción de cualquier producto, pero aún lo es más en la construcción de LPS, donde un error en el diseño de la línea de productos, en la arquitectura base, en los activos, etc. seguramente repercutirá en los productos finales producidos. Debido a las características propias de las LPS (gestión de las partes comunes, variabilidad, etc.) los métodos y técnicas tradicionales para la evaluación del software no pueden ser directamente aplicados a las líneas de productos software.

En los últimos años han aparecido diferentes métodos y técnicas para la evaluación de la calidad para LPS, tanto métodos nuevos como adaptados de ya existentes.

Sin embargo, no existe ningún estudio que recoja el conocimiento actual sobre la evaluación para LPS. Para comprobarlo se ha llevado a cabo una revisión sistemática limitada en las bibliotecas digitales IEEExplore, ACM e Inspec con el objetivo de averiguar si existe alguna revisión sistemática u otros trabajos que recojan el conocimiento actual sobre la evaluación de la calidad en líneas de productos software. La cadena de búsqueda utilizada es la siguiente: software AND product AND line* AND (review OR study OR systematic OR evaluation) AND quality.

La búsqueda no produjo ningún resultado. Así que también realizamos una búsqueda similar para encontrar revisiones sistemáticas en el campo de líneas de productos software, con la cual se localizaron los siguientes trabajos:

- Khurum *et al.* [49] condujeron una revisión sistemática de soluciones propuestas para la economía en líneas de productos. El objetivo de esta revisión fue analizar la aplicación práctica y la validación de soluciones propuestas en la industria para evaluar su usabilidad y utilidad práctica.
- Souza Filho *et al.* [65] llevaron a cabo una revisión sistemática para analizar aproximaciones en el dominio del diseño.
- Osatretin [58] llevo a cabo una revisión sistemática sobre pruebas en líneas de productos software. El estudio tuvo en cuenta preguntas de investigación relacionadas con la relación entre la línea de productos y el producto, los pasos necesarios para desarrollar y ejecutar las pruebas en líneas de productos software, las aproximaciones que podrían ser utilizadas para desarrollar casos de prueba, y las conexiones entre la reutilización del software y la reusabilidad de los casos de prueba.

Además de estas revisiones sistemáticas, encontramos distintos estudios que comparan métodos y técnicas para evaluar líneas de productos software (por ejemplo [23] [1]).

Un ejemplo es el estudio llevado a cabo por Etxeberria *et al.* [23]. Este estudio compara siete métodos de modelado que tienen en cuenta la variabilidad en los atributos de calidad: modelo basado en objetivos, grafo de interdependencias de características soft-goal, el marco de trabajo COVAMOF para el modelado de la variabilidad, el modelo de características extendido, una jerarquía de definición, redes bayesianas, y los requisitos de calidad para la familia software. El estudió concluyó que ninguno de estos métodos puede considerarse una aproximación global para asegurar los atributos de calidad a través del ciclo de vida de desarrollo por completo.

En [57] , se presenta una comparación entre ATAM (Architecture Tradeoff Analysis Method) y HoPLAA (Holistic Product Line Architecture Assessment). De hecho, HoPLAA es definido sobre las bases de ATAM. Ali Babar [5] también presentó un tutorial en la conferencia APSEC donde compara diferentes técnicas y métodos para la evaluación de líneas de productos software.

Aunque distintas comparaciones sobre métodos de evaluación para LPS han sido presentadas, nosotros no conocemos ninguna revisión sistemática publicada en este tema. La mayoría de los estudios publicados son cuestionarios o comparaciones de literatura informal, los cuales no definen unas preguntas de investigación, un proceso de búsqueda, la definición de cómo extraer la información o el proceso de análisis de la información.

3.2. Métodos y técnicas para evaluar la Calidad en LPS

Seguimos la aproximación propuesta por Kitchenham para realizar una revisión sistemática de la literatura, la cual es explicada en [46]. Una revisión sistemática es una forma de evaluar e interpretar toda la investigación disponible que es relevante a una pregunta de investigación, un tema o un fenómeno de interés en particular [46]. Apunta a presentar una evaluación limpia de un tema de investigación utilizando una fidedigna, rigurosa y auditable metodología.

Una revisión sistemática tiene diferentes fases y actividades, las cuales explicamos brevemente:

1. Planificación de la revisión: la necesidad de la revisión es identificada, las preguntas de investigación son especificadas, y el protocolo de revisión es definido.
2. Conducción de la revisión: los estudios primarios son seleccionados, el aseguramiento de la calidad utilizada parar incluir los estudios es identificado, la información a extraer

y la monitorización es llevada a cabo, y la información es obtenida y sintetizada.

3. Informe de resultados: el mecanismo de diseminación es especificado, y el informe de la revisión es presentado.

A continuación se presentan las actividades de planificación y conducción de la revisión, para, finalmente, mostrar los resultados de la misma.

3.2.1. Pregunta de investigación

De acuerdo a la metodología de las revisiones sistemáticas, el primer paso es establecer la pregunta de investigación, la cual definimos de la siguiente forma:

¿Qué métodos y técnicas han sido utilizados por los investigadores parar asegurar la calidad de las líneas de productos software y cómo han sido utilizados?

La pregunta de investigación nos permitirá resumir el estado actual sobre la evaluación de la calidad in LPS e identificar debilidades en el estado actual de la investigación, con lo que se podrán sugerir nuevas áreas de investigación. El estudio de la población y la intervención es el siguiente:

- Población: trabajos de investigación (específicamente artículos) que exploren formas de evaluar la calidad en LPS.
- Intervención: métodos y técnicas para asegurar la calidad en el desarrollo o evolución de LPS.
- Comparación: análisis de todas las propuestas basadas en los criterios especificados.
- Resultado: mejora de la calidad en LPS.

La presente revisión es más limitada que la revisión sugerida por Kitchenham [46] ya que nosotros no ampliamos el estudio con las referencias de los artículos encontrados. Además, no incluimos otras referencias a informes técnicos, trabajos y tesis doctorales si estos no habían sido publicados en las librerías digitales seleccionadas.

3.2.2. Identificación y selección de estudios primarios

Las fuentes que utilizamos para buscar los estudios primarios son las bibliotecas digitales IEEExplore, ACM e Inspec. Además, incluimos las actas de congresos de distintas conferencias en el campo de la Ingeniería del Software:

- Actas de congresos de conferencias sobre Líneas de Productos Software: SPLC (Software Product Line Conference), PFE (International Workshop on Product-Family Engineering), IWSAPF (International Workshop on Software Architecture for Product Families).

- Actas de congresos de conferencias sobre Calidad del Software: ESEM (Empirical Software Engineering and Measurement), ISESE (International Symposium on Empirical Software Engineering), MENSURA (Int. Conference on Software Process and Product Measurement), METRICS (IEEE International Software Metrics Symposium), QSIC (Conference on Quality Software), QoSA (Quality of Software Architectures), RefsQ (Requirements Engineering: Foundation for Software Quality).
- Actas de congresos de conferencias sobre Ingeniería del Software: ICSE (International Conference on Software Engineering), CAiSE (Int. Conference on Advanced Information Systems Engineering), MODELS (Model-Driven Engineering Languages and Systems), ECSA (European Conference on Software Architecture).

El periodo de la revisión cubre los estudios publicados desde 1998 a 2008. Respecto a las librerías digitales, nos aseguramos que nuestra estrategia de revisión fuera aplicada a revistas y actas de congresos. Experimentamos con distintas cadenas de búsqueda para asegurarnos de que obteníamos el mayor número de estudios relevantes. Finalmente, la cadena seleccionada fue:

software AND product AND (line OR famil*) AND quality AND (method* OR evaluat* OR stud* OR measur* OR assess* OR approach* OR improv* OR optimization OR assur* OR deploy* OR architec* OR develop* OR framework)*

La búsqueda fue realizada utilizando el título y el resumen de los artículos. Esa búsqueda fue realizada en las bibliotecas digital IEEExplore, ACM e INSPEC además de las fuentes inspeccionadas manualmente.

3.2.3. Criterios de Inclusión y Exclusión

Cada estudio identificado por la cadena de búsqueda fue evaluado para decidir si debería ser incluido o no en la revisión sistemática. Las discrepancias fueron resueltas por consenso. Los artículos que cumplían las siguientes condiciones fueron incluidos:

- Artículos completos que evalúan y/o aseguran la calidad de LPS (consideramos un trabajo como artículo completo si tiene 6 o más páginas).

Los siguientes tipos de artículos fueron excluidos:

- Artículos no escritos en inglés.
- Artículos que sólo presentan recomendaciones y principios para la calidad.
- Artículos de introducción a ediciones especiales, libros y talleres.

- Artículos que no aseguran explícitamente la calidad (la calidad sólo es vista como un beneficio obtenido del uso de LPS).

3.2.4. Estrategia de Extracción de Información

Para extraer la información, la pregunta de investigación fue descompuesta en los siguientes criterios (indicamos las posibles respuestas para cada criterio):

1. Tipo de aproximación utilizada para asegurar la calidad (Método, Técnica).
2. Fase del ciclo de vida en el cual el aseguramiento de la calidad es realizado (Ingeniería del Dominio: Requisitos, Diseño, Realización o Pruebas; Ingeniería de la Aplicación: Requisitos, Diseño, Realización o Pruebas; o Evolución).
3. Artefacto(s) evaluado(s) (Arquitectura base, Activo, Arquitectura del producto, Producto final).
4. Mecanismo utilizado para capturar los atributos de calidad (Modelo de Características, Árboles, Especificación formal, Modelos orientados a objetivos, Otros modelos, Representación textual y/o plantillas, Otros mecanismos, Ninguno).
5. Tipo de atributo de calidad evaluado. (Atributos específicos de las LP, Atributos relevantes para el dominio).
6. Niveles de prioridad para el atributo de calidad. (Si, No)
7. Tipo de evaluación de los atributos de calidad. (Cuantitativo, Cualitativo, Híbrido).
8. Análisis de Impactos. (Impactos de requisitos funcionales en atributos de calidad, Impactos entre atributos de calidad, Impactos en la arquitectura de la LP, Ninguno).
9. Soporte para la evaluación. (Manual, Automático).
10. Cumplimiento de estándares. (Si, No)
11. Retroalimentación. (Si, No)
12. Procedimiento de validación. (Cuestionarios, Casos de estudios, Experimentos controlados, No evaluado).
13. Uso actual (Industrial, Académico).

Cada criterio es explicado a continuación:

1. **Tipo de aproximación utilizada para asegurar la calidad.** Clasificamos un artículo como método si presenta una evaluación con unos pasos claramente definidos, su orden, y cuando deben ser utilizados. De lo contrario, consideramos que se trata de una técnica (por ejemplo métricas, guías, etc.).

2. **Fase del ciclo de vida.** La calidad en las LPS puede ser asegurada en dos fases: en la fase de la Ingeniería del Dominio, para asegurar que el desarrollo de los artefactos software (por ejemplo activos) satisface ciertos nivel de calidad y que los atributos son relevantes para el dominio; y en la fase de Ingeniería de la Aplicación, para asegurar que los productos software desarrollados desde los activos reutilizables satisfacen la calidad de los atributos especificados por el producto. Además, cada una de estas fases pueden contener un conjunto de actividades. En este trabajo hemos adoptado el ciclo de vida de desarrollo de LPS propuesto por Van der Linden et al. [71] para clasificar los artículos seleccionados.

   De acuerdo a este marco de trabajo, la fase de la Ingeniería del Dominio contiene las siguientes fases: (1) Requisitos en la ingeniería del dominio es el proceso de crear y gestionar los requisitos para la línea de productos completa. Así, el artículo es clasificado en esta fase si evalúa artefactos en la ingeniería del dominio; (2) Diseño en la ingeniería del dominio es el proceso de crear una arquitectura base para la plataforma. Un artículo es clasificado en esta fase si asegura la calidad del diseño de la línea de productos completa y/o la arquitectura base toma en cuenta la variabilidad, las partes comunes, etc. (3) Realización en la ingeniería del dominio es el proceso de crear activos comunes. Puede ocurrir que valga la pena utilizar implementaciones existentes en lugar de hacerlas nuevas. Así, un artículo es clasificado en esta fase si presenta una evaluación sobre los activos núcleo. (4) Pruebas en el diseño es el proceso de probar la arquitectura base y los activos base creados. Así, un artículo es clasificado en esta fase si presenta una aproximación para testear los artefactos producidos en la fase de la Ingeniería del Dominio.

   Las actividades para la Ingeniería de la Aplicación son: (5) Requisitos en la Ingeniería de la Aplicación direccionan los requisitos de un producto particular con la LP. Así, un artículo es clasificado en esta fase si presenta una evaluación de los artefactos relacionados con los requisitos de la ingeniería de la aplicación (por ejemplo casos de uso, escenarios, etc.). (6) Diseño en la Ingeniería de la Aplicación es la arquitectura de un producto es derivada desde la arquitectura base. Así, un artículo es clasificado en esta fase si evalúa la calidad de la arquitectura del producto. (7) Realización en la Ingeniería de la Aplicación direcciona la implementación de productos utilizando los activos

comunes y los componentes especificados. Así, un artículo es clasificado en esta fase si presenta una evaluación en la implementación de los productos. (8) Pruebas en la ingeniería de la aplicación direcciona las pruebas de los productos implementados. Aunque los componentes comunes han sido testeados en la fase de la ingeniería del dominio, las pruebas en esta fase aseguran que la configuración de los componentes son apropiados para la especificación de la aplicación. Así, un artículo es clasificado en esta fase si presenta un método o técnica que evalúe la calidad del producto final.

Aparte de estas fases, también incluimos otra fase, la cual fue sugerida por Bosch [11]: (9) Evolución, la cual es dirigida por los cambios en los requisitos de los productos de la familia. Así, un artículo es clasificado en esta fase si presenta un método o técnica para asegurar la calidad en la línea de productos al incluir nuevos requisitos o atributos de calidad.

3. **Artefacto(s) evaluado(s).** Además de la fase en la cual la calidad es evaluada, también es importante saber que artefacto es el evaluado. Éste puede ser: (1) Arquitectura base, si la evaluación es realizada en la arquitectura de la línea de productos entera en la fase de la Ingeniería del Dominio, (2) Activos, si la evaluación es realizada en un componente (una unidad de reutilización de grano grueso) de la LP; (3) Arquitectura del Producto, si la evaluación es realizada en la arquitectura del producto; (4) Producto final, si la evaluación es llevada a cabo en el producto entero que es producido en la fase de la Ingeniería de la Aplicación.
4. **Mecanismo utilizado para capturar los atributos de calidad.** Hay diferentes mecanismos para capturar los atributos de calidad. Un artículo podría emplear los siguientes mecanismos para capturar los atributos de calidad: Modelo de Características, Árboles, Especificación formal, Modelos orientados a objetivos, Otros modelos, Representación textual y/o plantillas, Otros mecanismos, Ninguno.
5. **Tipo de atributo de calidad evaluado.** En la literatura, distintas clasificaciones de atributos de calidad pueden ser encontradas. Nosotros seguimos la clasificación propuesta por Etxeberria et al. [23]: atributos de calidad específicos de la línea de productos y atributos de calidad relevantes para el dominio. Los atributos de calidad específicos de las líneas de productos son aquellos que son inherentes o específicos

a las LP. Estos atributos están relacionados con la variabilidad o flexibilidad. Los atributos relevantes para el dominio son aquellos relacionados con un dominio específico de la LP (tales como seguridad en dominio de seguridad críticos, rendimiento en dominios de tiempo real, etc.). Estos atributos deber ser direccionados en la LP, de otro modo las implicaciones o consecuencias podrían ser muy serias y difíciles de solucionar.

6. **Niveles de prioridad para el atributo de calidad.** Un artículo es clasificado como Sí, si indica la posibilidad de establecer diferentes prioridades a los atributos de calidad. En otro caso, es clasificado como No.
7. **Tipo de evaluación de los atributos de calidad.** El método o técnica de evaluación presentada en un artículo en particular es clasificado como cualitativo, cuantitativo o híbrido. Las evaluaciones *cualitativas* están relacionadas con la evaluación de artefactos (arquitectura base, activos, producto) de forma numérica, con valores continuos (por ejemplo un atributo puede ser medido con valores continuos entre 0 y 1). Las evaluaciones *cuantitativas* son aquellas que indican cualidades (por ejemplo un atributo puede ser medido como alto, medio o bajo). Finalmente, consideramos la evaluación *híbrida* si utiliza ambas evaluaciones.
8. **Análisis de Impactos.** Nosotros clasificamos los artículos teniendo en cuenta los siguientes tres tipos de impactos: (1) *Impactos de requisitos funcionales en atributos de calidad*, si la evaluación del método o técnica indica como los requisitos funcionales cambian/afectan a los atributos de calidad; (2) *Impactos entre los atributos de calidad*, si el métodos/técnica permite hacer un análisis trade-off entre los atributos de calidad; (3) *Impacto en la arquitectura de la LPS*, si el método/técnica analiza los impactos de los requisitos en la arquitectura. Finalmente, el artículo es clasificado como (4) Ninguno, si no proviene ningún impacto.
9. **Soporte para la evaluación.** El artículo es clasificado como *Manual* si presenta un método/técnica que es llevado a cabo de forma manual. En otro caso, es clasificado como *Automático*. Sin embargo, en ocasiones la evaluación es semi-automática. En este caso hemos clasificado el artículo como Manual y Automático.
10. **Cumplimiento de estándares.** El artículo es clasificado como *Sí* si presenta un método o técnica que sigue un

estándar de calidad (por ejemplo ISO/IEC 9126, ISO/IEC 9241). En otro caso es clasificado como *No*.

11. **Retroalimentación.** El artículo es clasificado como *Sí* si presenta un método/técnica de evaluación que proviene de retroalimentación al diseñador. La retroalimentación son recomendaciones de cómo identificar los defectos de calidad que pueden ser solucionados. En otro caso, el artículo es clasificado como *No* si presenta un método/técnica que sólo informa de resultados sobre la evaluación de la calidad.
12. **Procedimiento de validación.** Dependiendo del propósito de la evaluación y las condiciones para la investigación empírica, hay tres diferentes estrategias [26] que pueden ser aplicadas para evaluar el método o técnica: cuestionarios, casos de estudio y experimentos controlados. Un *cuestionario* es una investigación llevada de forma retrospectiva, cuando el método ha sido utilizado durante un cierto periodo de tiempo. Un *caso de estudio* es un estudio observacional y la información es recogida para un propósito específico a través del estudio. Un *Experimento Controlado* proviene de un alto nivel de control y es utilizado para comparar métodos de evaluación de la calidad de una forma más rigurosa.
13. **Uso actual.** El artículo es clasificado como *Industrial* si presenta un método de evaluación o técnica que propone (y es utilizada) en el contexto de un empresa. En otro caso, es clasificada como *Académico* si el artículo describe un método/técnica propuesta en un entorno académico y no hay evidencias sobre si su uso actual también está en una empresa.

3.2.5. Ejecución (realización) de la revisión

La búsqueda para identificar los estudios primarios en las bibliotecas digitales IEEExplore, ACM e INSPEC fue conducida el 21 de Octubre de 2008. La aplicación del protocolo de revisión obtuvo un total de 40 artículos de investigación. No pudimos conseguir uno de los artículos, por lo que realizamos el estudio con 39 artículos. El listado de los artículos puede consultarse en el Apéndice A. Estos fueron los resultados de aplicar los criterios de inclusión y exclusión:

- Las bibliotecas digitales identificaron 518 potenciales relevantes publicaciones (100 desde IEEExplore, 125 desde ACM y 293 desde INSPEC). Después de aplicar los criterios de inclusión/exclusión, 24 publicaciones fueron finalmente

seleccionadas (5 desde IEEExplore, 9 desde ACM y 10 desde INSPEC).

- La bibliografía manual identificó 870 potenciales publicaciones relevantes. Después de aplicar los criterios de inclusión/exclusión fueron seleccionados 16 artículos (10 desde conferencias de LPS, 4 de conferencias de Calidad del Software y 2 de conferencias de Ingeniería del Software). Desafortunadamente, ningún artículo relevante fue encontrado en las conferencias ESEM, MENSURA, QoSA, MODELS y ECSA.

Algunos estudios aparecieron en más de una fuente. Estas publicaciones fueron tenidas en cuenta sólo una vez (en el siguiente orden: (1) actas de congresos, (2) IEEExplore, (3) ACM y (4) INSPEC). Nótese que hemos unido las conferencias SPLC, FPE y SPRINT ARES en el grupo SPL.

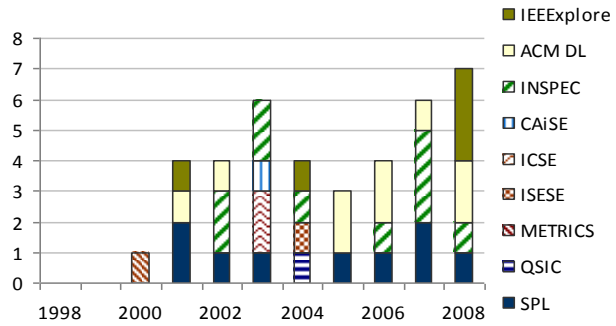

Figura 3.1. Número de publicaciones de calidad en LPS por año.

El análisis del número de estudios de investigación en calidad para LPS (ver Figura 3.1) muestra que ha habido un creciente interés en esta área de estudio. Muchos estudios fueron encontrados en conferencias de LPS, INSPEC y ACM.

3.2.6. Resultados

Hemos organizado los resultados del estudio por los criterios de extracción de información y fuente de publicación. Estos son mostrados en la

Tabla 3.1. Las fuentes de las publicaciones son divididas en distintas conferencias (desde búsquedas manuales) y librerías digitales (IEEExplore, ACM e INSPEC).

Tabla 3.1. Resultados de la revisión sistemática ordenados por criterio de extracción de información y fuente

Selection criteria				Sources									Total
				IEEE	ACM	INSPEC	SPLC	ISESE	METRICS	QSIC	CAiSE	ICSE	
1	Type of method	Method		3	4	6	4	1	1	0	0	0	19
		Technique		2	5	4	5	0	1	1	1	1	20
2	Phase of the SPL life cycle	Domain Engin.	Req.	2	4	3	5	1	1	1	0	0	17
			Des.	2	3	9	7	0	1	0	1	0	23
			Real.	2	2	0	2	0	0	0	0	0	6
			Test.	1	0	0	2	0	0	0	0	1	4
		Applic. Engin.	Req.	2	5	2	2	0	0	0	0	0	11
			Des.	1	4	4	2	0	1	0	1	0	13
			Real.	2	1	0	2	0	0	0	0	0	5
			Test.	1	0	0	4	0	0	0	0	0	5
		Evolution		2	0	1	1	0	1	0	0	0	5
3	Artifacts evaluated	Base Architecture		3	4	9	7	0	2	1	1	1	28
		Core Asset		1	2	1	2	1	1	1	0	0	9
		Product Architecture		1	3	2	1	0	0	0	1	0	8
		Final Product		1	0	1	1	0	0	0	0	0	3
4	Mechanims used to capture the quality attribute	Feature Models		1	1	2	2	0	0	0	0	0	6
		Trees		2	1	1	1	0	0	0	0	0	5
		Formal Specif.		1	0	2	2	0	0	1	0	1	7
		Goal-oriented Model		0	1	1	0	0	1	0	0	0	3
		Other Models		0	4	2	0	0	0	0	0	0	6
		Text/Templates		1	3	0	1	0	0	0	0	0	5
		Other Mechan.		0	0	2	2	1	1	0	1	1	8
		None		1	4	4	3	0	0	0	0	0	12
5	Type of quality attr. evaluated	Specifics SPL		3	2	6	7	1	1	1	0	0	21
		Relevant dom.		2	6	5	5	0	1	1	1	1	22
6	Priority levels	Yes		1	2	3	3	0	1	1	0	0	11
		No		4	7	7	6	1	1	0	1	1	28
7	Type of evaluation	Qualitative		3	6	6	3	1	0	1	0	1	21
		Quantitative		1	2	2	6	1	2	0	0	0	14
		Hybrid		1	1	1	0	0	0	0	1	0	4
8	Impact analysis	FR to Qual. Att.		1	1	3	1	1	1	0	1	0	9
		Between Quality Attrib.		1	1	1	1	1	1	0	0	0	6
		On Architectur.		2	0	1	0	0	0	0	0	1	4
		No impacts		2	7	5	7	0	1	1	0	0	23
9	Support for assessment	Manual		5	9	8	9	0	2	1	0	1	35
		Tool		1	3	3	2	1	1	0	1	1	13
10	Standards conform.	Yes		0	2	2	1	0	0	0	0	0	5
		No		5	7	8	8	1	2	1	1	1	34
11	Feedback	Yes		0	0	1	0	0	2	0	0	0	3
		No		5	9	9	9	1	0	1	1	1	36
12	Validation procedure	Surveys		0	0	0	0	0	0	0	0	0	0
		Case Studies		4	6	8	6	1	2	1	1	1	30
		Experiments		0	0	0	0	0	0	0	0	0	0
		No evaluation		1	3	2	3	0	0	0	0	0	9
13	Actual usage	Industrial		1	0	1	2	0	0	0	0	0	4
		Academic		4	9	9	7	1	2	1	1	1	35

Los resultados para el **criterio 1** (*tipo de aproximación*) indican que la mayoría de las aproximaciones son técnicas (19%). Entre ellas se encuentran métricas tales como las propuestas por [69] (quien introduce métricas de utilización de servicios) o [55] (quien presenta la métrica key node safety para comparar árboles de fallos software con LP). También hay técnicas de reingeniería inversa tales como [61] (quien identifica y evalúa distintas mejoras en la calidad que podrían conseguirse como resultados de la reingeniería inversa).

Con respecto a los métodos (51%) algunos de los trabajos proponen extensiones de métodos existentes para el uso en LPS (por ejemplo EATAM [45], QFD-PPP [35], HoPLAA [57]. Otros trabajos también incluyen extensiones de métodos de LP existentes para incluir calidad [68][24], y otros trabajos presentan nuevos métodos [72]. Por ejemplo, EATAM extiende ATAM para analizar los requisitos específicos y los atributos de calidad para asegurar la arquitectura de la LP [45]. Helferich [35] propone adaptar el método de gestión Quality Function Deployment (QFD) para utilizarlo con LPS. HoPLAA adapta ATAM utilizando una aproximación holística para analizar la arquitectura de la LP y las arquitecturas de los productos individuales [57]. Trinidad et al. [68] extiende FAMA para soportar la implementación del análisis de errores. Etxeberría et al. [22] extienden FeatuRSEB para identificar y especificar la variabilidad que impacta en los atributos de calidad.

Los resultados para el **criterio 2** (*fase del ciclo de vida*) son mostrados en la Figura 3.2. Muchos de los métodos llevan a cabo la evaluación en las fases de Requisitos y Diseño de la Ingeniería de Dominio (45%) e Ingeniería de la Aplicación (26%). Por ejemplo, Mellado et al. [54] provienen una gestión sistemática de la variabilidad de los requisitos de seguridad desde etapas tempranas del desarrollo de las LP; Niemelä e Immonen [56] capturan los requisitos de calidad en la fase de la Ingeniería del Dominio.

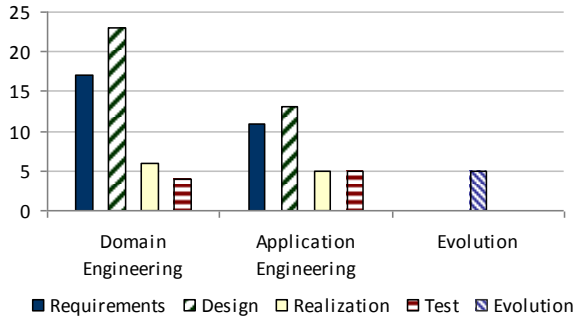

Figura 3.2. Resultados del criterio 2 (fase del ciclo de vida).

Además, algunos métodos aseguran la calidad en más de una fase (por ejemplo Bayer et al. [6] monitorizan la calidad en la fase de diseño de la ingeniería del dominio y de la aplicación, mientras que Etxeberría et al. [24] gestionan la calidad en todo el ciclo de vida de la LPS).

Los resultados para el **criterio 3** (*artefacto(s) evaluado(s)*) muestran que la mayoría de los artículos (58%) evalúan la calidad de la Arquitectura Base (ver Figura 3.3). Este resultado es consistente con la fase del ciclo de vida más evaluada: el diseño en la ingeniería del dominio. Además, pensamos que es un buen resultado para los desafíos puestos por las LPS. Al asegurar la calidad de la arquitectura base, la calidad en la arquitectura del producto puede ser inherente. Sin embargo, el aseguramiento de la calidad en las siguientes fases del ciclo de vida de la LPS es igualmente importante ya que la calidad en estas fases podría ser modificada (por ejemplo al incluir nuevos requisitos).

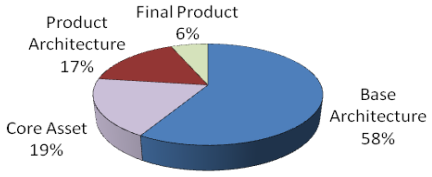

Figura 3.3. Resultados del criterio 3 (artefacto(s) evaluado(s))

Los resultados para el **criterio 4** (*mecanismo utilizado para capturar los atributos de calidad*) muestra que el 23% de los artículo revisados aplican modelos (aproximadamente la mitad de ellos son modelos de características o extensiones). Las especificaciones formales son utilizadas un 13% de los casos. Cerca del 10% de los artículos utilizan árboles para capturar los atributos de calidad. Por ejemplo, Dehlinger y Lutz [16] utilizan el análisis Software Fault Tree para analizar la seguridad en la LP. Además, un 10% expresan los atributos de una forma textual (algunos de ellos utilizando plantillas). Cerca del 15% de los artículos emplean otros mecanismos tales como matrices o escenarios. Por ejemplo, Zhang et al. [72] utiliza Bayesian Belief Networks las cuales ayudan a capturar el impacto de variantes en los atributos de calidad, y ayudan a predecir y asegurar la calidad de un miembro en la LP llevando a cabo un análisis cuantitativo. Finalmente, el 23% no utilizan ninguna técnica para capturar los atributos de calidad (ver Figura 3.4).

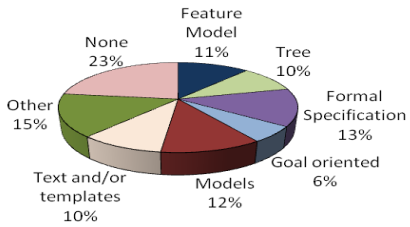

Figura 3.4. Resultados para el criterio 4 (mecanismo utilizado para captar los atributos de calidad)

Los resultados para el **criterio 5** (*tipo de atributo de calidad evaluado*) muestra que el 49% de los artículos evalúan atributos específicos de las LP. Por ejemplo, Del Rosso [17] evalúa atributos tales como modificabilidad, portabilidad, escalabilidad, extensibilidad o rendimiento. El restante 51% evalúa atributos específicos del dominio. Por ejemplo, Capilla y Dueñas [14] usan la capacidad, throughput, tolerancia a fallos, disponibilidad, capacidad de procesamiento o tiempo de retraso.

Los resultados para el criterio 6 (niveles de prioridad para los atributos de calidad) muestra que sólo el 28% de los métodos define niveles de prioridad. Por ejemplo, Jaring [42] formaliza una jerarquía de dependencias de variabilidad.

Los resultados para el **criterio 7** (*tipo de evaluación de atributos de calidad*) muestra que los estudios emplean tanto evaluaciones cualitativas, como cuantitativas o híbridas (ver Figura 3.5). Los métodos evalúan atributos de calidad de forma cualitativa en un 54% de los casos. Los atributos son asegurados de una forma cuantitativa un 36% de los casos. Finalmente, el 10% de los artículos realizan evaluaciones híbridas (incluyendo ambos tipos de evaluaciones). Por ejemplo, Gannod y Lutz [30] razonan sobre escenarios para evaluar atributos de la arquitectura.

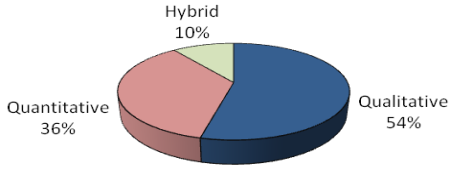

Figura 3.5. Resultados para el criterio 7 (tipo de evaluación de atributos de calidad)

Los resultados para el **criterio 8** (*análisis de impactos*) se muestran en la Figura 3.6. Éstos indican que hay varios métodos que no toman en cuenta ningún tipo de impacto (30%). El 22% de los artículos realizan un análisis trade-off entre los atributos de calidad (por ejemplo Olumofin y Mišić [57]). El 2% de los artículos estudian el impacto en la arquitectura. Por ejemplo, Gannod y Lutz [30] evalúan la modificabilidad de la arquitectura vía escenarios. Finalmente, el 26% de los artículos analizan los impactos entre los requisitos funcionales y los atributos de calidad (por ejemplo, Niemelä e Immonen [56] identifican la variación en la calidad y definen las variaciones de dependencias de los requisitos de calidad en el dominio y diferentes stakeholders).

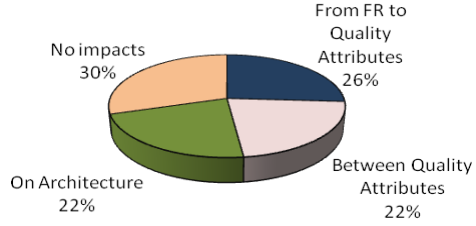

Figura 3.6. Resultados para el criterio 8 (análisis de impactos)

Los resultados para el **criterio 9** (*soporte para el aseguramiento*) muestran que la mayoría de los métodos y técnicas no han desarrollado una herramienta que de soporte para la evaluación (73%). En algunos casos, la herramienta ofrece una evaluación semi-automática. En estos casos, hemos clasificado estos métodos en ambas opciones (manual y automática). Entre las herramientas disponibles se encuentran COVAMOF [64], UA generator [37], y FAMA [68].

En los resultados para el **criterio 10** (*conformidad a estándares*), fue sorprendente observar que el 87% de los métodos y técnicas revisados no seguían ningún estándar. El restante 13% seguía uno o más de los siguientes estándares: IEEE Standard for a SW Quality Metrics Methodology o ISO/IEC: 27001, 13335, 9126, 14598.

Los resultados para el **criterio 11** (*retroalimentación*) muestran que alrededor de un 92% de los estudios no provienen de retroalimentación al diseñador de cómo los artefactos evaluados deberían ser cambiados para satisfacer un cierto nivel de calidad. En general, los métodos provienen algunos resultados de las evaluaciones y los expertos en el dominio son los que deberían interpretarlos para determinar los cambios necesarios. El resto de los estudios (8%) ofrecen sugerencias para realizar cambios en el diseño. Por ejemplo, Van der Hoek et al. [69] utiliza métricas para ayudar a identificar los potencias problemas estructurales y evaluar soluciones alternativas para estos problemas.

Los resultados para el **criterio 12** (*procedimiento de validación*) muestran que el 23% de los estudios no conducen ningún tipo de validación (ver Figura 3.7). Respecto a los que sí lo hicieron, la validación se realizó utilizando el caso de estudio como "prueba de concepto". En general, el método de caso de estudio no ha sido correctamente empleado. Por ejemplo, la mayoría de los artículos no recogían o analizaban la información.

Una excepción es el estudio realizado por Her [36], el cual también realizó un limitado análisis teórico basado en el framework propuesto por Kitchenham [47], además de un análisis de conformidad con algunos criterios descritos en [18] y [38]. Sin embargo, este análisis fue llevado a cabo en un muy alto nivel. Por lo tanto, vemos que hay una necesidad de estudios empíricos en el campo ya que no encontramos ningún estudio que comparara empíricamente los métodos y técnicas de una forma rigurosa.

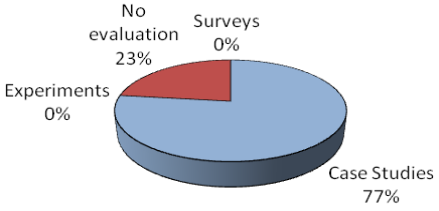

Figura 3.7. Resultados para el criterio 8 (procedimiento de validación).

Finalmente, los resultados para el **criterio 13** (*uso actual*) muestran que los autores de los artículos, comúnmente pertenecen a universidades. Por esta razón, el uso del método es a menudo académico (90%). Sin embargo, a veces el método ha sido adaptado para un uso industrial (10%) [50] [61] [17]. Quizás investigaciones futuras deberían intentar averiguar por qué los métodos de aseguramiento de la calidad no están siendo ampliamente utilizados en el sector industrial.

3.2.7. Amenazas a la validez

Hemos validado nuestro protocolo de revisión para asegurar que la investigación fue lo más correcta, completa y objetiva posible. Con respecto a las fuentes seleccionadas tuvimos en cuenta tres bibliotecas digitales (IEEExplore, ACM e Inspec) conteniendo un gran conjunto de publicaciones en el campo de la Ingeniería del Software. La cadena de búsqueda fue refinada tras probar distintas combinaciones de términos extraídos desde artículo sobre LPS y calidad. También aplicamos terms patterns y adaptamos la cadena de búsqueda a cada biblioteca digital con el propósito de hacer más fácil la replicación del proceso. Para asegurar la fiabilidad de la inclusión de los estudios candidatos en la muestra e identificación de los métodos de evaluación de la calidad in LPS en cada estudio de la muestra, tres evaluadores clasificaron de forma independiente un conjunto aleatorio de 15 estudios de la muestra (10 incluidos y 5 excluidos). El coeficiente Cohen's kappa para la inclusión fue de .82, .87, y .91 para cada uno de los evaluados. Globalmente, esto sugiere un excelente coeficiente de agreement (acuerdo) entre los evaluadores [27].

Las posibles limitaciones de este estudio están relacionadas con la selección del sesgo de las publicaciones, la inexactitud en la extracción de datos y una incorrecta clasificación.

El sesgo de la publicación se refiere al problema de que los resultados positivos son comúnmente más publicados que los resultados negativos [46]. Creemos que hemos aliviado esta amenaza escaneando las actas de congresos más relevantes en el área. Además, las bibliotecas digitales contienen muchas revistas y actas de congresos relevantes. Sin embargo, no consideramos literatura gris (por ejemplo informes de la industria, tesis doctorales, etc.) o resultados no publicados. Con respecto a la selección de las publicaciones, elegimos fuentes donde artículos sobre calidad en LPS son normalmente publicados. Sin embargo, nosotros excluimos algunas bibliotecas digitales a las que no teníamos acceso (por ejemplo Springer). Seleccionamos en nuestra revisión distintos artículos publicados por algunas editoriales o bibliotecas digitales a las cuales no teníamos acceso. Contactamos con los autores de estos artículos en distintas ocasiones para obtener una copia de estos trabajos. Sin embargo, no pudimos conseguir uno de ellos.

Respecto a la cadena de búsqueda, intentamos recoger todos los strings que son representativos de la pregunta de investigación. Refinamos la cadena de búsqueda en distintas ocasiones basándonos en los resultados obtenidos para maximizar la selección de artículos relacionados con la revisión sistemática. Además, tuvimos en cuenta sinónimos y también incluimos sólo el lexema de las palabras.

Intentamos aliviar las amenazas de la inexactitud en la extracción de la información y la incorrecta clasificación realizando la clasificación de los artículos con dos revisores. Las discrepancias entre las evaluaciones fueron discutidas por consenso.

3.3. Conclusiones

En este capítulo se ha visto la importancia de la calidad en la Ingeniería de Software en general y las claves que hacen que aún ésta sea más importante en el contexto de las LPS en particular.

Además, se ha mostrado el estado del arte en métodos y técnicas para la evaluación de la calidad en LPS mediante una revisión sistemática. Decidimos conducir el estudio mediante una revisión sistemática porque es un método de evaluación objetivo y repetible.

Los resultados obtenidos indican que hay varias técnicas para la evaluación, pero ninguna de ellas considera todas las características deseables que aseguren la calidad en LPS. Por ejemplo, muchos métodos no tienen en cuenta: niveles de prioridad en los atributos de calidad (72%); seguimiento de estándares de calidad (87%); provienen de una herramienta para el soporte de la evaluación (73%); provienen de retroalimentación al diseño (92%). Además, hay un número considerable de artículos (el 30% de ellos) que no realizan un análisis de impactos. Muchos de los artículos (54%) llevan a cabo un análisis cualitativo de los atributos. Hay diferentes mecanismos para capturar atributos de calidad (por ejemplo modelos de características, especificaciones formales, árboles, etc.). Sin embargo, es necesario llevar evaluaciones empíricas para proveer de evidencia sobre su efectividad. Muchos de los artículos revisados utilizan casos de estudio para "validar" su método/técnica. Sin embargo, estos casos de estudio fueron llevados a cabo como una estrategia "prueba de concepto". Por lo tanto, no provienen de un significado efectivo para validar los métodos/técnicas. En general, el método de caso de estudio no fue correctamente aplicado. Nosotros observamos problemas respecto al diseño, el análisis de la información y los informes de los casos de estudio.

Los resultados también muestran que ninguno de los artículos revisados realizó estudios empíricos. Los estudios empíricos son la base para construir la evidencia y determinar qué método es mejor en ciertas situaciones. Por lo tanto, hay una urgente necesidad en llevar a cabo experimentos en el campo de LPS.

Capítulo 4

Modelos, Métricas y Atributos de calidad

Los modelos de calidad son artefactos con los que se puede especificar la calidad de un sistema. En la literatura es posible encontrar diversos modelos de calidad para la evaluación del producto software, los cuales se presentan en este capítulo.

El estándar de calidad SQuaRE propuesto por la norma ISO propone un modelo de calidad compuesto básicamente por una jerarquía de características de calidad donde los elementos finales son atributos que pueden ser evaluados mediante métricas. Siguiendo este modelo, y con el objetivo de proponer un modelo de calidad para LPS, se ha realizado un estudio de los atributos de calidad para LPS que ha sido discutido por los investigadores y las métricas que han sido propuestas.

4.1. Modelos de calidad existentes

Un modelo de calidad es una herramienta muy útil para la ingeniería de requisitos de calidad así como para la evaluación temprana y el control de la calidad. Se define como un conjunto de características y relaciones entre ellas con las cuales proveer las bases para especificar los requisitos de calidad y evaluar la misma.

Distintos modelos de calidad han sido propuestos. A continuación vamos a realizar un breve repaso por los modelos de calidad más relevantes que han dado lugar a los más utilizados en la actualidad. Así comentaremos los modelos de calidad de McCall, Boehm y FURPS, para terminar con los modelos de calidad propuestos por el reconocido estándar ISO. Como veremos en este subcapítulo, ISO ha ido proponiendo distintos modelos que han dado lugar al actual modelo SQuaRE donde se recogen los modelos anteriores, realizando mejoras y añadiendo conceptos nuevos.

4.1.1. Modelo de Calidad de McCall (1977)

Uno de los más reconocidos predecesores de los modelos de calidad actuales es el modelo de calidad presentado por Jim McCall *et al.* [52][48] (también conocido para el Modelo Eléctrico General de 1977). Este modelo, al igual que otros modelos contemporáneos, original desde los militares de EEUU (fue desarrollado por la Fuerza Aérea de EEUU, promovido con DoD) y es primeramente direccionado hacía los desarrolladores del sistema y los procesos de desarrollo del sistema. Este modelo de calidad intenta unir el agujero entre los usuarios y los desarrolladores enfocando un número de factores de calidad software que reflejan las vistas de los usuarios y las prioridades de los desarrolladores.

El modelo de calidad de McCall tiene, como se muestra en la Figura 4.1, tres principales perspectivas para definir e identificar la calidad de un producto software: revisión del producto (capacidad para rebajar cambios), transición del producto (adaptabilidad a nuevos entornos), y operaciones del producto (sus caractarísticas operacionales).

La revisión del producto incluye mantenibilidad (el esfuerzo requerido para localizar y corregir un fallo en el programa con su entorno operacional), flexibilidad (la facilidad de hacer cambios requeridos por cambios en el entorno operacional), la realización de pruebas (la facilidad para testear un programa, asegurar que está libre de errores y conforme a su especificación).

La transición del producto es todo lo relacionado con la portabilidad (el esfuerzo requerido para transferir un programa desde un entorno a otro), reusabilidad (la facilidad de reutilizar software en un contexto diferente), y la interoperacionalidad (el esfuerzo requerido para emparejar el sistema con otro sistema).

La calidad de las operaciones del producto dependen de la correctitud (la extensión con la cual el programa cumple su especificación), fiabilidad (la capacidad del sistema a no fallar), la eficiencia (la lejana categorización entre la eficiencia de la ejecución y la eficiencia del almacenaje y generalmente el significado del uso de recursos, por ejemplo, el tiempo del procesador, almacenamiento), integridad (la protección del programa desde acceso no autorizado), y usabilidad (la facilidad del software).

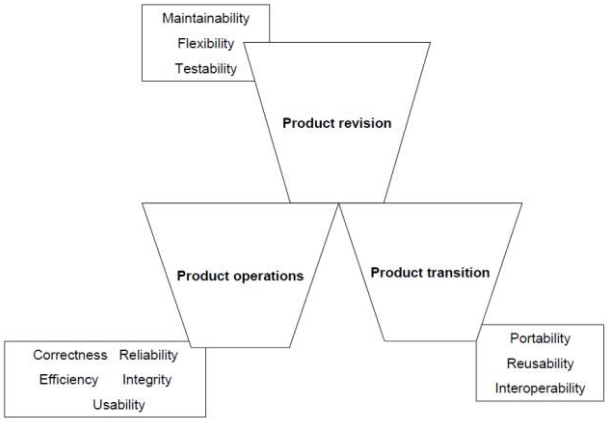

Figura 4.1. Modelo de calidad de McCall.

Además, el modelo detalla los tres tipos de características de calidad (perspectivas mayores) en una jerarquía de factores, criterios y métricas:

- 11 factores (para especificar): describen la vista externa del software como es vista por los usuarios.
- 23 criterios de calidad (para construir): describen la vista interna del software como es vista por los desarrolladores.
- Métricas (para controlar): son definidas y utilizadas para proveer una escala y un método para la medición.

Los factores de calidad describen diferentes tipos de características de comportamiento del sistema, y los criterios de calidad son atributos para uno o más de los factores de calidad. La métrica de calidad captura algunos aspectos de un criterio de calidad.

La idea del Modelo de Calidad de McCall es que los factores de calidades sintetizados puedan proveer una imagen completa de la calidad del software [48]. La métrica de calidad en realidad es conseguida por respuestas "sí" y "no" que son puestas en relación una con otra. Esto es, si el número de respuestas "sí" es el mismo que "no" en las cuestiones medidas, entonces el criterio de calidad alcanza un 50%. Las métricas pueden entonces ser sintetizadas por los criterios de calidad, por factores de calidad, o si es relevante por productos o servicios.

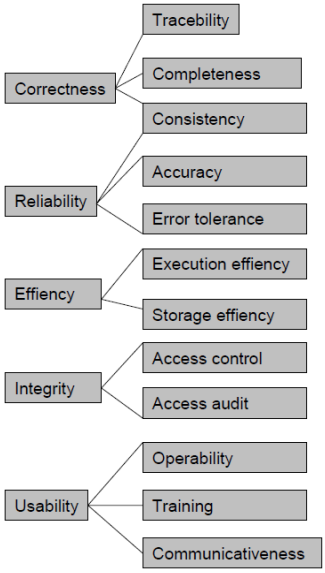

Figura 4.2. Modelo de calidad de McCall.

4.1.2. Modelo de Calidad de Boehm (1978)

El segundo de los predecesores básicos de los modelos actuales es el modelo de calidad presentado por Barry W. Bohem [9][10]. Boehm direcciona los defectos contemporáneos de modelos que evalúan la calidad del software de forma automática y cuantitativa. En esencia su modelo intenta definir cualitativamente la calidad del software por un conjunto de atributos y métricas dados. El modelo de Bohem es similar al Modelo de Calidad de McCall en el que es también presentada una estructura jerárquica del modelo de calidad alrededor de características de alto nivel, características de nivel medio, y características primitivas (cada una de las cuales contribuye al nivel de calidad global).

Las características de alto nivel representan los requisitos de alto nivel del uso real para las cuales la evaluación de la calidad podría ser puesto (la utilidad general del software). Las características de alto nivel direccionan tres principales preguntas que un comprador de software tiene:

- Utilidad: ¿Cómo de bien (facilidad, fiabilidad, eficiencia) puedo utilizarlo?
- Mantenibilidad: ¿Cómo de fácil es de entender, modificar y probarlo?
- Portabilidad: ¿Puedo seguir utilizándolo si cambio de entorno?

Las características de nivel intermedio representan los siete factores de calidad de Boehm que juntos representan las cualidades esperadas del sistema software:

- Portabilidad (características de utilidad general): el código posee la característica portabilidad para la extensión que puede ser operada fácilmente y bien en otras configuraciones de ordenadores.
- Fiabilidad (as-is utility characteristics): el código posee fiabilidad si puede esperarse que lleve a cabo sus funciones de forma satisfactoria.
- Eficiencia (as-is utility characteristics): el código posee la característica eficiencia si cumple su propósito sin desperdiciar recursos.
- Usabilidad (as-is utility characteristics, human engineering): el código posee la característica usabilidad si es fiable, eficiente y human-engineered.
- Testabilidad (Maintainability characteristics): el código posee la característica de testabilidad si facilita el establecimiento de criterios de verificación y soporte a la evaluación de su comportamiento.
- Entendibilidad (maintainability characteristics): el código posee la característica entendibilidad si su propósito es claro para el inspector.
- Flexibilidad (maintainability characteristics, modifiability): el código posee la característica modificabilidad si facilita la incorporación de cambios, una vez que la naturaleza del cambio deseado ha sido determinado. (notar que el alto nivel de abstracción de esta característica es comparada con la aumentabilidad).

El nivel más bajo de la jerarquía estructural de características en el modelo de Boehm es la jerarquía de métricas características primitivas. Las características primitivas provienen los cimientos para definir las métricas de calidad (las cuales son uno de los objetivos cuando Boehm construyó su modelo de calidad). Consecuentemente, el modelo presenta una o más métricas para medir una característica primitiva.

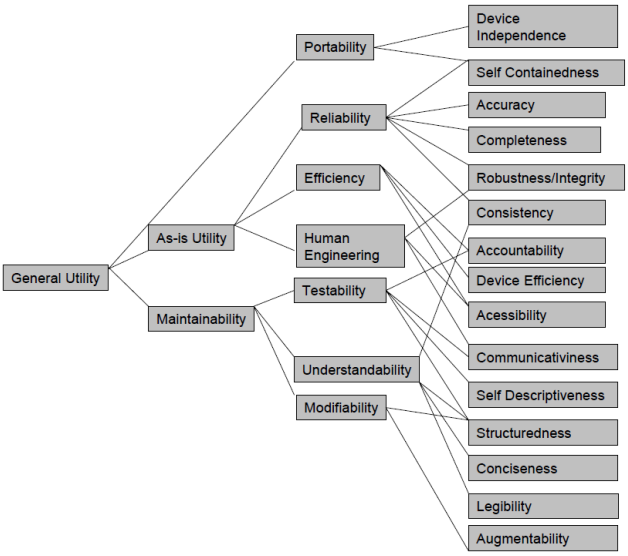

Figura 4.3. Árbol de características de calidad del software de Bohem.

Aunque los modelos de Boehm y McCall podrían parecer muy similares, la diferencia es que el modelo de McCall primeramente se centra en la precisión de las medidas de las características de alto nivel "As-is utility", mientras que el modelo de calidad de Boehm se basa en un ancho rango de características con extensiones y detalles centradas en la mantenibilidad.

4.1.3. FURPS/FURPS+

Después, y por supuesto algo menos reconocido, el modelo que es estructurado en básicamente la misma manera como los dos modelos de calidad previos (pero aún valió la pena que fueran mencionados en este contexto) es el modelo FURPS originalmente presentado por Robert Grady [34] (y extendido por Rational Software [41][62], ahora IBM Rational Software en FURPS+ (nota: el "+" en FURPS+ incluye requisitos como restricciones de diseño, implementación, interface y físicos). FURPS se basa en:

- Funcionalidad: la cual incluye conjuntos de características, capacidades y seguridad.
- Usabilidad: la cual incluye factores humanos, estéticos, consistencia en la interfaz de usuario, ayuda sensible al contexto y en línea (online), asistentes (wizards) y agentes, documentación del usuario, y materiales de entrenamiento.
- Fiabilidad: la cual incluye frecuencia e intensidad de los fallos, recuperabilidad, predicibilidad, precisión, y significado del tiempo entre fallos (mean time between failure MFBF).

- Comportamiento (performance): impone condiciones en requisitos funcionales tales como la velocidad, eficiencia, disponibilidad, precisión, rendimiento (throughput), tiempo de respuesta, tiempo de recuperación, y uso de recursos.
- Soportabilidad: el cual puede incluir testabilidad, extensibilidad, adaptabilidad, mantenibilidad, compatibilidad, configurabilidad, resistibilidad (serviceability), instalabilidad (installability), localizacionalidad (internacionalización).

Las categorías FURPS son de dos tipos diferentes: Funcional (F) y No Funcional (URPS). Estas categorías pueden ser utilizadas tanto como requisitos del producto como en el aseguramiento de la calidad del producto.

4.1.4. Modelo de calidad de Dromey

Un modelo más reciente y similar a los modelos de McCall, Boehm y FURPS(+) es el modelo presentado por R. GeoffDromey [19][20] del artículo]. Dromew propone un producto basado en el modelo de calidad que reorganiza que la evaluación de la calidad difiere de cada product y que una idea más dinámica para modelar el proceso es necesitada para ser lo suficientemente ancho para aplicar diferentes sistemas. Dromey se centra en las relaciones entre los atributos de calidad y los subatributos, y en cómo intentar conectar las propiedades del producto software con atributos de calidad software.

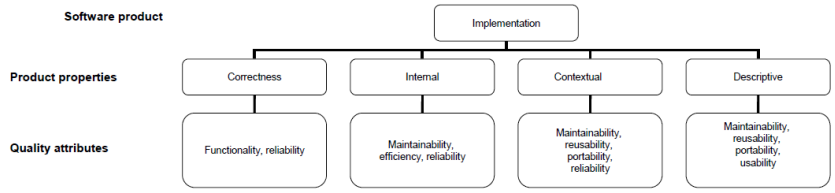

Figura 4.4. Principios del Modelo de calidad de Dromey.

Como muestra la Figura 4.4, hay tres principales elementos del modelo de calidad genérico de Dromey:

1. Propiedades del producto que influencian en la calidad.
2. Atributos de calidad de alto nivel.
3. Significado de los enlaces entre las propiedades de los productos y los atributos de calidad.

El Modelo de Calidad de Dromey se estructura alrededor de cinco pasos:

1. Elegir un conjunto de atributos de calidad de alto nivel necesarios para la evaluación.
2. Listar los componentes/módulos del sistema.

3. Identificar las propiedades que llevan a cabo la calidad para los componentes/módulos (cualidades del componente que tiene el mayor impacto en las propiedades del producto desde la lista anterior).
4. Determinar cómo cada propiedad afecta a los atributos de calidad.
5. Evaluar el modelo e identificar las debilidades.

4.1.5. Estándar ISO 9126

El estándar ISO 9126 está basado en los modelos de McCall y Bohem. Además de estar estructurado en básicamente la misma forma que estos modelos, el ISO 9126 también incluye la funcionalidad como un parámetro, e identifica tanto las características internas como las externas de los productos software.

El modelo de calidad categoriza los atributos de calidad software en seis características (functionality, reliability, usability, efficiency, maintainability y portability), las cuales son subdivididas en subcaracterísticas (ver Figura 4.5). Las subcaracterísticas pueden ser medidas por métricas internas o externas.

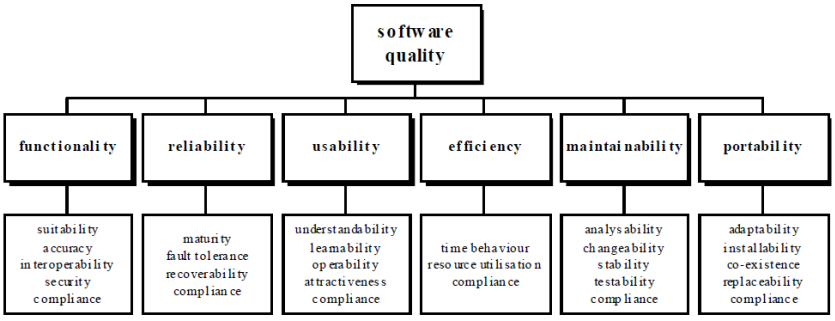

Figura 4.5. Calidad interna y externa.

Las definiciones para cada característica de calidad y las subcaracterísticas del software que influencia la característica de calidad. Para cada característica y subcaracterística, la capacidad del software es determinado por un conjunto de atributos internos que pueden ser medidos. Las características y subcaracterísticas se pueden medir externamente por la capacidad de que es provisto el sistema que contiene el software.

1. Functionality: La capacidad del producto software para proveer funciones que respondan a necesidades expresadas o implícitas, cuando el software es utilizado bajo condiciones determinadas.

1.1. Suitability: La capacidad del producto software para proveer un conjunto apropiado de funciones para tareas específicas y objetivos de usuario.

1.2. Accuracy: La capacidad del producto software para proveer resultados o efectos correctos.

1.3. Interoperability: La capacidad del producto software para interactuar con uno o más sistemas especificados.

1.4. Security: La capacidad del producto software para proteger información y datos de modo que un sistema o persona no autorizada no pueda leerlos o modificarlos y que un sistema o persona autorizada no tenga el acceso denegado.

1.5. Compliance: La capacidad del producto software al cumplimiento de estándares, convenciones o regulaciones legales y prescripciones similares.

2. Reliability: La capacidad del producto software para mantener un nivel específico de rendimiento cuando es utilizado bajo condiciones específicas.

2.1. Maturity: La capacidad del producto software para evitar fallos como resultado de fallos producidos en el sistema.

2.2. Fault tolerante: La cacapidad de los productos software para mantener un nivel específico de rendimiento en caso de fallos software o de violación de sus interfaces específicas.

2.3. Recoverablity: La capacidad del productos software para restablecer un nivel específico de rendimiento y recuperar los datos directamente afectados en el caso de un fallo.

2.4. Compliance: La capacidad del producto software para seguir estándares, convenciones o regulaciones relacionadas con la fiabilidad.

3. Usability: La capacidad del producto software para ser entendido, aprendido, utilizado y atractivo para el usuario, cuando es usado bajo condiciones específicas.

3.1. Understandability: La capacidad del producto software para permitir al usuario entender si el software es adecuado, y se puede utilizar para determinadas tareas y condiciones de uso.

3.2. Learnability: La capacidad del producto software para permitir al usuario aprender su aplicación.

3.3. Operability: La capacidad del producto software para permitir al usuario operarlo y controlarlo.

3.4. Attractiveness: La capacidad del producto software para ser atractivo al usuario.

3.5. Compliance: La capacidad del producto software para seguir estándares, convenciones, guías de estilos o regulaciones relacionadas con la usabilidad.

4. Efficiency: La capacidad del producto software para proveer un rendimiento apropiado, relacionado con la cantidad de recursos utilizados, bajo condiciones establecidas.

4.1. Time behaviour: La capacidad del producto software para proveer respuestas, tiempos de procesamiento y tasas de rendimiento adecuados, bajo condiciones establecidas.

4.2. Resource utilisation: La capacidad del producto software para utilizar cantidades y tipos de recursos apropiados cuando lleva a cabo sus funciones bajo condiciones establecidas.

4.3. Compliance: La capacidad del producto software al cumplimiento de estándares o convenciones relacionados con la eficiencia.

5. Maintainability: La capacidad del producto software para ser modificado. Las modificaciones pueden incluir correcciones, mejoras o adaptaciones del software a cambios en el entorno, y en requisitos y especificaciones funcionales.

5.1. Analysability: La capacidad del producto software para diagnosticar deficiencias o causas de fallos en el software.

5.2. Changeability: La capacidad del producto software para permitir una modificación específica que debe implementarse.

5.3. Stability: La capacidad del producto software para evitar los efectos inesperados de las modificaciones del software.

5.4. Testability: La capacidad del producto software para permitir que el software modificado pueda ser validado.

5.5. Compliance: La capacidad del producto software para cumplir estándares o convenciones relacionados con la mantenibilidad.

6. Portability: La capacidad del producto software para ser transferido desde un entorno a otro.

6.1. Adapatability: La capacidad del producto software para ser adaptado a diferentes entornos específicos sin aplicación de acciones o medios distintos de los previstos para este fin, para el software considerado.

6.2. Instalability: La capacidad del software para ser instalado en un entorno específico.

6.3. Co-existence: La capacidad del software para coexistir con otro software independiente en un entorno común compartiendo recursos comunes.

6.4. Replaceability: La capacidad del software para ser usado en lugar de otro producto software especificado para el mismo propósito en el mismo entorno.

6.5. Compliance: La capacidad del producto software para cumplir estándares o convenciones relacionados con la portabilidad.

Las principales diferencias entre el estándar ISO 9126 y los modelos de McCall y Bohem se muestran en la Tabla 4.1.

Tabla 4.1. Comparación entre los criterios/objetivos de los modelos de calidad de McCall, Boehm e ISO 9126.

Criteria/Goals	McCall	Bohem	ISO 9126
Correctness	✓	✓	
Reliability	✓	✓	Maintainability
Integrity	✓	✓	✓
Usability	✓	✓	
Efficiency	✓	✓	✓
Maintainability	✓	✓	✓
Testability	✓		✓
Interoperability	✓		Maintainability
Flexibility	✓	✓	
Reusability	✓	✓	
Portability	✓	✓	✓
Clarity		✓	
Modifiability		✓	Maintainability
Documentation		✓	✓
Resilience		✓	
Understandability		✓	
Validity		✓	
Functionality			
Generality		✓	
Economy		✓	

4.1.6. Estándar SQuaRE

El estándar de calidad ISO SQuaRE define un modelo de calidad del producto software compuesto por ocho características, las cuales son divididas en subcaracterísticas que pueden ser medidas interna o externamente. Estas subcaracterísticas son manifestadas externamente cuando el software es utilizado como parte de un sistema en el ordenador, y son un resultado de los atributos de software internos y del comportamiento del ordenador.

Un modelo de la calidad en uso del sistema compuesto por tres características, las cuales son divididas en subcaracterísticas que pueden ser medidas cuando un producto es utilizado en un contexto de uso real. Cuando se utiliza para especificar o medir los efectos de la calidad del software en un contexto particular de uso, la calidad en uso puede estar influenciada por cualquiera de las ocho características de la calidad de producto. Aunque la calidad en uso es descrita en el contexto de la calidad del producto software, como una propiedad de los sistemas completos, también puede ser utilizada para evaluar otros componentes del sistema (incluyendo hardware, el usuario o el entorno).

Las características de los modelos son aplicables a cada tipo de software. Las características y subcaracterísticas provienen una

terminología consistente para la calidad del producto software. Éstas también provienen de un conjunto de características de calidad contra las cuales los requisitos de calidad indicados pueden ser comparados completamente.

Los modelos de calidad pueden ser utilizados para soportar especificación y evaluación del software desde distintas perspectivas para estas asociadas a la adquisición, requisitos, desarrollo, uso, evaluación, soporte, mantenimiento, aseguramiento de la calidad y auditoría del software. Éstos pueden por ejemplo ser utilizados por desarrolladores, compradores, personal encargado de asegurar la calidad y evaluadores independientes, particularmente los responsables para especificar y evaluar la calidad del producto software. Las actividades durante el desarrollo del producto que pueden beneficiar desde el uso de los modelos de calidad incluyen:

- Identificar los requisitos de calidad.
- Validar la comprensión de la definición de requisitos.
- Identificar los objetivos del diseño software.
- Identificar los objetivos de las pruebas del software.
- Identificar los criterios de aseguramiento de calidad.
- Identificar los criterios de aceptación para un producto software completo.

ISO/IEC 25012 contiene un modelo para la calidad de los datos que es complementario a este modelo.

Los modelos de calidad pueden ser utilizados en conjunto con los procesos del ISO/IEC 12207 asociados con la definición de requisitos, verificación y validación con un enfoque específico en la especificación y evaluación de los requisitos de calidad. Adicionalmente, los modelos pueden ser utilizados para evaluar un sistema en operación para caracterizarlo en términos de características de calidad.

Este Estándar Internacional puede ser utilizado en conjunto con el ISO/IEC 15504 (el cual se ocupa del aseguramiento del proceso software) para proveer:

- Un marco de trabajo para la definición de la calidad del producto software en el proceso cliente-proveedor.
- Soporte para revisar, verificar y validar, y un marco de trabajo para la evaluación cuantitativa y cualitativa, en el proceso soportado.
- Soporte para establecer los objetivos de la calidad organizacional en el proceso de gestión.

Este Estándar Internacional también puede ser utilizado en conjunto con el ISO 9001 (el cual se ocupa de los procesos de aseguramiento de la calidad) para proveer:

- Soporte para establecer objetivos de calidad.
- Soporte para revisar el diseño, verificar y validar.

El estándar indica que cualquier requisito, especificación o evaluación del producto software que sea conforme al Estándar Internacional deberá usar las características y subcaracterísticas descritas, explicando razonadamente cualquier exclusión, o describir su propia categorización de atributos de calidad del producto y proveer las relaciones entre las características y subcaracterísticas propuestas con las de este estándar.

La calidad del producto software debería ser evaluada utilizando un modelo de calidad definido. El modelo de calidad debe ser utilizado al establecer los requisitos de calidad para los productos software y los productos intermedios. La calidad del producto software debe ser descompuesta jerárquicamente en un modelo de calidad compuesto de características y subscaracterísticas que pueden ser utilizadas como una lista de cuestiones relevantes para la calidad. A continuación se define la jerarquía de los modelos de calidad (aunque otras formas de categorizar la calidad pueden ser más apropiadas en circunstancias particulares).

No es prácticamente posible medir todas las subcaracterísticas internas y externas para todas las partes de un gran producto software. Similarmente no es normalmente práctico medir la calidad en uso para todos los posibles escenarios de las tares de usuarios. La importancia relativa de las características de calidad dependerá del producto y el dominio de aplicación. Así, el modelo debería ser hecho a medida antes de utilizarlo, y los recursos para evaluar las asignaciones entre diferentes tipos de medidas dependiendo de los objetivos de negocio y la naturaleza del producto y los procesos de diseño.

4.1.6.1. Modelo de calidad del producto software

SQuaRE categoriza los atributos de calidad software en ocho características (functional suitability, reliability, performance efficiency, operability, security, compatibility, maintainability y transferability), las cuales son subdivididas en subcaracterísticas (ver Figura 4.6). Las subcaracterísticas pueden ser medidas por métricas internas y externas.

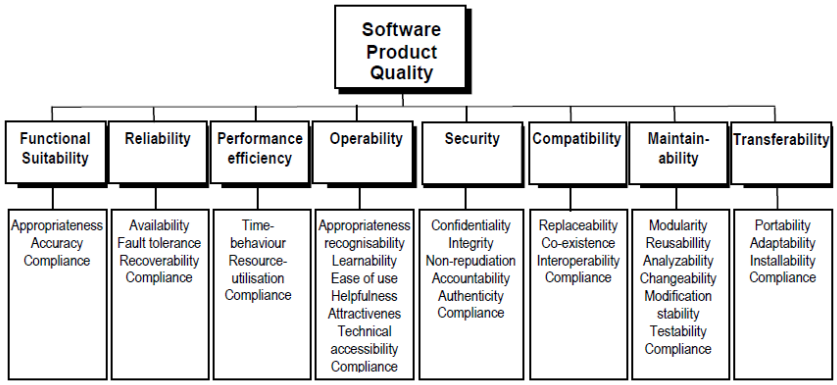

Figura 4.6. Modelo de calidad del producto software.

1. Functional suitability: El grado con el cual el producto software proviene de funciones que cumplen las necesidades implícitas cuando el software es utilizado bajo unas condiciones específicas.

1.1. Appropriateness: El grado con el cual el producto software proviene un conjunto de funciones apropiadas para especificar tareas y objetivos del usuario.

1.2. Accuracy: El grado con el cual el producto software proviene los resultados correctos o específicos con el grado necesario de precisión.

1.3. Functional suitability compliance: El grado con el cual el producto software es conforme a estándares, convenciones o regulaciones preescritas similares relacionadas con la idoneidad de la funcionalidad.

2. Reliability: El grado con el cual el producto software puede mantener un nivel específico de rendimiento cuando se utiliza bajo condiciones específicas.

2.1. Availability: El grado con el cual un componente software es operacional y disponible cuando se requiere su uso.

2.2. Fault tolerante: El grado con el cual el producto software puede mantener un nivel específico de rendimiento en casos de fallos en el software o de violación de sus interfaces específicas.

2.3. Recoverability: El grado con el cual el producto software puede reestablecer un nivel específico de rendimiento y recuperar la información directamente afectada en caso de un fallo.

2.4. Reliability compliance: El grado con el cual el producto software es conforme a estándares, convenciones o regulaciones relacionadas con la fiabilidad.

3. Performance efficiency: El grado con el cual el producto software proviene de un apropiado rendimiento, relativo a la cantidad de fuentes utilizadas, bajo condiciones establecidas.

3.1. Time behaviour: Grado con el cual el producto software proviene de la respuesta adecuada y el tiempo de procesamiento y las tasas de rendimiento para llevar a cabo su función, bajo condiciones establecidas.

3.2. Resource utilisation: El grado con el cual el producto software utiliza las cantidades y tipos de recursos apropiados cuando el software lleva a cabo sus funciones bajo condiciones establecidas.

3.3. Performance efficiency compliance: El grado con el cual el producto software es conforme a estándares o convenciones relacionados con la eficiencia del rendimiento.

4. Operability: El grado con el cual el producto software puede ser entendido, aprendido, utilizado y es atractivo para el usuario, cuando se utiliza bajo condiciones específicas.

4.1. Appropriateness recognisability: El grado con el cual el producto software posibilita a sus usuarios a reconocer si el software es apropiado para sus necesidades.

4.2. Learnability: El grado con el cual el producto software posibilita a los usuarios a aprender esa aplicación.

4.3. Ease of use: El grado con el cual el producto software hace fácil para los usuarios operarlo y controlarlo.

4.4. Helpfulness: El grado con el cual el producto software proviene de ayuda cuando los usuarios necesitan asistencia.

4.5. Attractiveness: El grado con el cual el producto software es atractivo para el usuario.

4.6. Technical accessibility: El grado de operabilidad del producto software para usuarios con discapacidades específicas.

4.7. Operability compliance: El grado con el cual el producto software es conforme a estándares, convenciones, guías de estilos o regulaciones relacionadas con la operabilidad.

5. Security: La protección de las partes del sistema de un acceso, uso, modificación, destrucción o revelación accidental o malicioso.

5.1. Confidentiality: El grado con el cual el producto software proviene de protección en el caso de un acceso de datos o información no autorizado, tanto accidental como intencionado.

5.2. Integrity: El grado con el cual la exactitud y completitud de los activos se ve salvaguardada.

5.3. Non-repudiation: El grado con el cual puede demostrarse que las acciones o eventos han ocurrido, de modo que las acciones o eventos no puedan ser negados con posterioridad.

5.4. Accountability: El grado con el cual las acciones de una entidad pueden ser localizadas únicamente a la entidad.

5.5. Authenticity: El grado con el cual la identidad de un sujeto o recurso puede ser probado.

5.6. Security compliance: El grado con el cual el producto software es conforme a estándares, convenciones o regulaciones relacionadas con la seguridad.

6. Compatibility: La capacidad de dos o más componentes software para intercambiar información y/o llevar a cabo sus funciones requeridas mientras comparten el mismo entorno software o hardware.

6.1. Replaceability: El grado con el cual el producto software puede ser utilizado en lugar de otro producto software específico para el mismo propósito en el mismo entorno.

6.2. Co-existence: El grado con el cual el producto software puede coexistir con otro software independiente en un entorno común compartiendo recursos comunes sin impactos perjudiciales.

6.3. Interoperability: El grado con el cual el producto software puede ser operable en cooperación con uno o más productos software.

6.4. Compatibility compliance: El grado con el cual el producto software es conforme a estándares, convenciones o regulaciones relacionados con la compatibilidad.

7. Maintainability: El grado con el cual el producto software puede ser modificado. Las modificaciones pueden incluir correcciones, mejoras o adaptación del software a cambios en el entorno, y en los requisitos y las especificaciones funcionales.

7.1. Modularity: El grado en que un sistema o programa de ordenador se compone de componentes discretos de tal manera que un cambio en un componente tiene un impacto mínimo en otros componentes.

7.2. Reusability: El grado con el cual un activo puede ser utilizado en más de un sistema software, o en la construcción de otros activos.

7.3. Analysability: El grado con el cual el producto software puede diagnosticar deficiencias o causas de fallos en el software, o identificar las partes que deben ser modificadas.

7.4. Changeability: El grado con el cual el producto software permite implementar modificaciones especificadas. La facilidad con la que un producto software puede ser modificado.

7.5. Modification stability: El grado con el cual el producto software puede esquivar efectos inesperados por modificaciones del software.

7.6. Testability: El grado con el cual el producto software permite modificar el software para ser validado.

7.7. Maintainability compliance: El grado con el cual el producto software es conforme a estándares o convenciones relacionados con la mantenibilidad.

8. Transferability: El grado con el cual el producto software puede ser transferido desde un entorno a otro.

8.1. Portability: Facilidad con la cual un sistema o componente puede ser transferido desde un entorno software o hardware a otro.

8.2. Adapability: El grado con el cual el producto software puede ser adaptado para diferentes entornos específicos sin aplicar acciones o medios distintos de los previstos para este fin para el software considerado.

8.3. Installability: El grado con el cual el producto software puede ser instalado y desinstalado satisfactoriamente en un entorno específico.

8.4. Transferability compliance: El grado con el cual el producto software sigue estándares o convenciones relacionados con la portabilidad.

4.2. Métricas para Líneas de Producto Software

En las subsecciones anteriores se ha visto diferentes modelos de calidad, los cuales se basan fundamentalmente en una descomposición de características y atributos.

En este trabajo se pretende proponer un modelo de calidad para LPS siguiendo SQuaRE. Un primer paso para su composición es conocer las métricas existentes y los atributos que miden. Se han realizado diferentes búsquedas en las bibliotecas digitales IEEEXplore, ACM Digital Library e INSPEC con el objetivo de averiguar si existe alguna revisión sistemática que recoja este conocimiento. Sin embargo, no hemos obtenido ningún resultado que coincida con el objetivo planteado. No obstante en la literatura podemos encontrar otros trabajos en los que se realiza una revisión de la literatura para descubrir y analizar las métricas existentes en otras áreas de la ingeniería del software. Calero *et al.* [12] presentan una clasificación de las métricas más importantes propuestas para los sistemas de información Web, con el objetivo de ofrecer al usuario una visión global del estado de la investigación en esa área. La clasificación se realiza utilizando WQM (Web Quality Model), un modelo de calidad tridimensional para la Web. Por otra parte, Gómez *et al.* [32] resumen en su trabajo un estado del arte de las métricas existentes para la ingeniería del software. Para descubrir las métricas utilizan una revisión sistemática, y para clasificarlas hacen uso de una Ontología para la Medición del Software. De este modo, parece razonable el uso de una revisión sistemática como método de investigación.

4.2.1. Formulación de la pregunta de investigación

El objetivo de esta revisión sistemática es obtener y analizar todas las métricas existentes para evaluar la calidad en líneas de productos software. Siguiendo la *metodología* de revisión sistemática, la pregunta de investigación que formulamos en la presente revisión es la siguiente:

¿Qué métricas han sido utilizadas por los investigadores para la evaluación de calidad en líneas de productos software y cómo han sido utilizadas?

Conforme el estándar SQuaRE [40], una métrica es el conjunto de operaciones que tienen por objetivo determinar el valor de una medida. La pregunta de investigación nos permitirá resumir el conocimiento actual acerca de las métricas existentes para la evaluación de calidad en las líneas de productos software y para identificar las carencias existentes con el fin de sugerir nuevas áreas de investigación. El estudio de la población y la intervención es el siguiente:

- *Población:* Trabajos de investigación en los que se proponen métricas para evaluar la calidad de líneas de productos software.
- *Intervención:* Métricas para evaluar la calidad en el desarrollo de líneas de productos software.
- *Resultados:* Análisis de una colección de métricas que sirvan de base para la elaboración de un modelo de calidad para líneas de productos software.
- *Diseño experimental:* Ningún diseño.

Esta revisión es más limitada que una revisión sistemática completa, como se sugiere en [46] ya que no realizamos un seguimiento de las referencias de los artículos seleccionados.

4.2.2. Selección de Fuentes

La búsqueda de estudios primarios se llevó a cabo consultando las bibliotecas digitales de las organizaciones y editoriales más relevantes para la comunidad de Ingeniería del Software. La lista de fuentes iniciales es la siguiente: IEEEXplore, ACM Digital Library, Science Direct e INSPEC.

Además, se han considerado las actas de los siguientes congresos:

- *Actas de los congresos sobre líneas de productos software*: SPLC (Software Product Line Conferences), PFE (International Workshop on Product-Family Engineering) y IWSAPF (International Workshop on Software Architectures for Product Families).
- *Actas de congresos sobre calidad del software*: ESEM (Empirical Software Engineering and Measurement), ISESE (International

Symposium on Empirical Software Engineering) y METRICS (IEEE International Software Metrics Symposium).

- *Actas de congresos sobre Ingeniería del Software*: ICSE (International Conference on Software Engineering) y ECSA (European Conference on Software Architecture).

El período de búsqueda comprende desde el año 1996 hasta principios del año 2009. Hemos escogido el año 1996 porque fue el primero en el que se celebró un congreso dedicado específicamente a las líneas de productos software.

Se realizaron búsquedas con diferentes cadenas con el fin de refinar los resultados obtenidos. La cadena finalmente seleccionada fue la siguiente:

(metric OR measur*) AND (software) AND*
("product line" OR "product famil*")*

La búsqueda se llevó a cabo utilizando el título y el resumen de los artículos. Con el propósito de conseguir que nuestro estudio fuera lo más amplio posible, añadimos literatura gris a la revisión. En concreto, añadimos estudios relacionados que ya teníamos conocimiento, así como otros estudios hallados introduciendo combinaciones de la cadena de búsqueda anterior en el motor de búsqueda *Google*.

4.2.3. Definición de criterios de Inclusión y Exclusión

Una vez realizada la búsqueda automática y manual, los estudios primarios seleccionados fueron clasificados considerando los siguientes criterios:

- *Criterios de Inclusión*: Artículos que presenten métricas para la evaluación de la calidad en líneas de productos software.
- *Criterios de Exclusión*: Artículos que no estén escritos en inglés, artículos que proponen métricas que no estén directamente relacionadas con la calidad (por ejemplo, modelos de coste), artículos que proponen métricas pero no explican cómo medirlas y artículos que definen atributos pero no sus métricas.

4.2.4. Estrategia de Extracción de Información

Para obtener la información relevante de cada artículo que dé respuesta a la pregunta de investigación planteada, se ha descompuesto dicha pregunta en los siguientes criterios (se indican las posibles respuestas para cada criterio):

1. **Característica de calidad evaluada** (Funcionalidad, Fiabilidad, Usabilidad, Eficiencia, Mantenibilidad, Portabilidad).
2. **Fase del Ciclo de Vida en el que se aplica la Métrica** (Ingeniería del Dominio (ID): Requisitos, Diseño, Realización y Pruebas;

Ingeniería de la Aplicación (IA): Requisitos, Diseño, Realización y Pruebas; Evolución).

3. **Artefacto sobre el cual se realiza la Evaluación** (Arquitectura de la línea de producto, Activo, Arquitectura del producto, Producto final).
4. **Procedimiento de Validación de la Métrica** (Validación teórica, Validación empírica, No validada).
5. **Soporte para la Medición** (Manual, Automática).

A continuación se explica cada criterio y en qué casos debe seleccionarse cada opción:

1. **Característica de calidad evaluada**. Siguiendo una estrategia similar a la empleada por Calero *et al.* para clasificar un conjunto de métricas Web [12], hemos clasificado cada métrica extraída de los artículos seleccionados según las características de calidad que pretenden medir las descritas en el estándar ISO/IEC: funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad. Cada métrica ha sido clasificada de la siguiente forma: (1) *Funcionalidad*, si mide atributos relacionados con la existencia de un conjunto de funciones y sus propiedades específicas, comprobando que las funciones satisfacen lo indicado; (2) *Fiabilidad*, si mide atributos relacionados con la capacidad de la línea de productos para mantener un nivel específico de rendimiento cuando se utiliza bajo ciertas condiciones específicas; (3) *Usabilidad*, si mide atributos relacionados con la facilidad de entendimiento, de aprendizaje o uso de ciertos artefactos de líneas de productos (4) *Eficiencia*, si mide atributos relacionados con la relación entre el nivel de desempeño de una línea de productos y la cantidad de recursos utilizados; (5) *Mantenibilidad*, si mide atributos relacionados con la facilidad de extensión, modificación o corrección de una línea de productos; (6) *Portabilidad*, si mide atributos relacionados con la facilidad de transferir una línea de productos de una plataforma a otra.

2. **Fase del ciclo de vida en el cual se aplica la métrica**. El ciclo de vida de las líneas de productos software se divide básicamente en dos fases: la fase de la Ingeniería del Dominio y la fase de la Ingeniería de la Aplicación. Cada una de estas fases se divide, además, en otras cuatro: Requisitos, Diseño, Realización y Pruebas. En este trabajo, hemos seguido el ciclo de vida propuesto por [71] para clasificar los artículos y métricas seleccionados.

De acuerdo a esta propuesta, la clasificación de métricas es la siguiente: (1) *Requisitos en la Ingeniería del Dominio,* se clasifica en esta fase si es posible aplicar la métrica sobre los artefactos relacionados a la especificación de requisitos de toda la línea de productos; (2) *Diseño en la Ingeniería del Dominio,* si la métrica evalúa la calidad del diseño de la línea de productos completa y/o la arquitectura base tomando en cuenta la variabilidad, las partes comunes, etc.; (3) *Realización en la Ingeniería del Dominio,* si la métrica realiza una evaluación sobre los activos; (4) *Pruebas en la Ingeniería del Dominio,* si la métrica evalúa la familia de productos una vez

ésta esté creada; (5) *Requisitos en la Ingeniería de la Aplicación*, si la métrica evalúa artefactos relacionados con la especificación de requisitos de un producto concreto (como escenarios o casos de uso); (6) *Diseño en la Ingeniería de la Aplicación,* si la métrica evalúa la calidad de la arquitectura del producto; (7) *Realización en la Ingeniería de la Aplicación,* si la métrica evalúa el proceso de implementación de un producto; (8) *Pruebas en la Ingeniería de la Aplicación,* si la métrica evalúa el producto final.

Finalmente, hemos incluido una última fase, la cual fue sugerida por [11]: (9) *Evolución*, una métrica es clasificada en esta fase si el atributo que evalúa está relacionado con la evaluación de los cambios introducidos con posterioridad a la creación de una línea de productos.

3. **Artefacto sobre el cual se realiza la evaluación**. Además de la fase en la que el atributo es evaluado, es importante conocer qué artefacto se evalúa. Clasificamos las métricas, en función del artefacto evaluado, de la siguiente forma: (1) *Arquitectura Base*, si evalúa la arquitectura de la línea de productos completa; (2) *Activos*, si evalúa componentes por separado; (3) *Arquitectura del Producto*, si evalúa la arquitectura de un solo producto; (4) *Producto final*, si evalúa un producto de la familia.
4. **Procedimiento de validación de la métrica**. Existen dos tipos de validación: (1) *Validación teórica*, que asegura que la métrica mide el atributo que pretende medir; (2) *Validación empírica*, que proporciona evidencia sobre la utilidad de la métrica. Normalmente la validación empírica se lleva a cabo mediante experimentos controlados que comprueban la relación existente entre los resultados obtenidos por la métrica y alguna característica de calidad externa, como las propuestas en el estándar SQUARE. Si la métrica carece de validación, se clasifica como (3) *No validada.*
5. **Soporte para la medición**. La medición y/o evaluación de la métrica puede realizarse de forma (1) *Manual*. En el caso de que exista una herramienta que de soporte a la evaluación, indicaremos que se realiza de forma (2) *Automática*.

4.2.5. Conducción de la revisión

La búsqueda de los estudios primarios en las bibliotecas digitales fue realizada el 10 de Marzo de 2009. En la búsqueda automática se identificaron 442 potenciales publicaciones relevantes (346 de IEEEXplore, 33 de ACM Digital Library, 9 de Science Direct y 54 de Inspec). Algunas publicaciones aparecieron en más de una fuente. En estos casos, se consideraron una vez con el siguiente orden de prioridad: (1) actas de congresos, (2) IEEEXplore, (3) ACM, (4) Science Direct e (5) Inspec).

Tras aplicar los criterios de inclusión y exclusión descritos en la Sección 3.3, fueron seleccionados un total de 16 artículos: 4 artículos procedentes de la búsqueda manual (1 de PFE, 1 de SPLC y 2 de

METRICS), 9 procedentes de la búsqueda automática (4 de IEEEXplore y 5 de ACM) y 3 estudios considerados literatura gris. Desafortunadamente, no encontramos artículos relevantes en las otras conferencias. Respecto al resto de las bibliotecas digitales, los artículos relevantes encontrados ya habían sido incluidos en la revisión por alguna de las fuentes con mayor prioridad.

4.2.6. Resultados de la revisión sistemática

La clasificación de las métricas extraídas de los artículos seleccionados según cada criterio se presenta en el Anexo B. Se ha indicando la referencia del artículo dónde se proponen dichas métricas.

A continuación, se presenta un análisis de los resultados para cada criterio utilizado para extraer la información relevante de los artículos seleccionados.

1. **Característica de calidad evaluada**: Los resultados han mostrado que la mayoría de las métricas extraídas de los artículos pretenden medir atributos relacionados con la Mantenibilidad (ver Figura 4.7). Por ejemplo, Aldekoa *et al.* [4] proponen una métrica para calcular el Índice de Mantenibilidad de una línea de productos; Alves de Oliveira *et al.* [3] proponen un conjunto de métricas para calcular la complejidad de la arquitectura de la línea de productos, obtenidas a partir de la herramienta SDMetrics; Ganesan *et al.* [6] proponen un conjunto de métricas para calcular el coste de la realización de pruebas en los componentes. Con relación a la eficiencia, la segunda característica con más métricas, se encuentran, entre otras, las métricas propuestas por Van der Hoek *et al.* [13] y Sun Her *et al.* [14]. Las métricas de Van der Hoek *et al.* permiten calcular ratios de utilización de servicios provistos y requeridos y evaluar la arquitectura de una línea de productos, basándose en dos características específicas de las mismas (la opcionalidad y la variabilidad). Las métricas de Sun Her *et al.* permiten evaluar la reusabilidad de los activos. En general, la mayoría de las métricas están relacionadas con la reusabilidad de componentes y la variabilidad.

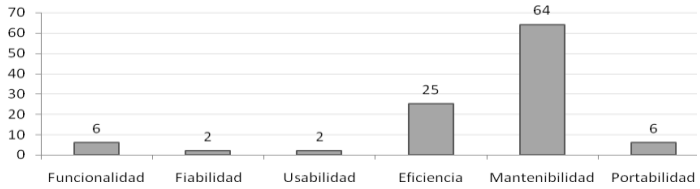

Figura 4.7. Número de métricas por característica de calidad.

2. **Fase del Ciclo de Vida en el que se aplica la Métrica**. La Figura 4.8 muestra que una gran cantidad de métricas se aplican en la fase del diseño de la Ingeniería del Dominio. Esto es debido a que muchas de ellas están enfocadas a características concretas de las líneas de productos, como la reusabilidad o la variabilidad, aspectos que se tienen muy presente en

esta fase del ciclo de vida. La segunda fase más evaluada es la Evolución. Por ejemplo, Ajila *et al.* [2] identifican un conjunto de métricas que pueden ser utilizadas para recordar los cambios en la línea de productos, con las cuales podrá observarse su evolución.

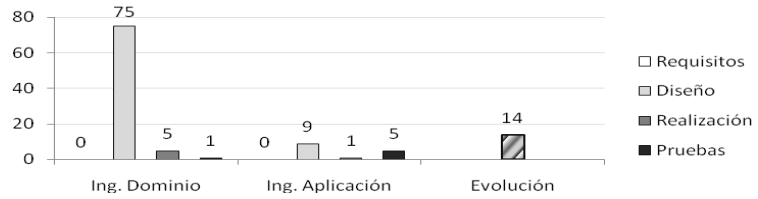

Figura 4.8. Número de métricas aplicables a cada fase del ciclo de vida.

3. **Artefacto sobre el cual se realiza la Evaluación**. Como se puede observar en la Figura 4.9, la mayoría de las métricas miden artefactos obtenidos en etapas tempranas del proceso de desarrollo de una línea de productos: el 54% de las métricas se aplican sobre la arquitectura de la línea de productos mientras que un 27% se aplican sobre los activos. En algunos casos, se utiliza más de un artefacto para realizar la medición. Por ejemplo, Johansson y Höst [8] hacen uso del código fuente, archivos y otros artefactos para comprobar las violaciones del diseño en el producto.

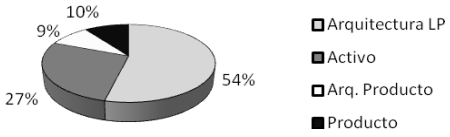

Figura 4.9. Porcentaje de métricas según artefacto software que evalúa.

4. **Procedimiento de Validación de la Métrica**. Los resultados han mostrado que el 87% de las métricas carecen de cualquier tipo de validación. Sólo un 12% de ellas han sido validadas teóricamente. Éstas son las métricas propuestas por Sun Her *et al.* [14], las cuales han sido, por una parte, analizadas teóricamente mediante la aproximación propuesta por Kitchenham *et al.* [46], y por otra, aseguradas mediante seis criterios derivados del estándar IEEE Std 1061 [38]. Únicamente la métrica *"Error propagation"* propuesta por Abdelmoez *et al.* [1] ha sido validada empíricamente. La validación ha consistido en un estudio comparativo entre los resultados analíticos y los conseguidos experimentalmente. Sin embargo, ninguna métrica propuesta ha validado empíricamente la relación existente entre los valores obtenidos por la métrica y una característica de calidad externa.

5. **Soporte para la Evaluación**. El 80% de las métricas no disponen de una herramienta que de soporte a su medición. Como ejemplos de herramientas para la evaluación del restante 20% de las métricas, se encuentran la combinación de la herramienta Crystal Ball con la simulación Monte-Carlo para analizar los resultados de las métricas propuestas por [6] en un documento Excel; o Zhang *et al.* [16] que definen sus métricas sobre

una especificación formal de la arquitectura de línea de productos con la especificación vADL, la cual permite hacer un análisis automático.

4.2.7. Amenazas a la Validez

Las posibles limitaciones de este estudio están relacionadas con el sesgo de la selección de las publicaciones, el posible uso de una cadena de búsqueda incompleta, y la inexactitud en la extracción de datos o errores en la clasificación.

El sesgo de publicación se refiere al problema de que los resultados positivos tienen más probabilidades de ser publicados que los resultados negativos [46]. Las bibliotecas digitales contienen un gran número de revistas y actas de congresos. Sin embargo, no consideran *literatura gris* (por ejemplo, informes de la industria, tesis doctorales) o resultados no publicados. Esperamos haber disminuido esta amenaza mediante la exploración manual de actas de congresos y estudios considerados como *literatura gris*.

Con respecto a la selección de las publicaciones, tratamos de elegir las fuentes en las que suelen publicarse artículos relacionados con métricas para el desarrollo del software y líneas de productos.

Referente a la cadena de búsqueda, procuramos recoger todas las cadenas que son representativas de la pregunta de investigación. Refinamos la búsqueda en varias ocasiones en base a los resultados obtenidos para optimizar el conjunto de trabajos seleccionados por las bibliotecas digitales. Además, consideramos los posibles sinónimos e incluimos únicamente el lexema de las palabras para cubrir la búsqueda de toda la familia léxica.

Intentamos suavizar las amenazas de la inexactitud en la extracción de datos mediante la realización de las clasificaciones de los trabajos con dos revisores. Las discrepancias entre las evaluaciones fueron discutidas y consensuadas.

4.3. Conclusiones

Se ha presentado una revisión sistemática de la literatura con el propósito de localizar y obtener información relevante sobre las métricas utilizadas por los investigadores en los últimos años para la evaluación de la calidad de líneas de productos software.

Los resultados muestran que la mayoría de las métricas se utilizan en la fase del ciclo de vida de las líneas de productos software correspondiente al diseño en la ingeniería del dominio (68%) y que los artefactos sobre los que más se aplican estas métricas son la arquitectura de la línea de productos (54%) y los activos (27%). Hemos tenido en cuenta el modelo de calidad SQuaRE propuesto en la ISO/IEC 25000 en nuestra clasificación, observando que la mayoría de las métricas miden

características relacionadas con la mantenibilidad (61%), seguida de la eficiencia (24%). Desafortunadamente, sólo un 1% de las métricas han sido validadas empíricamente y sólo un 12% teóricamente. Existe la necesidad de estudios empíricos que validen dichas métricas. Finalmente, se observa que la mayoría de las métricas (80%) no disponen de una herramienta que facilite su medición o evaluación.

Debido a la utilidad de los modelos de calidad, parece interesante proponer uno que recoja las métricas y los atributos localizados, sin embargo es necesario proponer más métricas para el resto de características de calidad y realizar una validación de las métricas.

Capítulo 5

Modelo de Calidad para LPS

A partir de los estudios realizados para conocer el estado actual sobre la evaluación de la calidad en LPS que se ha presentado en los Capítulos 3 y 4, en este capítulo, se propone un Modelo de Calidad específico para líneas de producto software en el que recogemos las características de calidad más relevantes de la Ingeniería de LPS y de la Ingeniería del Software en general. Además, la propuesta se ha realizado dentro del marco del estándar de calidad más actual propuesto por la ISO/IEC: SQuaRE.

5.1. Definición del Modelo de Calidad

En esta sección definimos un modelo de calidad para líneas de producto software. El desarrollo del modelo de calidad propuesto consiste en las siguientes actividades: definición de objetivos, especificación de características de calidad, especificación de relaciones, y la operacionalización del modelo. Estas actividades son descritas a continuación.

5.1.1. Definición de los objetivos de calidad

En esta actividad, se tratan las características que debería cubrir la evaluación y otros temas relacionados con el modelado de la calidad tales como los objetos (artefactos) y los stakeholders (puntos de vista) que deben ser claramente definidos. De acuerdo al paradigma Goal-Question-Metric (GQM) [ref. 2 artículo Silvia], los objetivos de nuestro modelo de calidad se resumen en la Tabla 5.1.

Tabla 5.1. Definición de los objetivos del modelo de calidad.

	Descripción	Definición
Objeto	¿Qué artefacto(s) debe ser analizado(s)?	Especificación de requisitos, Arquitectura de la LP, Activos, Arquitectura del Producto, Producto final, Casos de Prueba
Propósito	¿Por qué debería ser el objeto analizado?	Para evaluar su calidad y proporcionar retroalimentación al diseño.
Enfoque de la calidad	¿Qué características del objeto deben ser analizadas?	Las características definidas en el estándar de calidad SQuaRE y las propias de las LPS.
Punto de vista	¿Quién utilizará la información recogida?	Desde el punto de vista del investigador y de la empresa.
Contexto	¿En qué entornos se realiza el análisis?	En el contexto del grupo de investigación ISSI y de empresas interesadas en la aplicación del modelo en la práctica.

A continuación vamos a comentar brevemente cada uno de los puntos que aparecen en la Tabla 5.1.

Los **objetos** que deben ser analizados son todos aquellos artefactos que están implicados activamente en el desarrollo de la línea de productos y el producto final. El modelo de calidad debe proporcionar mecanismos para cubrir la evolución de los artefactos producidos en todo el ciclo de vida del desarrollo de la línea de productos software (desde la especificación de requisitos hasta la implementación del producto final).

De acuerdo al ciclo de vida de las líneas de productos software, podemos tener especificaciones de requisitos para dos fines: (1) especificar los requisitos de toda la familia de productos, o (2) especificar los requisitos para un producto concreto de acuerdo a la variabilidad ofrecida en la LP. En el caso de la especificación de requisitos de la línea de

productos, los artefactos involucrados son los propios de la fase de requisitos (diagramas de casos de uso, diagramas entidad relación, etc.) y deben especificar los requisitos de toda la línea de productos. En el caso de la especificación de requisitos para un producto concreto, los artefactos involucrados son los propios de la fase de requisitos, los cuales recogen un subconjunto de los posibles requisitos de la línea de productos, añadiendo, si procede, nuevos requisitos que conforman las características propias del producto. Por lo tanto, el modelo debería proporcionar atributos y métricas para evaluar la calidad de ambos tipos de especificaciones de requisitos.

La arquitectura de la LP es otro artefacto que debe evaluarse, pues conforma la base común de todos los productos. Los errores que pudieran cometerse en la especificación/elaboración de la arquitectura de la LP podrían impactar en la arquitectura del producto final. De ahí que la evaluación de este artefacto sea tan importante.

Los activos constituyen los bloques de construcción con los que van a construirse los sistemas software finales, por lo que será necesario evaluar que éstos contienen la funcionalidad esperada, que funcionan correctamente, que pueden comunicarse correctamente con otros activos, etc.

Por último, el modelo de calidad debe proporcionar atributos y métricas para evaluar la arquitectura del producto final, los activos creados o modificados a consecuencia de las características propias del producto y, finalmente, evaluar el producto finalizado.

El **propósito** del modelo de calidad no es sólo el de evaluar la calidad, sino también de asegurarla y ofrecer retroalimentación al diseño. Por lo tanto, el modelo debería ser utilizado como parte de un método de evaluación que asegure la calidad de los artefactos evaluados. Respecto a la retroalimentación al diseño, se refiere a que no sólo se va a ofrecer información acerca de la evaluación, sino que también se debe indicar al diseñador información sobre qué partes de la arquitectura, activos, producto final, etc. deben ser modificadas para garantizar los atributos de calidad deseados.

Ya que uno de los objetivos planteados con respecto al modelo de calidad es que éste siga estándares (en concreto el estándar SQuaRE), las características mínimas que deben ser evaluadas, es decir, el **enfoque de la calidad**, son las descritas en este estándar. Además, también se pretende que el modelo recoja aquellas características que son propias de las LPS.

El modelo de calidad se define desde el **punto de vista** del investigador. Aunque la calidad es un factor clave en el desarrollo del software, no siempre es tenido en cuenta. Por lo que uno de los objetivos es que el modelo de calidad sea lo más comprensible y sencillo de utilizar, para que pueda ser extensamente empleado.

El modelo se define en el **contexto** del grupo de investigación en Ingeniería del Software y Sistemas de Información (ISSI) y debe esperarse que pueda ser utilizado por otros investigadores. Además, se pretende que también pueda ser utilizado por la industria.

5.1.2. Especificación de Características de calidad

La especificación del Modelo de Calidad para LPS propuesto parte de la descripción de las ocho características y sus correspondientes subcaracterísticas de primer nivel del estándar de calidad SQuaRE para la evaluación del producto software. Un producto software está definido en un sentido amplio como los ejecutables, código fuente, descripciones de arquitectura, etc. Como resultado, la noción de usuario se amplía tanto a operadores como a programadores, usuarios de componentes y el arquitecto software.

A partir de este modelo, se han incluido atributos medibles que puedan cuantificar los valores de las características. Un atributo es medible cuando es posible definir una o más métricas para cuantificarlo. Además, en los casos en los que ha sido necesario, se ha incluido otro nivel de subcaracterísticas (tercer nivel) para que la jerarquía de características quede esquematizada de forma más clara.

5.1.2.1. Idoneidad Funcional

Las líneas de productos software se basan en la especificación y uso de un conjunto de activos, cada uno de los cuales contiene sus propias funcionalidades, con los que posteriormente se constituirá un producto. Los activos pueden ser construidos expresamente para la línea de productos, o reutilizados. Tanto si son de nueva creación o de reutilización, es imprescindible garantizar que la funcionalidad que los activos realizan se corresponde con la funcionalidad esperada, es decir, asegurar la *adecuación de un activo a la línea de productos*. En el caso de un activo de nueva creación, la necesidad de comprobar esta característica coincide con la creación de un componente software habitual (*completitud de la implementación funcional*). En el caso de un activo reutilizado, además de comprobar su funcionalidad es importante conocer si tiene otras funcionalidades que no están especificadas en los requisitos de la línea de productos, es decir, si *soporta funciones no requeridas*. Las funcionalidades no especificadas podrían estar consumiendo recursos, tiempo o espacio que perjudique el funcionamiento de un producto final, con lo que en estos casos tal vez sería preferible crear un activo nuevo o modificar uno existente. En otras ocasiones, estas funcionalidades no especificadas podrían enriquecer la línea de productos sin coste alguno ya que el activo ya está implementado (*aportación de funciones no requeridas por un activo*).

Los activos reutilizados puede ser que tengan la funcionalidad requerida, pero no respondan exactamente como se espera que lo haga,

por ejemplo, es posible que esté devolviendo la información solicitada pero en un formato que no es compatible. En estos casos, sería importante evaluar la cantidad de funciones que deben modificarse para que la funcionalidad del activo sea exactamente la especificada, es decir, conocer la cantidad de *funciones parcialmente especificadas o implementadas*. Este atributo puede confundirse con la completitud de la implementación funcional. La diferencia entre ambos es que la completitud de la implementación funcional se refiere a la cantidad de funciones completamente desarrolladas contra el número total de funciones que deben desarrollarse, mientras que las funciones parcialmente especificadas o implementadas se refiere al número de funciones que responden en parte a la funcionalidad requerida pero que necesitan de ciertas modificaciones para que puedan ser añadidas a la línea de productos.

Otro aspecto a tener en cuenta para las líneas de productos software es el *nivel de cubrimiento*. Considerando el análisis de la tasa de ganancia (proporción de dinero ganado o perdido en una inversión en relación con la cantidad de dinero invertido), o ROI (Return On Investment), es muy costoso preparar todos los artefactos desarrollados o usados en el ciclo de vida de una línea de productos software [39]. Por lo que el nivel de cubrimiento debe conocerse y adaptarse a las condiciones de la organización y de la línea de productos. En este modelo, se proponen los siguientes atributos para medir el nivel de cubrimiento de la línea de productos: *nivel de cubrimiento de los artefactos* y *nivel de cubrimiento de los activos* (ver tabla 5.2). El *nivel de consistencia* describe cómo de consistentes son un conjunto de activos en una línea de productos. Esto indica que los activos no se contradigan entre sí [39].

Tabla 5.2. Subcaracterísticas y atributos para Idoneidad Funcional.

Subcaracterística		**Atributo**	**Significado**
1.1. Adecuación	1.1.1. Adecuación de un activo a la línea de productos	1.1.1.1. Completitud de la implementación funcional	Apropiación de las funciones desarrolladas completamente con las que se necesitan.
		1.1.1.2. Soporte a funciones no requeridas	Cantidad de funciones de un activo que no son necesarias.
		1.1.1.3. Aportación de funciones no requeridas por un activo	Cantidad de funciones de un activo, que aunque no estaban especificadas en los requisitos, enriquecen la línea de productos.
		1.1.1.4. Funciones parcialmente especificadas o implementadas	Cantidad de funciones de un activo que coinciden parcialmente con los requisitos.

Subcaracterística		Atributo	Significado
		1.1.1.4. Corrección de la implementación funcional	Correctitud de las funciones que han sido especificadas y/o implementadas.
	1.1.2. Adecuación de un producto a la LPS	1.1.2.1. Adecuación de los requisitos funcionales	Adecuación de los requisitos funcionales de un producto a la LPS
	1.1.3. Nivel de cubrimiento de la línea de productos.	1.1.3.1. Nivel de cubrimiento de los activos	Cantidad de activos que son especificados y/o implementados en la línea de productos.
		1.1.3.2. Nivel de cubrimiento de otros artefactos	Cantidad de artefactos (sin incluir los activos) que son especificados y/o implementados en la línea de productos.
		1.3.4. Nivel de Consistencia	Consistencia de un conjunto de activos de la línea de productos.
1.2. Exactitud		1.2.1. Precisión computacional	Como de precisa ha sido la implementación de los requisitos especificados.
1.3. Adherencia a normas		1.3.1. Adecuación a otras normas	Grado de adecuación a otros estándares, convenciones o reglamentaciones relacionadas a la idoneidad funcional de LPS.

La *exactitud* se refiere al grado con que un producto software proporciona los resultados correctos con el grado de precisión esperado. Esa subcaracterística es tratada como cualquier otro producto software, a diferencia que puede ser medida para diferentes artefactos y fases del ciclo de desarrollo (exactitud de un activo, de la línea de productos, del producto final, etc.). Por último, la *adherencia a normas* considera evaluar el grado de adecuación a otros estándares, convenciones o reglamentaciones relacionadas a la idoneidad funcional. En la Tabla 5.2 se muestran todas las subcaracterísticas y atributos relativos a la Idoneidad funcional.

5.1.2.2. Fiabilidad

La fiabilidad (Reliability) es la característica de calidad que se preocupa del grado con el que un producto software puede mantener un cierto nivel de rendimiento cuando es utilizado bajo condiciones específicas. Al igual

que para cualquier sistema software esta propiedad es aún más importante para las líneas de productos software en lo referido a su arquitectura y la base común de la que se compone. Las líneas de productos software se forman a partir de una arquitectura común a la que se van uniendo activos que conforman la variabilidad de los productos. Por lo tanto, asegurar un cierto nivel de fiabilidad en la base común y en los activos que componen los productos, a priori, garantiza un cierto nivel de fiabilidad para el producto final. El modelo de calidad relativo a la Fiabilidad se muestra en la Tabla 5.3.

Tabla 5.3. Subcaracterísticas y atributos para Fiabilidad.

Subcaracterística		Atributo	Significado
2.1. Disponibilidad	2.1.1. Ratio de llamadas a funciones satisfactorias	2.1.1.1. Ratio de llamadas a funciones satisfactorias de un activo	Porcentaje de llamadas a funciones que han funcionado correctamente en un activo frente al número de llamadas totales.
		2.1.1.2. Ratio de llamadas a funciones satisfactorias de un producto	Porcentaje de llamadas a funciones que han funcionado correctamente en un producto (componente, arquitectura,…) frente al número de llamadas totales.
2.2. Tolerancia a fallos	2.2.1. Tratamiento de errores	2.2.1.1. Tratamiento de errores de un activo	Cuantos posibles errores han sido tratados bajo control en un activo para evitar fallos serios y críticos.
		2.2.1.2. Tratamiento de errores de un producto	Cuantos posibles errores han sido tratados bajo control en un producto específico para evitar fallos serios y críticos.
	2.2.2. Propagación de errores	2.2.2.1. Propagación de errores producidos en la arquitectura de la LP	Cantidad de posibles errores producidos en la base común de la LP susceptibles de propagarse al ejecutar un producto.
		2.2.2.2. Propagación de errores producidos en un activo	Cantidad de posibles errores producidos en un activo susceptibles a propagarse por otros activos al ejecutar un

Subcaracterística		Atributo	Significado
			producto.
2.3. Recuperabilidad	2.3.1. Restaurabilidad	2.3.1.1. Restaurabilidad de un activo	Capacidad de un activo a restaurarse el mismo después de un evento o petición anormal.
		2.3.1.2. Restaurabilidad de un producto	Capacidad de un producto a restaurarse el mismo después de un evento o petición anormal.
2.4. Adherencia a normas		2.4.1. Adecuación a otras normas	Grado de adecuación a otros estándares, convenciones o reglamentaciones relacionadas a la fiabilidad de LPS

En SQuaRE la Fiabilidad se divide en tres subcaracterísticas: *disponibilidad*, *tolerancia a fallos*, y *recuperabilidad*. En las líneas de productos software puede medirse estas tres subcaracterísticas a nivel de los activos y del producto final. Se han especificado claramente los atributos a medir para los activos y el producto final, debido a que evaluar la línea de productos sería un valor derivado de todos los activos involucrados, pero el valor obtenido sería un valor global y orientativo de la posible calidad de esa línea de productos, pero no tiene porque indicar el posible valor de los productos derivados. Puede ocurrir que la línea de productos obtenga un valor global de fiabilidad no deseado pero que los productos generados tengan un mejor nivel de fiabilidad que los generados en otra línea de productos que prometía mejores resultados. Esto podría ocurrir si los valores negativos de fiabilidad provienen de activos que, por la especificación de la línea de productos, nunca puedan coexistir en el mismo producto.

5.1.2.3. Eficiencia

La eficiencia proporciona un conjunto de atributos relacionados con la relación entre el nivel de desempeño del software y la cantidad de recursos necesarios bajo condiciones establecidas. La eficiencia es dividida en la SQuaRE en las siguientes subcaracterísticas: *comportamiento en el tiempo (time behaviour)*, *utilización de recursos* y *adherencia a estándares*. En la Tabla 5.4 se muestra la descomposición del modelo de calidad propuesto para esta característica.

En el modelo de calidad se ha propuesto dividir la subcaracterística *comportamiento en el tiempo*, en las subcaracterísticas *tiempo de respuesta* (tiempo necesario para completar una tarea específica), *rendimiento* (cantidad de tareas que es posible realizar por unidad de tiempo) y *tiempo de espera*

(tiempo de respuesta para ejecutar un conjunto de tareas). Cada una de estas subcaracterísticas ha sido adaptada a los casos concretos de las LPS, donde es relevante conocer estas subcaracterísticas para un *activo independiente*, un *activo dependiente* o un *producto*. Por una parte, es relevante conocer el valor de estas subcaracterísticas para un determinado activo, ya que garantizar la eficiencia de los activos a priori garantiza un cierto nivel de eficiencia del producto que se cree a partir de ellos. Puede ocurrir que un activo por si solo tenga un buen nivel de eficiencia, pero que necesite de otros activos para completar ciertas funciones, de manera que la ejecución en serie de los activos de lugar a bajos niveles de eficiencia. Es por ello, que en la evaluación se han distinguido dos tipos de activos: los independientes, es decir, aquellos que no necesitan las funciones de otros activos para ejecutarse; y los dependientes, es decir, aquellos que necesitan de las funciones de otros activos para completar ciertas tareas.

Tabla 5.4. Subcaracterísticas y atributos para Eficiencia.

Subcaracterística		Atributo	Significado
3.1. Comportamiento en el tiempo	3.1.1. Tiempo de respuesta	3.1.1.1. Tiempo de respuesta de un activo independiente	Tiempo estimado o calculado que necesita un activo autosuficiente en completar una tarea específica.
		3.1.1.2. Tiempo de respuesta de un activo dependiente	Tiempo estimado o calculado que necesita un activo dependiente de otro(s) activo(s) para completar una tarea específica.
		3.1.1.3. Tiempo de respuesta de un producto	Tiempo estimado o calculado que necesita un producto para completar una tarea específica.
	3.1.2. Rendimiento	3.1.2.1. Tiempo de respuesta de un activo independiente	Número estimado o calculado de tareas que pueden ser llevadas a cabo por un activo independiente por unidad de tiempo.
		3.1.2.2. Tiempo de respuesta de un activo dependiente	Número estimado o calculado de tareas que pueden ser llevadas a cabo por un activo dependiente de otro(s) activo(s) por unidad de tiempo.
		3.1.2.3. Tiempo de respuesta de un producto	Número estimado o calculado de tareas que pueden ser llevadas a cabo por un producto

Subcaracterística		Atributo	Significado
			por unidad de tiempo.
	3.1.3. Tiempo de espera	3.1.3.1. Tiempo de respuesta de un activo independiente	Tiempo de respuesta estimado o calculado para que un activo independiente complete un grupo de tareas relacionadas con un trabajo.
		3.1.3.2. Tiempo de respuesta de un activo dependiente	Tiempo de respuesta estimado o calculado para que un activo dependiente de otro(s) activo(s) complete un grupo de tareas relacionadas con un trabajo.
		3.1.3.3. Tiempo de respuesta de un producto	Tiempo de respuesta estimado o calculado para que un producto complete un grupo de tareas relacionadas con un trabajo.
3.2. Utilización de recursos	3.2.1. Cantidad de recursos utilizados por un activo o un producto	3.2.1.1. Utilización de I/O de un activo o producto	Estimación o cálculo de utilización de I/O de un activo o un producto para completar una tarea específica.
		3.2.1.2. Utilización de memoria por un activo o producto	Estimación o cálculo del tamaño de memoria que un activo o producto ocupará para completar una tarea específica.
		3.2.1.3. Utilización de otros activos por un activo	Estimación o cálculo de la cantidad activos utilizados por un activo para llevar a cabo una tarea específica.
		3.2.1.4. Utilización de otros recursos por un activo o producto	Estimación o cálculo de la cantidad de recursos utilizados por un activo o por un producto para llevar a cabo una tarea específica. Nota: Otros recursos se refiere a recursos no tenidos en cuenta en el resto de atributos.
	3.2.2. Uso compartido de recursos por	3.2.2.1. Utilización compartida de I/O	Estimación o cálculo de utilización de I/O que dos o más activos

Subcaracterística		Atributo	Significado
	los activos		susceptibles de coexistir en el mismo producto, requieren compartir para completar una tarea específica.
		3.2.2.2. Utilización de memoria compartida	Estimación o cálculo de utilización de memoria que dos o más activos susceptibles de coexistir en el mismo producto, requieren compartir para completar una tarea específica.
		3.2.2.3. Utilización de activos compartidos	Estimación o cálculo de utilización de otro(s) activo(s) que dos o más activos susceptibles de coexistir en el mismo producto, requieren compartir para completar una tarea específica.
		3.2.2.4. Utilización de otros recursos compartidos	Estimación o cálculo de utilización de otros recursos que dos o más activos susceptibles de coexistir en el mismo producto, requieren compartir para completar una tarea específica. Nota: Otros recursos se refiere a recursos no tenidos en cuenta en el resto de atributos.
3.3. Adherencia a normas		3.3.1. Adecuación a otras normas	Grado de adecuación a otros estándares, convenciones o reglamentaciones relacionadas a la eficiencia de LPS.

La *utilización de recursos* es dividida en las subcaracterísticas *cantidad de recursos utilizados por un activo o un producto* y *uso compartido de recursos por los activos*. Es significativo conocer la *cantidad de recursos utilizados tanto por un activo como por un producto*. Se ha dividido en cuatro atributos en función del tipo de recurso: *Utilización de I/O*, *Utilización de memoria*, *Utilización de otros activos* u *Otros recursos*. En particular, el atributo Utilización de otros activos sólo es evaluado para un activo y está ligado con la posibilidad de que un activo requiera de otros activos para completar sus tareas, tal y como se

indicaba en la subcaracterística comportamiento en el tiempo. La subcaracterística *Uso compartido de recursos por los activos* evalúa la cantidad de recursos que dos o más activos comparten para llevar a cabo sus tareas. Al igual que para la subcaracterística Cantidad de recursos utilizados por un activo o un producto, se han dividido los tipos de recursos en los siguientes: *Utilización compartida de I/O*, *Utilización de memoria compartida*, *Utilización de activos compartidos*, y *Utilización de otros recursos compartidos*.

5.1.2.4. Usabilidad (Operability)

La usabilidad (referida en la SQUARE como operabilidad) se refiere al grado en que un producto software puede ser *entendido*, *aprendido*, *usado* y *atractivo* a los usuarios, usado bajo condiciones específicas. Preferimos utilizar el término "usabilidad" antes que "operabilidad" puesto que es un término más conocido en la práctica. La usabilidad se descompone en un conjunto de subcaracterísticas y atributos descritos en la Tabla 5.5.

Se ha intentado recoger los atributos más relevantes para evaluar la usabilidad y medirlos tanto a nivel de activos como de producto final. Es cierto que en las líneas de producto software el grado de usabilidad de un producto final vendrá determinado por la usabilidad de los activos software que se utilizan para componer los productos finales. Sin embargo, no es suficiente conformarse con la evaluación de usabilidad a ese nivel, ya que podría darse el caso de tener unos activos muy usables cuya composición resultara un producto poco usable.

Tabla 5.5. Subcaracterísticas y atributos para Usabilidad

Subcaracterística		Atributo	Significado
4.1. Facilidad de entendimiento	4.1.1. Descriptividad de las funciones	4.1.1.1. Completitud de la descripción del activo	Proporción de funciones (o tipos de función) que son descritas en la descripción del activo.
		4.1.1.2. Capacidad de demostración	Proporción de activos que requieren capacidad de demostración.
	4.1.2. Legibilidad visual	4.1.2.1. Uniformidad entre las interfaces de la LP	Proporción de interfaces de usuario uniformes en cuanto a distribución de las funciones, colores, etc.
		4.1.3. Funciones evidentes para un usuario	Proporción de funciones del producto que son evidentes para el usuario.
4.2. Facilidad de aprendizaje		4.2.1. Completitud de la documentación de usuario	Proporción de funciones que son descritas en la

Subcaracterística		Atributo	Significado
			documentación del usuario de un producto final o un activo.
		4.2.2. Entendibilidad de un activo	Capacidad de un activo de ser fácilmente entendido
4.3. Facilidad de uso	4.3.1. Claridad	4.3.1.1. Claridad de los elementos de las interfaces	Proporción de elementos de la interfaz que son auto-explicativos.
		4.3.1.2. Claridad de los mensajes	Proporción de mensajes que son auto-explicativos.
	4.3.2. Cancelabilidad de acciones	4.3.2.1. Cancelabilidad de funciones por el usuario en un product final	Proporción de funciones que pueden ser canceladas antes de ser completadas.
		4.3.2.2. Funcionalidad deshacer	Proporción de funciones que pueden ser deshechas.
		4.3.3. Grado de personalización del producto	Proporción de funciones que pueden ser personalizadas en un producto final durante la operación
		4.3.4. Consistencia operacional de la LPS	Proporción de operaciones en un activo con comportamiento similar en otros activos.
4.4. Facilidad de ayuda		4.4.1. Completitud de las facilidades de ayuda de las funciones.	Proporción de funciones que tienen facilidades de ayuda, tanto a nivel de activo como de producto final.
		4.4.2. Completitud de las funciones de ayuda.	Grado de completitud de la explicación de la función (muy explicada, con ejemplos, etc.), a nivel de activo y de producto final
		4.4.3. Facilidad de ayuda de la LPS	Grado en el que la LPS y los productos finales contienen activos para la facilidad de ayuda.
4.5. Grado de		4.5.1. Atracción de la	Grado de atracción de

Subcaracterística	Atributo	Significado
atracción	interfaz	la interfaz (del activo o del producto final) considerada por los usuarios.
	4.5.2. Personalización de la apariencia de la interfaz de usuario	Proporción de los elementos de la interfaz de usuario (a nivel de activo o de producto) que pueden ser personalizados (considerando la apariencia)
4.6. Accesibilidad técnica	4.6.1. Accesibilidad física	Proporción de funciones en el activo o el producto final que pueden ser personalizadas para el acceso de usuarios con discapacidades físicas.
	4.6.2.Accesibilidad técnica de la LPS	Existencia de activos que incluyen accesibilidad técnica.
	4.6.3.Accesibilidad técnica en un activo	Número de funciones en un activo que incluyen accesibilidad técnica.
	4.6.4. Duplicación de activos para accesibilidad técnica	Número de tipos de activos duplicados para ofrecer accesibilidad técnica.
4.7. Adherencia a normas	4.6.1. Adecuación a otras normas	Grado de adecuación a otros estándares, convenciones o reglamentaciones relacionadas a la usabilidad de LPS.

5. Seguridad

La seguridad es la protección de un sistema encargada de que no existan usos, modificaciones o eliminación de la información por parte de accesos accidentales o maliciosos. En la ISO SQuaRE esta característica se divide en seis subcaracterísticas: Confidencialidad, Integridad, No repudiación, Responsabilidad, Autentificación y Adherencia a estándares. La Tabla 5.6 muestra la descomposición propuesta en subcaracterísticas y atributos .

La *Confidencialidad* ha sido descompuesta en tres subcaracterísticas: Seguridad de las contraseñas, Encriptación de datos, y Alcanzabilidad de información privada sin permisos. La *Seguridad de las contraseñas* puede

evaluarse con el *Grado de seguridad de una contraseña* (grado que debe tener como mínimo una contraseña, referido al mínimo número de caracteres, tipos de caracteres, caducidad de la contraseña, etc.) y *Coherencia de contraseñas entre activos* (para permitir que múltiples activos puedan compartir la misma contraseña si pertenecen al mismo producto y al mismo nivel de seguridad). La *Encriptación de datos* se refiere al ratio de datos encriptados respecto a la cantidad de datos totales. Estos datos pueden ser de *un activo*, de la información enviada y recibida durante la *comunicación entre activos*, y de los *datos de un producto*. La *Alcanzabilidad de información privada sin permisos* comprueba que no puedan accederse, modificarse o eliminarse información si un usuario no tiene permisos. La evaluación se realiza a nivel de activos y de producto. A nivel de activos, se evalúa si es posible alcanzar información privada de un activo que requiere autentificación, desde otro que no la requiere o tiene un nivel de seguridad inferior (*Alcanzabilidad de información privada desde otro activo*). A nivel de producto, se evalúa si es posible acceder a información privada por un usuario no autentificado o sin permisos (*Alcazabilidad de información privada sin permisos en un producto*).

Tabla 5.6. Subcaracterísticas y atributos para Seguridad.

Subcaracterística		**Atributo**	**Significado**
5.1. Confidencialidad	5.1.1. Seguridad de las contraseñas	5.1.1.1. Grado de seguridad de una contraseña	Seguridad que debe tener como mínimo una contraseña.
		5.1.1.2. Coherencia de contraseñas entre activos	Utilización de la misma contraseña para activos con el mismo nivel de seguridad que pueden formar un producto.
	5.1.2. Encriptación de datos	5.1.2.1. Encriptación de datos de un activo	Completitud en la implementación de la encriptación de datos de un activo.
		5.1.2.2. Encriptación en la comunicación entre activos	Completitud en la implementación de la encriptación de información en la comunicación entre activos.
		5.1.2.3. Encriptación de datos de un producto	Completitud en la implementación de la encriptación de datos de un producto.
	5.1.3. Alcanzabilidad de información privada sin	5.1.3.1. Alcanzabilidad de información privada desde otro activo	Alcanzabilidad de información privada por la llamada de una función entre activos con distinto nivel de

Subcaracterística		Atributo	Significado
	permisos		seguridad.
		5.1.3.2. Alcanzabilidad de información privada sin permisos en un producto	Acceso a información privada por un usuario no autorizado (usuario sin permisos) en un producto.
5.2. Integridad	5.2.1. Integridad de los activos	5.2.1.1. Integridad de la especificación de un activo	Aseguramiento de que los activos no pueden ser manipulados.
	5.2.2. Integridad de los productos	5.2.2.1. Integridad de la especificación de un producto	Aseguramiento de que un producto no puede ser manipulado.
5.3. No repudiación	5.3.1. Historial de acciones de un activo	5.3.1. Existencia de historial de acciones en un activo	Existencia de un historial de las acciones que se han llevado a cabo en el sistema.
		5.3.2. Acceso al historial de acciones de un activo	Aseguramiento de que los usuarios no puedan acceder al historial para modificarlo.
		5.3.3. Encriptación de la información del historial de un activo	Grado de encriptación de la información almacenada en el historial de acciones.
	5.3.2. Historial de acciones de un producto	5.3.1. Existencia de historial de acciones en un producto	Existencia de un historial de las acciones que se han llevado a cabo en el sistema.
		5.3.2. Acceso al historial de acciones en un producto	Aseguramiento de que los usuarios no puedan acceder al historial para modificarlo.
		5.3.3. Encriptación de la información del historial	Grado de encriptación de la información almacenada en el historial de acciones.
5.4. Responsabilidad	5.4.1. Responsabilidad de un activo	5.4.1.1. Uso de funciones de un activo sin conocimento del activo	Grado de utilización de funciones de un activo (por un usuario, una función de otro activo, etc.) sin conocimento del activo
5.5. Autentificación	5.5.1. Autentificación en un activo	5.5.1.1. Información personal requerida en un activo	Cantidad y relevancia de la información personal requerida para la autentificación.
		5.5.1.2. Número de activos que requieren autentificación	Cantidad de activos de la línea de productos que requieren autentificación.

Subcaracterística		Atributo	Significado
	5.5.2. Autentificación en un producto	5.5.1.1. Información personal requerida en un producto	Cantidad y relevancia de la información personal requerida para la autentificación.
		5.5.2.1. Finalización de sesión.	Aseguramiento de que cuando un usuario finaliza el uso de la aplicación, su cuenta es cerrada.
5.6. Adherencia a normas		5.6.1. Adecuación a otras normas	Grado de adecuación a otros estándares, convenciones o reglamentaciones relacionadas a la seguridad en LPS

La *Integridad* ha sido descompuesta en la *Integridad de los activos* e *Integridad de los productos*, para referirse al aseguramiento de que no sean modificados ni los activos por separado, ni el producto que ha sido derivado.

La *No repudiación* se encarga de evaluar que las acciones o eventos que han tenido lugar no puedan ser repudiados posteriormente. Para ello, se puede hacer uso de un *historial de eventos, tanto a nivel de activos como de productos*, teniendo que comprobar la *existencia de dicho historial*, la r*estricción de acceso por personal no autentificado*, y la *encriptación de la información del historial* en el caso de tratarse de información confidencial.

La *Responsabilidad* se refiere al grado con el que las acciones de una entidad pueden atribuirse únicamente a esa entidad. En el contexto de las LPS, se ha propuesto evaluar la responsabilidad de un activo como el *Uso de funciones de un activo sin conocimiento del activo*, para evaluar si es posible que otro activo, función o usuario pueda acceder a las funciones de un activo de forma que no quede constancia del acceso externo y las acciones no puedan ser atribuidas a ese activo.

Finalmente, la *Autentificación* es medida a nivel de activos y de productos. Para la subacaracterística *Autentificación en un activo*, se han propuesto los atributos *Información personal requerida en un activo* y *Ratio de activos que requieren autentificación*. El primero se refiere a la cantidad y calidad (o complejidad) de la información requerida para autentificarse ante un activo, mientras que la segunda obtiene la cantidad de activos que requiere autenticación. Es de suponer, que cuantos más activos de la LPS requieran autentificación, una mayor parte de las funciones del producto final requerirán autentificación. La subcaracterística *Autentificación en un producto* ha sido descompuesta en dos atributos. *Información personal requerida en un producto*, tiene el mismo significado que Información personal requerida en un activo, pero esta vez para el caso de un producto concreto. *Finalización*

de sesión asegura que cuando un usuario cierra la sesión (o permanece inactivo durante un determinado tiempo) la sesión expira para todo el producto software, no permitiendo el acceso a las áreas privadas de otras funciones del producto.

5.1.2.5. Compatibilidad

La compatibilidad es la habilidad de dos o más componentes software para intercambiar información y/o llevar a cabo sus funciones mientras comparten el mismo entorno software o hardware. En el ámbito de las líneas de productos software, la compatibilidad entre los activos es un factor crucial, pues obligatoriamente deberán coexistir diferentes activos en un mismo entorno cuando éstos conformen un producto final.

El estándar ISO SQuaRE divide la compatibilidad en cuatro subcaracterísticas: Reemplazabilidad, Capacidad a coexistencia, Interoperabilidad, y Adherencia a normas. La

Tabla 5.7 muestra las subcaracterísticas y los atributos propuestos para esta característica.

La *Reemplazabilidad* ha sido dividida en el modelo propuesto en dos subcaracterísticas: la Reemplazabilidad de activos en un producto y la Reemplazabilidad de componentes en un activo. La *Reemplazabilidad de activos en un producto* se refiere a la posibilidad de reemplazar un activo dentro de un producto, sin que este cambio afecte a la calidad del producto. Los impactos podrían afectar a la funcionalidad del producto (por incompatiblidad entre activos, funciones, etc.) o afectar la variabilidad del producto (debido a que el nuevo activo contiene o elimina puntos de variación, o es dependiente de otros activos que incorporan puntos de variación al producto). Los atributos *Integración funcional de un producto* e *Integración de la variabilidad de un producto* recogen esta información. La *Reemplazabilidad de componentes en un activo* se refiere a la posibilidad de reemplazar componentes dentro de activo. En este caso, se evalúa únicamente la *Integración funcional de un activo* para conocer qué funciones permanecen invariables al reemplazo.

Tabla 5.7. Subcaracterísticas y atributos para Compatibilidad.

Subcaracterística		Atributo	Significado
6.1. Reemplazabilidad	6.1.1. Reemplazabilidad de activos en un producto	6.1.1.1. Integración funcional de un producto	Cantidad de funciones que permanecen invariables en un producto al reemplazar un activo por otro.
		6.1.1.2. Integración de la variabilidad de un producto	Cantidad de puntos de variación que permanecen invariables en un producto al reemplazar un activo por otro.
	6.1.2.	6.1.2.1. Integración	Cantidad de funciones que

Subcaracterística		Atributo	Significado
	Reemplazabilidad de componentes en un activo	funcional de un activo	permanecen invariables en un activo al reemplazar uno o más de sus componentes.
6.2. Capacidad de coexistencia	6.2.1. Capacidad de coexistencia entre activos	6.2.1.1. No impacto en otros activos	Flexibilidad de un activo para compartir su entorno con otros activos de la LP sin impactar de forma adversa en otros activos.
		6.2.1.2. No impacto en el propio activo	Flexibilidad de un activo para compartir su entorno con otros activos de la LP sin impactar de forma adversa el propio activo.
		6.2.1.3. No impacto con el mismo activo	Flexibilidad de un activo para compartir su entorno con otras instancias del mismo activo sin impactar de forma adversa en el propio activo.
	6.2.2. Capacidad a coexistencia de un producto	6.2.2.1. Capacidad de coexistencia con otros productos de LP	Flexibilidad de un producto para compartir su entorno con otros productos de la LP sin impactar de forma adversa con los otros productos.
		6.2.2.2. Capacidad a coexistencia con productos no pertenecientes a la LP	Flexibilidad de un producto para compartir su entorno con otros productos no pertenecientes a la LP sin impactar de forma adversa con los otros productos.
6.3. Interoperabilidad	6.3.1. Interoperabilidad entre activos	6.3.1.1. Interoperabilidad en el formato de datos entre activos	Interoperabilidad entre los formatos de datos utilizados.
		6.3.1.2. Interoperabilidad en los parámetros de las funciones de los activos	Interoperabilidad entre los parámetros y los tipos de datos de las funciones.
6.4. Adherencia a normas		6.4.1. Adecuación a otras normas	Grado de adecuación a otros estándares, convenciones o reglamentaciones relacionadas a la compatibilidad de LPS

La *Capacidad a coexistencia* se refiere a la capacidad de un producto en poder coexistir con otros. En el modelo se ha divido en dos subcaracterísticas: Capacidad a coexistencia entre los activos de la LP y Capacidad a coexistencia de un producto. La *Capacidad de coexistencia entre activos* evalúa los impactos que pueden existir entre los activos de una misma LP. Éstos impactos pueden darse: (1) desde el activo evaluado a los otros activos (*No impacto en otros activos*), (2) en el propio activo al coexistir con otros activos (*No impacto en el otro activo*), y (3) al coexistir varias instancias del mismo activo, ya que podría requerirse la coexistencia de varios productos de la misma línea de productos que contuvieran activos comunes (*No impacto con el mismo activo*). La *Capacidad de coexistencia de un producto* puede evaluarse teniendo en cuenta si es capaz de coexistir con otros productos de la misma línea de productos (*Capacidad de coexistencia con otros productos de la LP*) y con productos que no pertenecen a la LP (*Capacidad de coexistencia con productos no pertenecientes a la LP*). En el primer caso la evaluación va ligada al atributo No impacto con el mismo activo, en el segundo caso la evaluación es similar a un producto construido sin seguir el enfoque de las LP.

Por último, la Interoperabilidad es evaluada teniendo en cuenta la *Interoperabilidad entre los activos*, ya que este factor es crucial para poder construir un producto a partir de los activos. Los atributos a evaluar son la *Interoperabilidad en el formato de datos entre activos* (tipo de datos utilizado, como se almacenan en memoria, etc.) y la *Interoperabilidad en los parámetros de las funciones entre activos* (mismos parámetros y tipos de datos en las llamadas a funciones de un activo y las funciones pertenecientes al activo llamado).

5.1.2.6. Mantenibilidad

La Mantenibilidad es el grado con el cual el producto software puede ser modificado. Las modificaciones pueden ser correcciones, mejoras o adaptaciones a cambios en el entorno y en los requisitos y especificación de funciones. Un alto grado de mantenibilidad indica que el producto está estructurado de forma que es fácil hacer modificaciones en él. Las LPS están estructuras de forma que en la ingeniería del dominio se especifican los activos reutilizables, y en la ingeniería de la aplicación se hace uso de estos activos. Es por ello que se caracterizan por un alta reusabilidad, y esta característica es una de las más evaluadas en la literatura. En los Modelos de Calidad propuestos por el estándar ISO anteriores a la norma SQuaRE, la Reusabilidad no estaba claramente identificada en el modelo. Sin embargo, en la norma SQuaRE ha sido especificado como una subcaracterística de la Mantenibilidad. Las subcaracterísticas indicas por la SQuaRE para la mantenibilidad son: Modularidad, Reusabilidad, Capacidad de análisis, Estabilidad frente a modificaciones, Capacidad a pruebas, y Adherencia a normas.

En las LPS la *Modularidad* es una característica muy importante, pues los activos deben ser especificados como módulos que pueden ser combinados e intercambiados fácilmente (*Modularidad de la LP*). Además, esta característica también puede ser vista para construir los propios activos (*Modularidad de un activo*). En la Tabla 5.8 se muestra la descomposición de subcaracterísticas y atributos para la Modularidad.

Tabla 5.8. Subcaracterísticas y atributos para Modularidad.

Subcaracterística	Atributo	Significado
7.1. Modularidad	Modularidad de un activo	Grado de modularidad del que está compuesto una activo
	Modularidad de la LP	Grado de modularidad del que está compuesta una LP.

La *Reusabilidad* es, probablemente, la subcaracterística más relevante y evaluada en la LPS. Se ha dividido en dos subcaracterísticas más, denominadas Comunalidad y Variabilidad. Mientras que *Comunalidad* es evaluada teniendo en cuenta aspectos relacionados con la reutilización de activos debido a que son partes comunes de diferentes productos, la *Variabilidad* es evaluada atendiendo a su riqueza y complejidad. Además, han sido añadidos dos atributos (Capacidad de confección a medida y Solidez de la estructura) que no han sido catalogados en otra subcaracterística. La *Capacidad de confección a medida* [36]evalúa la facilidad de construir un activo o producto a medida a partir de unos requisitos especificados y de unos recursos disponibles. La *Solidez de la estructura* se refiere a aspectos de utilización de servicios provistos y requeridos [69]. La Tabla 5.9 muestra la descomposición de la Reusabilidad en subcaracterísticas y atributos.

Tabla 5.9. Subcaracterísticas y atributos para Reusabilidad.

Subcaracterística		Atributo	Significado
7.2. Reusabilidad	7.2.1. Comunalidad	7.2.1.1. Comunalidad funcional	Mide si las funciones provistas por un activo son comunes a las aplicaciones definidas para la línea de productos. Estas funciones son reutilizadas para desarrollar aplicaciones específicas.
		7.2.1.2. Comunalidad no funcional	Mide si los requisitos no funcionales provistos por los activos son

Subcaracterística		Atributo	Significado
			comunes a los miembros definidos en la línea de productos.
		7.2.1.3. Aplicabilidad de un activo	Mide la capacidad de un activo a ser aplicado por un número de miembros de la familia.
	7.2.2. Variabilidad	7.2.2.1. Riqueza de la variabilidad	Mide si el activo comprende ampliamente la variabilidad del alcance de la LP.
		7.2.2.2. Complejidad de la variabilidad	Evalúa la complejidad de la LP teniendo en cuenta su variabilidad.
	7.2.3. Cubrimiento de un producto	7.2.3.1. Nivel de cubrimiento de un producto con la LPS	Nivel con el cual un producto reutiliza los activos de la LPS.
		7.2.3.2. Reutilización en características propias	Nivel de reutilización de activos para especificar/implementar características propias de un producto.
		7.2.4. Capacidad de confección a medida	Capacidad de un activo para ser hecho a medida a partir de la especificación de requisitos.
		7.3.5. Solidez de la estructura	Calidad de la solidez de la estructura.

La *Capacidad de análisis* es el grado con el cual un producto software puede diagnosticar deficiencias o fallos en el software. Para medir esta subcaracterística se han propuesto los atributos *Almacenamiento de la actividad del sistema* y *Provisión de funciones de diagnóstico* para medir la rigurosidad de la capacidad de análisis tanto a nivel de activos como de producto. La *Capacidad a cambios* y la *Estabilidad frente a modificaciones* son dos subcaracterísticas directamente relacionadas. La *Capacidad a pruebas* mide el grado con el que la LPS y los activos pueden probarse.

Tabla 5.10. Subcaracterísticas y atributos de las Subcaracterísticas, Capacidad de análisis, Capacidad a cambios, Estabilidad frente a modificaciones, Capacidad a pruebas y Adherencia a normas.

Subcaracterística	Atributo	Significado
7.3. Capacidad de análisis	7.3.1. Almacenamiento de la actividad del sistema	Rigurosidad del almacenamiento del estado del sistema.

Subcaracterística	Atributo	Significado
	7.3.2. Provisión de funciones de diagnóstico	Rigurosidad de la provisión de las funciones de diagnóstico.
7.4. Capacidad a cambios	7.4.1. Almacenamiento de cambios	Adecuación del almacenamiento de los cambios en la especificación y los módulos del programa en el código con líneas comentadas.
7.5. Estabilidad frente a modificaciones	7.5.1. Frecuencia de impactos adversos	Frecuencia de impactos adversos después de modificaciones.
	7.5.2. Tamaño del impacto en la arquitectura	Tamaño del impacto de la modificación de la arquitectura.
	7.5.3. Tamaño del impacto en los activos	Tamaño del impacto de la modificación del activo.
	7.5.4. Cantidad de activos impactados	Número de activos impactados por una modificación.
	7.5.5. Propagación de cambios	Número de cambios desencadenados por la introducción de un cambio.
7.6. Capacidad a pruebas	7.6.1. Autonomía de los activos para ser probados	Independencia con la que puede ser testeado el software.
	7.6.2. Observabilidad del progreso del test	Completitud de la información mostrada al usuario mientras se realiza el test respecto a las operaciones de test que se están realizando.
	7.6.3. Completitud de los resultados del test	Completitud de la información de los resultados del test mostrados al usuario.
7.7. Adherencia a normas	7.7.1. Adecuación a otras normas	Grado de adecuación a otros estándares, convenciones o reglamentaciones relacionadas a la Mantenibilidad de LPS

5.1.2.7. Transferibilidad

La Transferibilidad se define como el grado en que un producto software puede ser transferido de un entorno a otro. En la ISO SQuaRE esta característica se divide en 3 subcaracterísticas: Portabilidad, Adaptabilidad e Instalabilidad. Como se observa en la Tabla 5.11 se ha

descompuesto la Portabilidad en dos subcaracterísticas. Por una parte se mide el *soporte a la portabilidad de los activos*. Puesto que los productos son construidos a partir de la combinación de varios activos, cuantos más activos sean portables, más portable puede ser la línea de productos y los productos derivados. Es posible evaluar el grado de portabilidad de un activo desde dos puntos de vista: (1) si los *activos ya han sido especificados* e implementados en varios lenguajes, con lo cual se podría transferir de inmediato, o (2) si *es posible conseguir otra especificación o implementación de los activos a partir de la existent*e (por ejemplo, a partir del modelado de un activo es posible realizar transformaciones de modelo a modelo y/o de modelo a código). La *facilidad para transferir* está ligada a estos dos últimos atributos, puesto que en el primer caso la portabilidad puede darse de inmediato, mientras que en el segundo caso las transformaciones pueden ser automáticas o semiautomáticas, o en el peor de los casos, no existir ninguna portabilidad.

Para evaluar la adaptabilidad, existen diferentes factores como la *adaptabilidad a la estructura de información*, al *entorno organizacional*, al *entorno software* o al *entorno hardware*, todos ellos susceptibles de ser medidos tanto a nivel de activo como de producto.

Tabla 5.11. Subcaracterísticas y atributos para Transferibilidad.

Subcaracterística		Atributo	Significado
8.1. Portabilidad	8.1.1. Soporte a la portabilidad de los activos	8.1.1.1. Activos especificados en más de un lenguaje	Cantidad de activos que han sido implementados en más de un lenguaje.
		8.1.1.2. Activos susceptibles de transformarse a otras especificaciones	Cantidad de activos que han sido definidos con modelos o implementados en código, y que pueden ser transformados en otros modelos o implementaciones que les permitan ser portables.
	8.1.2. Facilidad para portar	8.1.2.1. Portabilidad de los activos	Esfuerzo requerido para llevar a cabo operaciones de portabilidad en el activo.
		8.1.2.1. Portabilidad del producto	Esfuerzo requerido para llevar a cabo operaciones de portabilidad en el producto.
8.2. Adaptabilidad		8.2.1. Adaptabilidad a la estructura de información	Adaptabilidad de un activo o de un producto a los cambios en la estructura de información.

Subcaracterística		Atributo	Significado
		8.2.2. Adaptabilidad al entorno organizacional	Adaptabilidad de un activo o de un producto a los cambios organizacionales.
		8.2.3. Adaptabilidad al entorno hardware	Adaptabilidad de un activo o de un producto a los cambios del entorno relacionados con el hardware.
		8.2.4. Adaptabilidad al entorno software	Adaptabilidad de un activo o de un producto a los cambios del entorno relacionados con el sistema software.
8.3. Instalabilidad	8.3.1. Instalabilidad de un producto	8.3.1. Facilidad de reinstalación	Facilidad para repetir la operación de instalación.
		8.3.2. Esfuerzo para la instalación	Nivel de esfuerzo requerido para la instalación.
		8.3.3. Flexibilidad de la instalación	Capacidad de la instalación para ser flexible y personalizable.
	8.3.2. Instalabilidad de activos en un producto	8.3.2.1. Esfuerzo requerido para instalar un activo en un producto	Esfuerzo requerido para instalar uno o varios activos (añadir funcionalidad) en un producto ya instalado.
		8.3.2.1. Facilidad de reinstalación de un activo en un producto	Facilidad para repetir la operación de instalación de un activo en un producto ya instalado.
8.4. Adherencia a normas		8.4.1. Adecuación a otras normas	Grado de adecuación a otros estándares, convenciones o reglamentaciones relacionadas a la transferibilidad de LPS

La instalabilidad puede verse desde dos puntos de vista. La *instalabilidad de un producto* puede calcularse como en cualquier otro producto software. En el contexto de las líneas de productos software, es posible que se quiera complementar un producto con funcionalidades que no se necesitaban cuando el producto fue instalado, o que simplemente han sido añadidas recientemente. Así, la *instalabilidad de un activo en un producto* que ya ha sido instalado también es evaluada.

5.1.3. Especificación de relaciones

Tras definir las características, subcaracterísticas y los atributos de calidad, es posible establecer relaciones entre ellos. La importancia relativa de las características y atributos viene determinada por la forma en que impactan positiva o negativamente entre ellos. Por ejemplo, un valor alto para la reusabilidad hace que la facilidad de prueba, la flexibilidad o la portabilidad aumenten, pero la eficiencia y la integridad disminuyan. Aunque el sentido común pueda marcar unas directrices básicas, es importante que los expertos en el dominio de aplicación identifiquen los impactos y les pongan los pesos adecuados.

5.1.4. Operacionalización del modelo

Este paso consiste en la cuantificación del modelo de calidad. Para ello se asocia una o más métricas a cada atributo de calidad del modelo. Además, una misma métrica puede ser operacionalizada de forma distinta en función del artefacto que desee medir y la fase del ciclo de vida en la que se aplique.

Por ejemplo, el atributo *Tiempo de respuesta* puede ser medido con las métricas *Tiempo de respuesta de una función de un activo* o *Porcentaje de tiempo de respuesta de un activo*. Y estas mismas métricas pueden medirse en tiempo de diseño (sobre modelos) o realización (sobre código) para obtener una estimación, o en tiempo de ejecución para obtener el valor real conseguido. Así, las métricas pueden ser cuantitativas o cualitativas, e internas o externas.

En el Anexo C se presenta una relación de atributos con posibles métricas para su evaluación y en el Anexo D se muestra una propuesta de operacionalización de las métricas.

5.2. Metamodelo de Calidad Propuesto

Con el objetivo de especificar los elementos que componen un modelo de calidad y las relaciones que existen entre estos, vamos a definir un metamodelo.

Los objetivos de la definición de este metamodelo son: (1) que el metamodelo siga el estándar de calidad SQuaRE, (2) obtener un marco de trabajo y una ontología común para que los investigadores utilicen los mismos conceptos para referirse a los mismos elementos, y (3) obtener un metamodelo con el que poder definir cualquier modelo de calidad (no sólo para LPS).

Para que el metamodelo siga el estándar SquaRE debemos especificarlo de forma que los modelos de calidad puedan definirse mediante una jerarquía de características, subcaracterísticas y atributos.

A este metamodelo deberíamos incluirle otros aspectos relevantes para los modelos de calidad tales como las métricas utilizadas, su

definición, el tipo de métricas, el tipo de datos utilizados por las métricas, si existen criterios de decisión para evaluar un atributo dado el valor de una métrica, cómo representar los criterios de decisión, los impactos entre los atributos de calidad, los impactos entre requisitos y atributos de calidad, etc.

Como punto de partida para construir el metamodelo, vamos a utilizar la ontología para la medición del software propuesta por García et al. [31]. En este artículo se realiza un análisis de los modelos de calidad existentes y la terminología utilizada hasta el momento. El resultado es un modelo de calidad que unifica conceptos (y que, por lo tanto, es un modelo más completo que los anteriores) y crea una ontología común con la que los investigadores pueden utilizar los mismos términos para referirse a los mismos elementos.

Tras realizar un análisis de la ontología detectamos que no utiliza exactamente los mismos términos que el estándar SquaRE y que no incluye algunas características que hemos considerado relevantes para un modelo de calidad. Es por ello que vamos a realizar tres tipos de modificaciones con respecto a la ontología propuesta en [31]:

- Si la SQuaRE utiliza una denominación distinta que la utilizada en [31] para referirse al mismo objeto, utilizaremos la definida en la SQuaRE.
- Inclusión de elementos o relaciones necesarios para expresar el modelo de calidad que no se han tenido en cuenta en la ontología.
- Adaptaciones de la ontología a una representación utilizando un metamodelo (por ejemplo, dos objetos relacionados en la ontología con cardinalidad 1..1, en función de su significado, podrían ser representados como un objeto y un atributo).

En la Tabla 5.12 se indican las discrepancias aparecidas entre la ontología y el modelo de calidad que se pretende proponer y cómo se han tratado.

Tabla 5.12. Discrepancias y tratamiento entre la ontología SMO y nuestro metamodelo de calidad.

Elemento de la ontología SMO	Discrepancia	Tratamiento
Elemento: Measurable Concept	En la SquaRE este elemento es denominado Characteristic.	Utilizar la denominación Characteristic.
Relación: Includes (de Measurable Concept a Measurable Concept).	En la SquaRE las características se descomponen en subcaracterísticas.	Utilizar la denominación SubCharacteristic en la relación.
----	No se tienen en cuenta los impactos entre atributos de	Introducir una metaclase Impact que relacione los impactos entre

Elemento de la ontología SMO	Discrepancia	Tratamiento
	calidad	atributos de calidad.
----	No se tienen en cuenta los impactos entre los requisitos funcionales y los atributos de calidad.	Relacionar la metaclase Impact con los requisitos funcionales.
Relación: defined for (de Quality Model a Entity Class)	La cardinalidad es *..1 por lo que un modelo de calidad sólo puede evaluar un tipo de Entity Class. Deseamos especificar en único modelo que permita evaluar diferentes tipos de artefactos.	Modificamos la cardinalidad *..1 por la cardinalidad *..*
----	No se tienen en cuenta la posibilidad de que la misma métrica puede estar operacionalizada para fases y artefactos distintos.	Inclusión de la metaclase "Operacionalization" que está relacionada con "Measure" indicando que una métrica puede tener distintas operacionalizaciones, pero una operacionalización concreta corresponde a una sóla métrica. La herencia que antes había de Measure, pasa a estar en Operazionalization.

Después de realizar un estudio y diversos casos de prueba, el metamodelo para expresar modelos de calidad es el mostrado en la Figura 5.1.

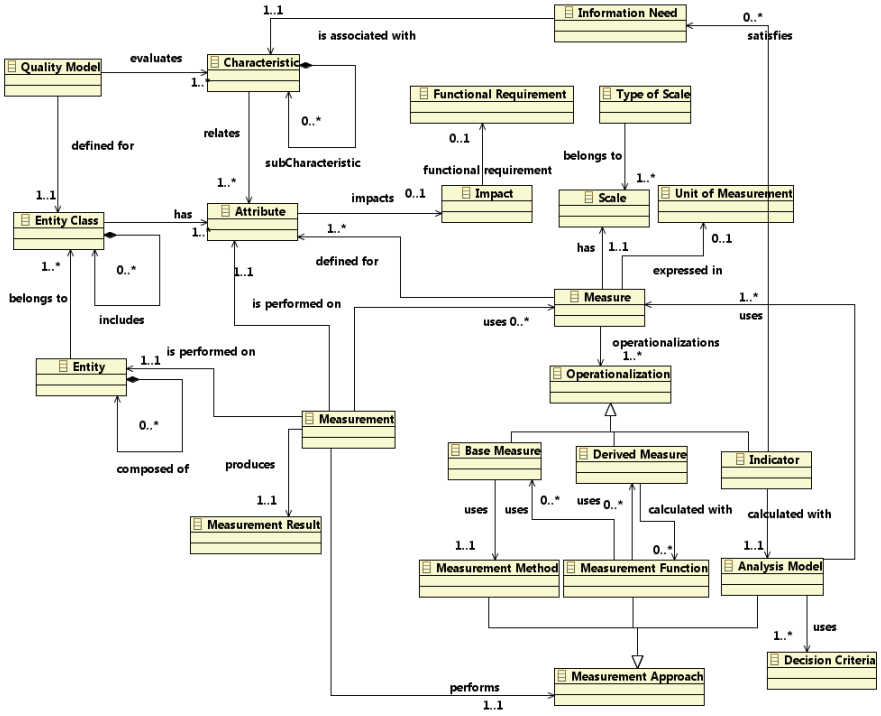

Figura 5.1. Metamodelo de calidad.

En la Tabla 5.13 se explica cada uno de los elementos del Metamodelo de Calidad. Para cada metaclase, se indican también sus atributos.

Tabla 5.13. Definición de los elementos del Metamodelo de Calidad

Metaclase	Descripción	Ejemplo
Analysis Model	La forma de medir un indicador. Algoritmo o cálculo realizado para combinar una o más medidas (base, derivada o indicadores) con criterios de decisión asociados.	Modelo de Análisis para obtener la medida PROD. Modelos de Análisis para obtener la medida CAR.
Attribute	Propiedad inherente o característica de un entidad que puede ser distinguida entre cuantitativa o cualitativa por los humanos o máquinas.	Complejidad de la LP, tamaño de código fuente, etc.
Base Measure	Una medida de un atributo que no depende de ninguna otra medida, y cuya forma de medir es un método de medición.	LCF (líneas de código fuente escritas), HPD (horas-programador diarias), CHP (coste por hora-programador, en unidades monetarias).
Characteristic	Categoría de un conjunto atributos de calidad software que soporta la calidad software. Las características	Funcionalidad, Usabilidad, etc.

Metaclase	Descripción	Ejemplo
	software puede ser refinadas en múltiples niveles de subcaracterísticas y finalmente en atributos de calidad.	
Decision criteria	Valores umbral, objetivos, o patrones, usados para determinar la necesidad de una acción o investigación posterior, o para describir el nivel de confianza de un resultado dado.	El nivel de confianza de la variabilidad de una LPS es 0.5 en una escala del 0 al 1.
Derived Measure	Una medida que es derivada de otra medida base o derivada, utilizando una función de cálculo como forma de medir.	HPT (horas-programador totales, que es la sumatoria de las HPD de cada día), LCFH (líneas de código fuente por hora de programador), CTP (coste total actual del proyecto, en unidades monetarias, que es el producto del coste unitario de cada hora por el total de horas empleadas).
Entity	Un objeto que va a ser caracterizado mediante una medición de sus atributos	La arquitecta de la LP, la arquitectura de un producto único, un activo.
Entity Class	Una colección de entidades caracterizadas por satisfacer un cierto predicado común.	Arquitecturas de la LP, Activos expresados mediante modelos, Activos expresados con código, Productos finales.
Impact	Factor que indica cómo se modifica un atributo de calidad o requisito funcional al introducir, eliminar o modificar otro atributo de calidad o requisito funcional.	La Reusabilidad impacta positivamente en la Mantenibilidad.
Indicator	Una medida que es derivada de otras medidas utilizando un modelo de análisis como forma de medir.	PROD (productividad de los programadores), CAR (carestía del proyecto).
Information Need	Información necesaria para gestionar un proyecto (sus objetivos, hitos, riesgos y problemas)	Conocer las ventajas de incorporar una característica nueva a la línea de productos en comparación a ofrecer la característica como propia de un único producto.
Measure	Descripción de cómo se obtiene el valor de un atributo.	La medida del atributo "Complejidad de la variabilidad" se obtiene con la métrica "Número de variabilidades en la LPS".
Measurement	Conjunto de operaciones que permite obtener el valor del resultado de la medición para un	Acción consistente en usar la forma de medir "contar el número de líneas de código"

Metaclase	Descripción	Ejemplo
	atributo de una entidad, usando una forma de medir.	para obtener el resultado de la medición del atributo "tamaño" de la entidad "módulo nóminas.c".
Measurement Approach	Secuencia de operaciones cuyo objeto es determinar el valor del resultado de la medición. Una forma de medir puede ser un método de medición, función de cálculo o modelo de análisis.	Proceso para la evaluación de un producto propuesto en el estándar ISO 14598.
Measurement Function	La forma de medir una medida derivada. Algoritmo o cálculo realizado para combinar dos o más medidas base y/o derivadas.	LCFH= LCF/HPT, CTP=CHP*HPT
Measurement Method	La forma de medir una medida base. Secuencia lógica de operaciones, descritas de forma genérica, usadas para realizar mediciones de un atributo respecto de una escala específica.	Contar líneas de código, anotar cada día las horas dedicadas por los programadores al proyecto.
Measurement Result	Categoría o número asignador a un atributo de una entidad como resultado de una medición.	35.000 líneas de código, 200 páginas, 50 clases, 5 meses desde el comienzo al fin del proyecto, 0.5 fallos por cada 1000 líneas de código.
Operability	Operacionalización de una métrica	
Quality Model	Conjunto de características, y las relaciones entre ellas, las cuales provienen de un marco de trabajo para especificar requisitos de calidad y evaluar la calidad.	Modelo de calidad para líneas de productos software, Modelo de calidad para evaluar la usabilidad web, etc.
Scale	Un conjunto de valores con propiedades definidas.	El nivel de madurez CMM: 1, 2, 3, 4 (Ordinal), el tamaño de un código software expresado en líneas de código: Conjunto de los números naturales (Ratio), etc.
Type of Scale	Indica la naturaleza de la relación entre los valores de la escala.	Nominal, Ordinal, Intervalo, Ratio y Absoluta.
Unit of Measurement	Una cantidad particular, definida y adoptada por convención, con la que se puede comparar otras cantidades de la misma clase para expresar sus magnitudes respecto a esa cantidad particular.	Kilómetros, metros, millas, líneas de código, páginas, persona-mes, número de módulos, número de clases.

Capítulo 6

Método para evaluar la Calidad en LPS

El modelo de calidad para LPS pretende servir de marco para evaluar cualquier LPS en cualquier de sus fases. Sin embargo, este es sólo un artefacto de evaluación. Para asegurar la calidad del software y explicar cómo hacer uso del modelo de calidad propuesto, en este capítulo se propone un Método de Evaluación de Calidad para LPS basado en SQuaRE.

6.1. Proceso genérico de evaluación de la calidad

El estándar ISO/IEC SQuaRE, junto con el estándar ISO/IEC 14598 definen los pasos que deben seguirse en el proceso de evaluación. Estos son: Establecimiento de los requisitos de evaluación, Especificación de la evaluación, Diseño de la evaluación y Ejecución de la evaluación. Además, cada uno de estos pasos es divido en varias actividades.

Con el objetivo de que el proceso para la evaluación de la calidad de líneas de productos software propuesto sea conforme al estándar ISO/IEC SQuaRE, van a seguirse estos pasos de evaluación y a adaptarse al caso concreto del desarrollo de líneas de productos software. Se pretende proponer un método de evaluación que pueda ser utilizado para cualquier método de desarrollo de líneas de productos. No obstante, la intención final es utilizarlo en un desarrollo mediante múltiples modelos.

A continuación se describen las acciones que deben tomarse en cada paso del proceso de evaluación de la calidad:

1. Establecimiento de los requisitos de evaluación.

1.1. Establecimiento del propósito de la evaluación.

El propósito de la evaluación del producto software es, en general, comparar la calidad de un producto software contra los requisitos de calidad que expresan las necesidades de los usuarios, o incluso seleccionar un producto software y compararlo con diferentes productos, o establecer un ranking de productos con respecto a sus competidores. Este objetivo general puede ser mejor especificado cuando se considera el punto de vista de la evaluación del producto software, tal como el de la adquisición, durante el desarrollo, o en ejecución.

1.2. Identificación de los tipos de productos a ser evaluados.

Los tipos de productos a ser evaluados dependen del propósito de la evaluación. Como primer paso, el evaluador debería definir los productos a ser evaluados como intermedios (durante el desarrollo del ciclo de vida) o como productos finales. Los productos a ser evaluados pueden ser medidos usando: (1) métricas externas, cuando el producto es parte del sistema software/hardware completo bajo ejecución; (2) métricas intrínsecas que pueden ser aplicadas para medir propiedades internas del software (por ejemplo la especificación o el código fuente); (3) y la calidad en uso, que mide los efectos del uso del software en un entorno específico.

1.3. Especificación del modelo de calidad.

El modelo de calidad especificado para la evaluación es la referencia para la definición de los requisitos del producto software. En este paso de evaluación los requisitos son descritos

para las características relevantes de calidad, siendo priorizadas de acuerdo a las necesidades de los usuarios.

2. Especificación de la evaluación.

2.1. Selección de las métricas.

La especificación cuantitativa y la medición de los requisitos de calidad del producto software pueden ser únicamente hechas utilizando métricas, las cuales están asociadas a características de calidad deseadas. Las métricas pueden ser: (1) internas, asociadas a la arquitectura del producto software y permitiendo predecir la calidad del producto final; (2) externas, medibles cuando el producto está en ejecución; y (3) de calidad en uso, que evalúan los efectos del uso del software.

La elección de las métricas utilizadas durante la evaluación del producto software depende del propósito de la evaluación, de la selección de las características de calidad y de cómo pueden ser aplicadas de manera fácil y económica. Esto significa que las métricas deben ser objetivas, reproducibles y validadas empíricamente, lo que indica que estamos utilizando una escala de medición válida.

2.2. Establecimiento de los niveles de ratio de las métricas.

Para cada métrica seleccionada debe definirse una escala de valores, donde se indique el nivel requerido para el atributo medido. La adopción de la escala puede indicar los límites para cada atributo, identificando si el valor de la medida es, por ejemplo, aceptable, mínimamente aceptable dentro del rango esperado, o excede los requisitos.

2.3. Establecimiento de los criterios para el aseguramiento.

El evaluador debe preparar un procedimiento que separe los criterios para diferentes características de calidad, cada una de las cuales puede estar en términos de subcaracterísticas individuales, o una combinación de pesos de subcaracterísticas. Esto incluye otros aspectos tales como el tiempo y el coste en el que se incurre para el aseguramiento de la calidad de un producto software en un entorno particular.

3. Diseño de la evaluación.

3.1. Producción del plan de evaluación.

Se debe describir el plan de evaluación, los métodos de evaluación empleados y las tareas del evaluador.

4. Ejecución de la evaluación.

4.1. Ejecución y anotación de las medidas.

Esta tarea consiste en la ejecución de la evaluación para productos intermedios y/o para el producto final. Selecciona las métricas a aplicar al producto software.

4.2. Comparación con los criterios.

Los valores medidos son comparados con los criterios establecidos en la especificación. Se consideran tanto los valores medidos para la evaluación interna como los valores medidos sobre el producto final. Las mediciones internas son consideradas indicadores para predecir la calidad del producto final.

4.3. Aseguramiento de los resultados.

El objetivo de esta tarea es resumir los resultados obtenidos y compararlos con otros aspectos, tales como el tiempo y el coste. Finalmente, se decide si los resultados son aceptables o no. Esta fase tiene una influencia sobre el siguiente paso del ciclo del desarrollo de software ya que, por ejemplo, puede plantear modificaciones sobre los requisitos, los recursos utilizados, etc.

6.2. Desarrollo de LPS utilizando multimodelos

El objetivo es incorporar la evaluación de calidad en el desarrollo de líneas de productos software siguiendo el enfoque dirigido por modelos. En concreto, se pretende describir el software utilizando varios modelos o vistas del sistema (modelo de funcionalidad, de características, de calidad, etc.) con relaciones entre ellas. De este modo se produce una parametrización del proceso de producción mediante un Multimodelo que es capaz de captar las diferentes vistas del producto y las relaciones entre ellas. El Modelo de Calidad propuesto en el Capítulo 5 representará una vista del multimodelo y será un artefacto "activo" que conducirá el plan de producción de la línea de productos.

La gestión de la variabilidad, como la base para el desarrollo de las LPS, implica por un lado el manejo de las características del dominio plasmadas en un **modelo de características**, y por otro lado el que dicha variabilidad debe ser soportada en activos software. La especificación de la variabilidad y la funcionalidad pueden ser manejadas en modelos independientes y separados. Además, para asegurar la calidad del producto, ésta también debería ser incorporada en el proceso del desarrollo de software, a través de un **modelo de calidad**. De este modo, se consideran (al menos) tres vistas de los sistemas software: (1) la Vista de la Variabilidad del Sistema (VVS); (2) la Vista Funcional del Sistema (VFS); y (3) la Vista de la Calidad del Sistema (VCS).

Este enfoque permite plantear los problemas de consistencia intra-modelo (por ejemplo, la consistencia del Modelo de Características) e inter-modelos (por ejemplo, la relación entre el Modelo de Características y el Modelo de Calidad) en un contexto más amplio y realista.

Aunque este trabajo está centrado en la Vista de la Calidad del Sistema, es relevante para su entendimiento conocer el contexto en el que se pretende evaluar la calidad. En general, los procesos de evaluación de la calidad son ejecutados de forma independiente al proceso de desarrollo del software. Sin embargo, siguiendo el enfoque del multimodelado, la VCS participa también en el desarrollo del software, siendo un artefacto activo durante todo el ciclo de vida. La VCS interactúa con el resto de vistas y sus relaciones conducen el plan de producción de un producto final (ver Figura 6.1).

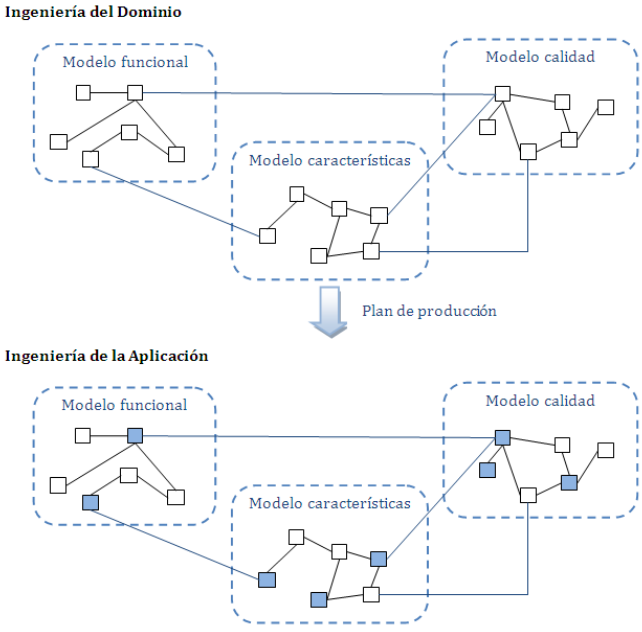

Figura 6.1. Plan de producción con múltiples modelos.

De esta manera, la existencia de varios modelos del sistema lleva a constituir el multimodelo que captura las diferentes vistas del mismo y sus relaciones y permite abordar formalmente en él, sus diferentes propiedades (por ejemplo, la consistencia a nivel de multimodelo). Este mutimodelo permite coadyuvar en los procesos de producción del software que satisfagan simultáneamente los requisitos de calidad, funcionalidad, complejidad, mantenimiento e incluso a la incorporación de nuevos productos en la familia a medida que aparezca la necesidad de producirlos.

6.3. Proceso de evaluación de la calidad para LPS

En el proceso de evaluación propuesto se recogen dos aspectos que se han considerado básicos: adaptar la evaluación al ciclo de vida de desarrollo de las LPS que ha sido explicado en el Capítulo 2 y especificar el

proceso de evaluación de la calidad de manera que sea conforme al estándar SQuaRE.

El método trata de combinar el ciclo de vida de las líneas de productos presentado en el Capítulo 2 con el proceso de evaluación presentado en la Sección 6.1 (ver Figura 6.2). De este modo, mientras se va siguiendo el ciclo de vida del desarrollo de software, se pueden ir siguiendo paralelamente una, dos, tres o las cuatro actividades del proceso de evaluación, en función de las necesidades del desarrollador de software y del evaluador. En cada una de las fases del ciclo de vida de las LPS es necesario, si el evaluador lo cree conveniente, establecer los requisitos de evaluación, especificar la evaluación, producir el plan de evaluación y ejecutar y anotar las medidas. Pero el momento para establecer los 3 primeros pasos es decisión del evaluador. Por ejemplo, una opción podría ser establecer al principio del desarrollo de la LPS todos los requisitos de evaluación para todo el ciclo de vida de desarrollo, establecer la evaluación y producir el plan de evaluación para todas las fases del desarrollo. De este modo, en cada fase del desarrollo de la LPS sólo quedaría ejecutar la evaluación. Otra posible opción podría ser establecer en la fase de requisitos de la ingeniería del dominio sólo los requisitos, la evaluación, el plan de evaluación y la ejecución de la evaluación sólo para esta fase. Así, el proceso de evaluación queda abierto y en la siguiente fase (diseño en la ingeniería del dominio) pueden realizarse las mismas actividades (establecer requisitos, evaluación, plan de evaluación y ejecución) pero para los artefactos concretos de la fase de diseño.

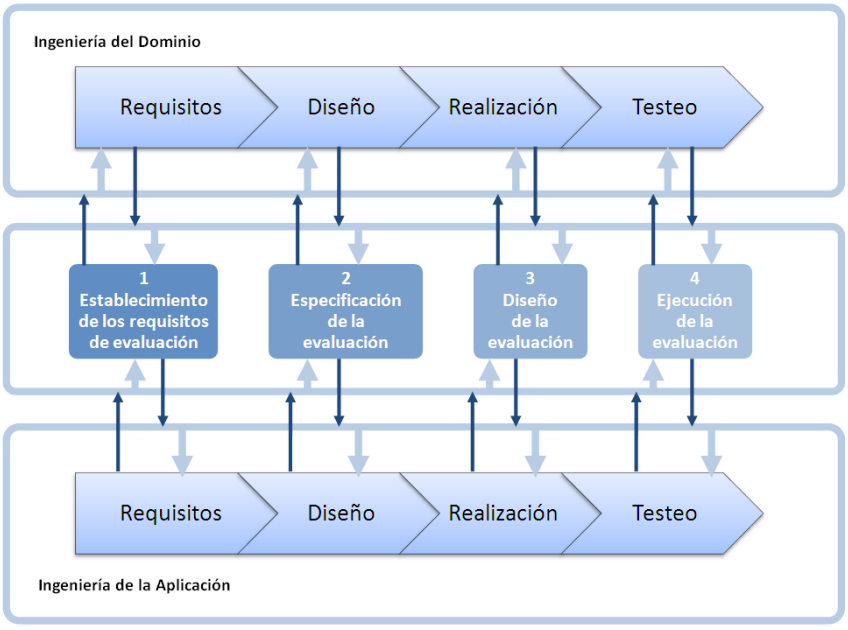

Figura 6.2. Método de evaluación de la calidad.

Resumiendo, la propuesta trata de no ser restrictiva en cuanto a establecer los requisitos de evaluación, seleccionar las métricas, elaborar el plan de evaluación y ejecutar la evaluación, de manera que este proceso pueda realizarse de forma incremental en la fase del desarrollo que el evaluador crea necesaria. En cada fase se puede utilizar una instancia del Modelo de Calidad propuesto en el Capítulo 5 (con un conjunto específico de atributos) y enriquecerlo con más atributos de calidad y métricas a medida que se produjeran nuevos artefactos en el desarrollo de la LPS que podrían no haber sido tenidos en cuenta en fases anteriores. Además, la ejecución de la evaluación ofrece retroalimentación tanto a la fase de desarrollo en la que se aplica como a las previas, pudiendo hacer modificaciones en base a los resultados obtenidos.

En cada fase del ciclo de vida de la LP, el proceso de evaluación de la calidad siempre sigue las mismas actividades (a excepción de que el evaluador ya las haya realizado previamente o considere oportuno desestimar la evaluación en una fase en concreto) pero adaptadas a las condiciones de cada fase (por ejemplo, al tipo de artefacto). Es por eso que explicar de nuevo todas las fases sería redundar en los mismos conceptos. No obstante, a continuación, se describen las ideas fundamentales del proceso de evaluación para las tres fases principales del desarrollo de LPS: la Ingeniería del Dominio, la Ingeniería de la Aplicación y la Evolución.

6.3.1. Evaluación en la Ingeniería del Dominio

En la Ingeniería del Dominio se establecen los productos que se van a desarrollar en la LPS, las partes comunes que tienen, la variabilidad, el plan de producción que se va a seguir, etc. El objetivo de la evaluación de calidad en esta fase es asegurar que los artefactos software desarrollados (activos) cumplen con ciertas características y atributos de calidad relevantes para el dominio. Se pueden realizar las cuatro actividades en cada una de las fases. Las ideas fundamentes de cada actividad se comentan a continuación.

1. Establecer los requisitos de evaluación

1.1. Establecer el propósito de la evaluación

En esta actividad se debe especificar cuál es el propósito de la evaluación. El método permite realizar la evaluación de la calidad en todas las fases del ciclo de desarrollo de las LPS. Sin embargo, en función del propósito, complejidad, costes, etc. el diseñador puede decidir si desea implantar todas las posibles evaluaciones o únicamente aquellas que le resulten de mayor interés.

1.2. Identificar los tipos de productos a ser evaluados

La elección de los tipos de productos a ser evaluados va directamente ligada con el propósito de la evaluación. El método da soporte a la evaluación de todos los artefactos involucrados susceptibles de ser medidos, como por ejemplo la línea de

productos completa, los activos de forma individual, la variabilidad de la línea de productos (relaciones, comunicaciones, impactos entre los activos, etc.), la arquitectura del producto final, etc.

No obstante, el diseñador debe especificar qué artefactos incorporar o excluir al proceso de evaluación.

1.3. Especificar el modelo de calidad

En el modelo de calidad se plasman todos los requisitos no funcionales relevantes, es decir, todos aquellos atributos de calidad que son susceptibles de ser medidos. En este paso se hace uso de una instancia del modelo de calidad propuesto en el Capítulo 5, el cual ha sido diseñado siguiendo el estándar SQuaRE y adaptado específicamente para las LPS. No obstante, es posible modificar el modelo para añadir o eliminar atributos de calidad en función de las necesidades del diseñador.

El modelo de calidad también es capaz de capturar los impactos existentes entre los atributos de calidad, y entre los atributos de calidad y los requisitos funcionales. En esta actividad también se incluye la identificación de estos impactos en la LPS.

2. Especificar la evaluación

2.1. Seleccionar las métricas

Cada atributo de calidad del modelo de calidad de la actividad 1.3. puede ser medido con una o más métricas. Puede hacerse uso de las métricas propuestas y/o combinarlas con otras métricas que el diseñador considere relevantes. En este punto hay que prestar atención tanto al atributo de calidad que se desea medir como a la fase del ciclo de vida en la que se desea aplicar, ya que en función del artefacto y de la fase del ciclo de vida, la operacionalización de la métrica es distinta.

2.2. Establecer los niveles de ratio de las métricas

Una vez seleccionadas las métricas, se deben establecer unos indicadores para conocer cuándo el valor obtenido es insuficiente, normal, suficiente, etc. Los valores pueden ser absolutos, porcentuales o discretos. Aunque se han propuesto unos indicadores por defecto, queda en la decisión del diseñador adaptar esos valores al dominio del problema.

2.3. Establecer los criterios para el aseguramiento

Se debe preparar un procedimiento para establecer los criterios con los que decidir finalmente el nivel de calidad para un atributo, subcaracterística o característica. Los criterios pueden ser una combinación de pesos de las métricas que lo miden, e incluir información adicional como el tiempo o el coste. El nivel de detalle de los criterios es decisión del diseñador. Por ejemplo, puede considerarse como criterio cada uno de los atributos. En

este caso, el criterio puede venir dado por una combinación de los resultados de las métricas que miden cada atributo. En el caso de una característica de calidad, el criterio puede venir dado por todas las métricas que miden cada una de sus subcaracterísticas, o si se ha obtenido el valor para cada una de las subcaracterísticas, usar estos valores para obtener el resultado final.

3. Diseñar el plan de evaluación

Una vez determinado qué se quiere evaluar y con qué se va a evaluar, es necesario indicar cómo se debe realizar la evaluación. En esta actividad el evaluador debe indicar cómo se debe utilizar el modelo de calidad, las métricas, etc.

Las métricas pueden ser evaluadas de forma manual o automática, si se dispone de una herramienta que así pueda hacerlo. Se recomienda seguir los siguientes pasos para la evaluación:

1. Identificar los atributos a medir.
2. Identificar los artefactos que se van a evaluar.
3. Identificar las métricas a medir.
4. Identificar los tipos de métricas y las dependencias entre ellas de manera que:
   - Se calculen primero las métricas base y después las derivadas que dependan de métricas ya calculadas.
   - Se calculen primero las internas y después las externas para cada atributo (siguiendo el ciclo de vida propuesto en la norma ISO SQuaRE).
5. Ejecutar la evaluación.

4. Ejecución de la evaluación

A partir de la información de las actividades previas se procede a la evaluación de la calidad. El modelo de calidad y las métricas operacionalizadas son utilizadas como se indica en el plan de evaluación para obtener los resultados. Las actividades para ejecutar la evaluación son:

4.1. Ejecución y anotación de las medidas

Se seleccionan las métricas destinadas a evaluar la fase en concreto (requisitos, diseño, realización o pruebas) de la ingeniería del dominio siguiendo el plan de evaluación establecido en la actividad *3. Diseñar el plan de evaluación* y se realizan las mediciones.

4.2. Comparación con los criterios

Se comparan los valores medidos con los criterios establecidos en la actividad *2.3. Establecer los criterios para el aseguramiento.*

4.3. Aseguramiento de los resultados

Se resume y analiza los resultados, comparándolos con otros aspectos como el tiempo y el coste, para decidir si es necesario o conveniente introducir cambios o mejoras.

6.3.2. Evaluación en la Ingeniería de la Aplicación

En la Ingeniería de la Aplicación se construye un producto final en base a unos requisitos indicados y mediante la reutilización de los activos indicados en la Ingeniería del Dominio. Por lo tanto, el objetivo de la evaluación en esta fase es asegurar que los productos desarrollados a partir de los activos reutilizables cumplen con los requisitos y expectativas de los clientes. Asimismo, la evaluación de la calidad durante la ingeniería de la aplicación puede proporcionar un feedback a la fase de ingeniería del dominio que permitirá mejorar la calidad de los activos de la línea de productos.

Pueden aparecer dos tipos de requisitos en función de su procedencia, cada uno de los cuales puede ser tanto requisitos funcionales como no funcionales. Éstos son:

- *Requisitos existentes en la línea de productos*: estos requisitos son un subconjunto de los requisitos especificados en la fase del dominio (tanto los funcionales como los de calidad).
- *Requisitos propios del producto concreto*: estos requisitos son introducidos para utilizarse en un producto concreto, con lo que hay que prestar especial atención para asegurar si pueden o no ser introducidos, si impactan o reciben impactos de otros requisitos, etc. Estos requisitos pueden ser tanto funcionales como de calidad.

Las actividades de evaluación que pueden realizarse en cada fase del ciclo de desarrollo para la Ingeniería del Dominio se comentan a continuación.

1. Establecer los requisitos de evaluación.

Los requisitos de evaluación ya han sido establecidos en la fase de la ingeniería del dominio. En esta fase se parte de los requisitos establecidos en la ingeniería del dominio y se selecciona un subconjunto de requisitos con los que evaluar el producto final. Además, puede ser necesario introducir nuevos requisitos que no forman parte de la línea de productos, y que son considerados como específicos del producto final. Con lo cual, se debe revisar que los nuevos requisitos son coherentes con los existentes y que no hay conflictos entre todos los requisitos del producto final.

1.1. Establecer el propósito de la evaluación.

Se debe establecer cuál es el propósito de la evaluación. Aunque el método permite evaluar el producto en todas sus fases de desarrollo en la ingeniería de la aplicación, es decisión del diseñador establecer cuál es el propósito y qué es lo que desea evaluar.

Uno de los objetivos de la evaluación en la ingeniería de la aplicación es corroborar que la selección de requisitos, el diseño de la arquitectura del producto final, los activos, etc. son lo suficientemente completos como para especificar e implementar el producto final deseado con los niveles de calidad exigidos y, si se han incluido nuevos requisitos, comprobar que son coherentes y no hay impactos perjudiciales con el resto de artefactos del producto final heredados de la LPS.

1.2. Identificar los tipos de productos a ser evaluados

De nuevo, es posible evaluar diferentes artefactos en el desarrollo del producto, pero la elección final queda en la decisión del diseñador.

1.3. Especificar el modelo de calidad

A partir de los requisitos no funcionales se selecciona el subconjunto del modelo de calidad especificado en la ingeniería del dominio que se va a utilizar para evaluar el producto final en la fase de la ingeniería de la aplicación. Si es necesario, se pueden incluir nuevas subcaracterísticas y atributos al modelo.

2. Especificar la evaluación

2.2. Seleccionar las métricas

A partir del modelo de calidad de la actividad *1.3. Especificar el modelo de calidad* anterior y de las métricas propuestas en la ingeniería del dominio, se seleccionan las métricas a aplicar para evaluar los atributos de calidad deseados con métricas operacionalizadas para la fase de la ingeniería de la aplicación (y de la(s) fase(s) en concreto que se desee evaluar). También es posible introducir nuevas métricas, en especial aquellas que puedan ser relevantes para evaluar requisitos específicos del producto.

2.2. Establecer los niveles de ratio de las métricas

Se indican los niveles con los que evaluar los resultados de las mediciones. Aunque las métricas reutilizadas de la fase de la ingeniería del dominio ya tienen establecidos unos valores, es posible modificarlos en función de las características del producto.

2.3. Establecer los criterios para el aseguramiento de calidad

Los criterios para decidir si, globalmente, una característica, subcaracterística o atributo tiene la calidad suficiente, son establecidos en este paso. Estos valores pueden heredarse de los

establecidos en la ingeniería de la aplicación, modificarse para el producto en concreto o incluir nuevos valores en el caso de nuevos atributos de calidad o subcaracterísticas.

Es habitual que exista impactos entre los requisitos funcionales y los atributos de calidad, y entre los propios atributos de calidad. Por ejemplo, permitir que un producto sea portable (es decir, aumentar la portabilidad) puede aumentar el tamaño de código y con ello la complejidad y la mantenibilidad. En la fase de la ingeniería del dominio se han establecido los impactos que pueden existir entre los elementos de la especificación, pero es en la fase de la ingeniería de la aplicación donde se debe decidir qué requisitos y atributos de calidad escoger, y con qué valor para cada uno de ellos. Sin embargo, no siempre se pueden conocer y/o conseguir los valores exactos que se desean para los atributos de calidad. Es por ello que parece interesante dejar ciertos atributos de calidad sin un valor específico, indicando que se desean maximizar ciertos atributos, que otros pueden tener un nivel intermedio, o incluso que algunos no son necesarios. De esta forma, es posible realizar un análisis trade-off entre los atributos de calidad y entre los atributos de calidad y los requisitos funcionales.

3. Diseñar el plan de evaluación

El plan de evaluación determina los pasos a seguir y cómo utilizar el modelo de calidad definido en la actividad *1.3. Especificar el modelo de calidad* y las métricas de la actividad *2. Especificar la evaluación*. Las métricas pueden ser evaluadas de forma manual o automática, en función de la especificación utilizada para los artefactos evaluados y las herramientas disponibles. Se recomienda seguir los siguientes pasos para la evaluación:

1. Identificar los atributos a medir.
2. Identificar los artefactos que se van a evaluar.
3. Identificar las métricas a medir.
4. Identificar los tipos de métricas y las dependencias entre ellas de manera que:
    - Se calculen primero las métricas base y después las derivadas que dependan de métricas ya calculadas.
    - Se calculen primero las internas y después las externas para cada atributo (siguiendo el ciclo de vida propuesto en la norma ISO SQuaRE).
5. Ejecutar la evaluación.

4. Ejecución de la evaluación

4.1. Ejecución y anotación de las medidas

Se seleccionan las métricas destinadas a evaluar los artefactos de la fase del ciclo de vida de la ingeniería de la aplicación que se desea evaluar y se realizan las mediciones.

4.2. Comparación con los criterios

Se comparan los valores medidos con los criterios establecidos en la actividad *2.3. Establecer los criterios para el aseguramiento de la calidad.*

4.3. Aseguramiento de los resultados

En base a los resultados obtenidos, se decide si se ha alcanzado un nivel sufiente de calidad o si deben introducirse cambios. Se ofrece retroalimentación a las fases de desarrollo anteriores para introducir las posibles mejoras.

6.3.3. Evaluación en la Evolución de la LPS

La introducción de nuevos requisitos da lugar a la Evolución de la LPS. Pueden existir dos tipos de Evolución: la Evolución de la LPS y la Evolución de un producto final.

La **evolución de la LPS** es debida a la necesidad de evaluar nuevos requisitos introducidos a la LPS, si estos impactan en los requisitos ya existentes, como especificar esos nuevos requisitos (introducirlos como una característica más, incluirlos con un nuevo activo, modificar activos existentes, etc.). La evolución de la LPS también puede venir por la aparición de características propias de un producto final, que, durante o posterior a su desarrollo, se ha desea incorporar a la LPS.

La **evolución de un producto final** es debida a la necesidad de evaluar nuevos requisitos de un producto ya terminado. Los nuevos requisitos pueden provenir de requisitos especificados en la LPS o ser requisitos no tenidos en cuenta anteriormente.

En la evaluación de la evolución se siguen las cuatro actividades de evaluación especificadas en las fases anteriores: Establecer los requisitos de evaluación, Especificar la evaluación, Diseñar la evaluación y Ejecutar la evaluación. Dependiendo del tipo de evolución (de la LPS o del producto final) se seguirán las actividades más relacionadas con la Ingeniería del Dominio o con la Ingeniería de la Aplicación.

Capítulo 7

Aplicación del método

En el presente capítulo se muestra un ejemplo de desarrollo de una línea de productos software para sistemas de comercio electrónico. El método para la evaluación de la calidad propuesto y el modelo de calidad han sido aplicados sobre esta línea de productos para evaluar la característica de mantenibilidad.

7.1. Introducción

En este capítulo se muestra un ejemplo de aplicación del Modelo de Calidad y del Método propuestos. La evaluación se realiza sobre una línea de productos para sistemas dedicados al comercio electrónico. El caso de estudio se ha basado en el descrito en [Hassan Gomaa] para modelar líneas de productos utilizando UML.

La línea de productos de Comercio Electrónico es un World Wide Web altamente distribuido que mantiene sistemas de "negocio a negocio" (business-to-business: B2B) y de "negocio a cliente" (business-to-consumer: B2C). Para llevar a cabo sus tareas, se utilizan agentes como intermediarios entre las interfaces de usuario de clientes y servicios. Además, hay objetos brokers que provienen de una interfaz estandarizada para distintas bases de datos heterogéneas.

7.2. Descripción del problema

Se desea crear una línea de productos para sistemas dedicados al comercio electrónico. Existen dos tipos de sistemas básicos que se deben poder crear: los sistemas B2B y los sistemas B2C.

En los sistemas B2B hay clientes (customers) y proveedores (suppliers). Cada cliente tiene un contrato con un proveedor para adquirir artículos de ese proveedor, además de una o más cuentas bancarias con las que pagar a los proveedores. Cada proveedor tiene un catálogo de artículos, acepta pedidos de los clientes, y tiene cuentas con cada cliente para recibir los pagos.

Un cliente puede hacer búsquedas a través del catálogo que está en la web de un proveedor y seleccionar los artículos para comprarlos. Los pedidos de los clientes se revisan junto con los contratos disponibles para determinar si hay un contrato válido entre el cliente y el proveedor, el cual será utilizado para el cargo del pedido. Cada contrato tiene fondos asignados a él. Es necesario determinar que hay fondos suficientes para el pedido del cliente. Asumiendo que el contrato y los fondos están disponibles, una orden de entrega es creada y enviada al catálogo del proveedor. El proveedor confirma que la orden ha llegado y planifica la fecha de envío. Un tiempo después, la orden de envío es seguida y el proveedor y el cliente son notificados si hay una orden de entrega. Cuando la orden es enviada, el cliente es notificado. El cliente reconoce que la recepción se ha efectuado y la orden de entrega es actualizada. Después de la recepción del envio, el pago de factura es autorizado. La factura se coteja con el contrato, los fondos disponibles, y el estado de la orden de entrega, y después se envía a las Cuentas sin pagar, las cuales autorizan el pago de los fondos. El pago se realiza a través de una transferencia electrónica de los fondos desde el banco del cliente al banco del proveedor. La aplicación

utiliza distintas bases de datos legadas, así que es necesario hacer uso de la tecnología de los objetos brokering y wrapper.

Opcionalmente, un proveedor puede crear una petición de orden de compra para solicitar suministros al mayorista. La orden de compra se envía directamente al mayorista. Cuando la orden de compra se entrega al proveedor, los nuevos artículos se incluyen en el inventario del proveedor, y el pago es realizado con una transferencia desde el banco del proveedor al banco del mayorista.

En los sistemas de comercio electrónico B2C un cliente solicita comprar uno o más artículos del proveedor. El cliente proporciona los datos personales, como la dirección y la información de la tarjeta de crédito. Si la tarjeta de crédito es válida, se crea una orden de entrega y se envía al proveedor. El proveedor confirma el pedido e indica una fecha de envío prevista. Cuando la orden es enviada, el cliente es notificado y se realiza el cargo en la tarjeta de crédito del cliente.

7.3. Desarrollo y evaluación de LPS Comercio Electrónico

En las siguientes subsecciones se muestra el desarrollo y evaluación de la LPS Comercio Electrónico. Debido al tamaño que supondría evaluar toda una línea productos, se ha dado prioridad a la evaluación de artefactos generados en las fases tempranas del desarrollo del software. Así, se muestran las fases de Requisitos y Diseño de la Ingeniería del Dominio para mostrar la evaluación de toda la línea de productos, y las fases de Requisitos y Diseño de la Ingeniería de la Aplicación para mostrar la evaluación de un producto concreto.

7.3.1. Evaluación en la Ingeniería del Dominio

En la Ingeniería de Dominio se analiza el problema, se establecen los requisitos y se diseña la LPS. La aportación del método introduce las fases de evaluación de la calidad. A continuación se evalúan las fases de Análisis y Requisitos, y de Diseño.

7.3.1.1. Análisis y Requisitos en la Ingeniería del Dominio

El **modelo de casos** de uso que describe el sistema es el mostrado en la Figura 7.1. donde se puede observar que se ha especificado el sistema con 9 casos de uso, en los cuales intervienen 5 actores. Los actores son:

- Customer: el cliente que desea hacer una compra.
- Supplier: el proveedor que realiza ventas a los clientes, y compras al mayorista.
- Wholesaler: el mayorista que suministra artículos al proveedor.
- Authorization center: el centro de autorizaciones que comprueba la existencia de contratos.

- Bank: el banco para realizar los pagos y las transferencias.

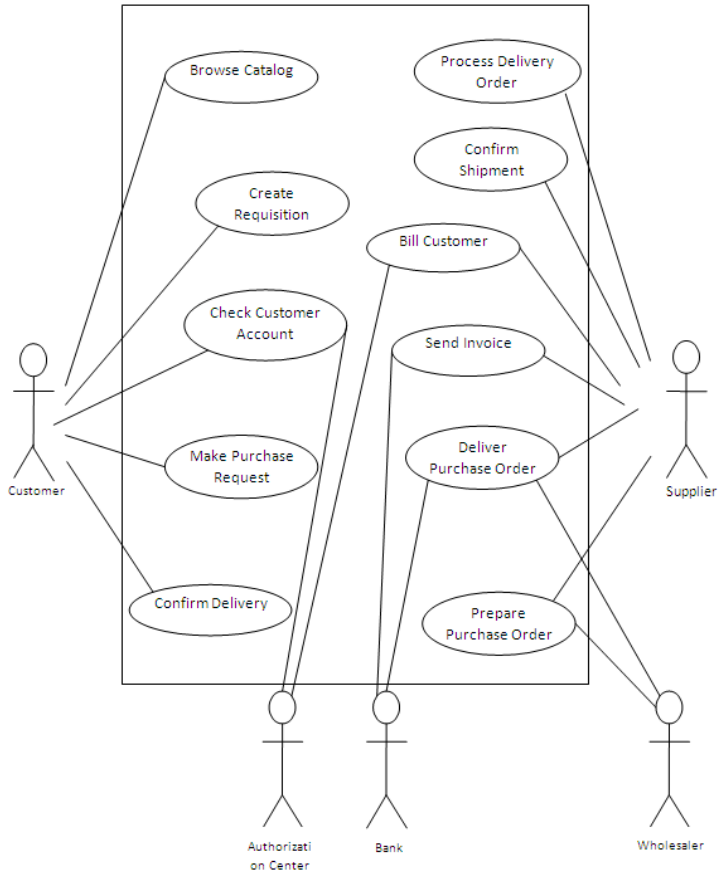

Figura 7.1. Diagrama de casos uso de la LPS Comercio Electrónico.

También se incluye la **descripción** de dos de los **casos de uso** del sistema. En la Tabla 7.1 se muestra la descripción del caso de uso Browse Catalog, en el cual el cliente realiza una búsqueda en el catálogo del proveedor, y en la Tabla 7.2 se muestra la descripción del caso de uso Confirm Shipment, en la cual el proveedor actualiza el estado del envío al sistema y al cliente.

Tabla 7.1.Caso de uso: Browse Catalog.

Nombre	Browse Catalog
Descripción	El cliente busca artículo en el catálogo.
Actores	Cliente
Precondiciones	
Flujo normal 1. El cliente solicita ver un catálogo. 2. El sistema muestra la información del catálogo permitiendo hacer una selección de artículos. 3. El cliente realiza una selección de artículos. 4. El sistema confirma la disponibilidad de los artículos.	
Flujo alternativo 4. El sistema indica al cliente que no hay disponibilidad de los artículos.	
Poscondiciones	

Tabla 7.2. Caso de uso: Confirm Shipment.

Nombre	Confirm Shipment
Descripción	Preparación manual del envío por parte del proveedor.
Actores	Cliente, Proveedor
Precondiciones	
Flujo normal 1. El proveedor introduce la información de envío en el sistema. 2. El sistema actualiza el servidor del inventario. 3. El sistema actualiza el servidor de la orden de envíos. 4. El sistema muestra al cliente el estado del envío.	
Flujo alternativo	
Poscondiciones	

El **diagrama de clases** de la Figura 7.2 muestra las clases que participan en el sistema.

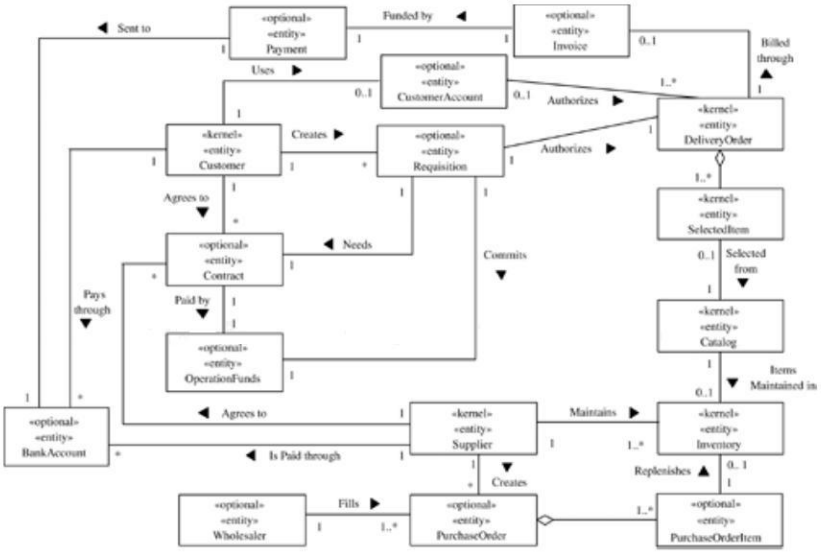

Figura 7.2. Diagrama de clases para la LP Comercio Electrónico

Una vez diseñados estos tres tipos de artefactos básicos de la fase de requisitos, se va a realizar la evaluación. Para este caso de aplicación del método, se ha considerado establecer en la fase de requisitos el modelo de calidad y la selección de las métricas tanto de la fase de requisitos, como de la de diseño de la ingeniería del dominio. No obstatne, se recuerda que el método permite al evaluar que establezca la mejor estrategia para su línea de productos en cuanto a cuando definir los atributos de calidad deseados y seleccionar las métricas. A continuación se detallan los pasos de evaluación.

1. Establecer los requisitos de evaluación

1.1. Establecer el propósito de la evaluación

El propósito de la evaluación es evaluar la característica de calidad Mantenibilidad de la línea de productos Comercio Electrónico en las fases de Requisitos y Diseño, tanto de la ingeniería del dominio como en la ingeniería de la aplicación. No obstante, en esta fase sólo se van a seguir las actividades para la Ingeniería del Dominio, repitiendo estos mismos pasos en la Ingeniería de la Aplicación para el caso concreto de la evaluación para un producto final.

1.2. Identificar los tipos de productos a ser evaluados

Los productos a evaluar son el diagrama de casos de uso, la descripción de un caso de uso, el diagrama de clases y el modelo de características.

1.3. Especificar el modelo de calidad

Siguiendo los requisitos de evaluación de las dos actividades anteriores (*1.1. Establecer el propósito de la evaluación* y *1.2. Identificar los tipos de productos a ser evaluados*) se ha seleccionado del Modelo de Calidad propuesto una instancia del mismo con subcaracterísticas y atributos de la característica Mantenibilidad. En concreto, las subcaracterísticas Modularidad y Reusabilidad. El Modelo de Calidad a utilizar en la Ingeniería el Dominio de esta LPS es el mostrado en la Tabla 7.3.

Tabla 7.3. Modelo de Calidad para la LPS Comercio Electrónico (Ing. Dominio).

7. Mantenibilidad		
Subcaracterísticas		**Atributos**
7.1. Modularidad		7.1.2. Modularidad de la LPS
7.2. Reusabilidad	7.2.1. Comunalidad	7.2.1.1. Comunalidad funcional
		7.2.1.3. Aplicabilidad de un activo
	7.2.2. Variabilidad	7.2.2.1. Cubrimiento de la variabilidad en un activo
		7.2.2.2. Complejidad de la variabilidad de la LPS

Nótese que se ha mantenido la numeración original del Modelo de Calidad propuesto a fin de localizar mejor estas subacaracterísticas y atributos.

2. Especificar la evaluación

En este paso se seleccionan las métricas para evaluar los atributos de calidad que se desean conocer, se establecen los niveles de ratio de las métricas y se establecen los criterios para el aseguramiento.

2.1. Seleccionar las métricas

En base a los atributos de calidad escogidos y los artefactos que van a evaluarse se realiza la selección de las métricas, la cual se recoge en la Tabla 7.4.

Tabla 7.4. Correspondencia entre atributos y métricas para Comercio Electrónico en la ID.

Atributo	Métrica
7.1.2. Modularidad de la LPS	Ratio de activos por característica
7.2.1.1. Comunalidad funcional	Cubrimiento funcional de un activo
7.2.1.3. Aplicabilidad de un activo	Aplicabilidad acumulativa
7.2.2.1. Cubrimiento de la variabilidad en un activo	Cubrimiento de la variabilidad
7.2.2.2. Complejidad de la variabilidad de la LPS	Puntos de variación en un caso de uso
	Número de variabilidades en la LPS
	Variabilidad en las clases de un diagrama de clases.

En las tablas comprendidas entre la Tabla 7.5 y la Tabla 7.11 se encuentra una descripción más detallada de cada métrica indicando el artefacto evaluado.

2.2. Establecer los niveles de ratio de las métricas

Los posibles valores de las métricas están comprendidos desde el 0 hasta el 1 conteniendo valores continuos. La información de cada métrica se muestra en una tabla. Las tablas comprendidas desde la Tabla 7.5 hasta la Tabla 7.11 muestran la descripción de dichas métricas.

2.3. Establecer los criterios para el aseguramiento

Los valores establecidos como aceptables para los atributos evaluados son los mostrados en las tablas comprendidas desde la Tabla 7.5 hasta la Tabla 7.11. Son valores subjetivos que son especificados por los expertos del dominio.

Tabla 7.5. Descripción de la métrica Número de activos en una LPS.

Atributo	**Modularidad de la LPS**			
Métrica	Ratio de activos por característica			
Descripción	Contar el número de activos del que está compuesta una LPS.			
Operacionalización				
Artefacto	**Fase**	**Fórmula**	**Umbrales**	**Valores aceptados**
Cantidad de activos. Modelo de características.	DD	X = A/B A = Contar el número de activos que se deben desarrollar en la LPS. B = Contar el número de características en el modelo de características.	0 - 1	X > 0.7

Tabla 7.6. Descripción de la métrica Cubrimiento funcional de un activo.

Atributo	**Comunalidad funcional**			
Métrica	Cubrimiento funcional de un activo			
Descripción	Mide la media de la comunalidad de cada característica funcional en un activo. La comunilidad de cada característica funcional puede ser medida por el cálculo del grado de miembros utilizando cada característica funcional.			
Operacionalización				
Artefacto	**Fase**	**Fórmula**	**Umbrales**	**Valores aceptados**
Activo	DD	FC = S/N A = Número de aplicaciones utilizando la característica funcional i B = Número total de aplicaciones en la línea de productos. S = Sumatorio de A/B para cada i, tal que i=1 hasta N. N = Número total de características funcionales.	0 - 1	FC > 0.7

Tabla 7.7. Descripción de la métrica Aplicabilidad acumulativa.

Atributo	**Aplicabilidad de un activo**			
Métrica	Aplicabilidad acumulativa			
Descripción	Mide cómo de aplicable es un activo para desarrollar varias aplicaciones.			
Operacionalización				
Artefacto	**Fase**	**Fórmula**	**Umbrales**	**Valores aceptados**
Modelo de características. Activo.	DD	CA = A * FC + C*CV A= 0.5 (Peso de FC) FC = Cubrimiento funcional de un activo C = 0.5 (Peso de CV) CV = Cubrimiento de la variabilidad en un activo	0 - 1	CA > 0.5

Tabla 7.8. Descripción de la métrica Cubrimiento de la variabilidad.

Atributo	**Cubrimiento de la variabilidad en un activo**			
Métrica	Cubrimiento de la variabilidad			
Descripción	Mide cuantos puntos de variación incluidos en el alcance de la LPS son realmente implementados en el activo.			
Operacionalización				
Artefacto	**Fase**	**Fórmula**	**Umbrales**	**Valores aceptados**
Activo. Modelo de Características	DD	CV = A/B A = Número de puntos implementados en el activo. B = Número de puntos de variación incluidos en el alcance de la LPS	0 - 1	CV < 0.3

Tabla 7.9. Descripción de la métrica Puntos de variación en un caso de uso.

Atributo	**Complejidad de la variabilidad**			
Métrica	Puntos de variación en un caso de uso			
Descripción	Obtener los puntos de variación en un caso de uso.			
Operacionalización				
Artefacto	**Fase**	**Fórmula**	**Umbrales**	**Valores aceptados**
Caso de uso	RD	PV= A/B A = Número de actividades alternativas. B = Número de actividades totales.	0 - 1	PV < 0.4

Tabla 7.10. Descripción de la métrica Número de variabilidades en la LPS.

Atributo	**Complejidad de la variabilidad**			
Métrica	Número de variabilidades en la LPS			
Descripción	Ratio del número de características alternativas u opcionales frente al número total de características.			
Operacionalización				
Artefacto	**Fase**	**Fórmula**	**Umbrales**	**Valores aceptados**
Modelo de características	DD	V = (A+B)/N A = Número de características alternativas. B = Número de características opcionales. N = Número total de características.	0 – 1	V > 0.5

Tabla 7.11. Descripción de la métrica Variabilidad en las clases de un diagrama de clases.

Atributo	**Complejidad de la variabilidad**			
Métrica	Variabilidad en las clases de un diagrama de clases.			
Descripción	Ratio del número de clases que son opcionales frente al número total de clases del diagrama de clases			
Operacionalización				
Artefacto	**Fase**	**Fórmula**	**Umbrales**	**Valores aceptados**
Diagrama de clases	RD	V = (A+B)/N A = Número de clases opcionales (pueden no ser instanciadas nunca en el sistema, se muestran con cardinalidad mínima igual a 0) N = Número total de clases.	0 – 1	0.1<V<0.5

3. Diseñar el plan de evaluación

En esta fase se ha decidido especificar el plan de evaluación para las fases de Requisitos y Diseño de la Ingeniería del Dominio. En las tablas comprendidas entre la Tabla 7.5 y la Tabla 7.11 se indica en qué fase del ciclo de vida aplicar cada métrica, indicando con las siglas RD la fase de Requisitos en la Ingeniería del Dominio y con las siglas DD la fase de Dominio en la Ingeniería de la Aplicación. A continuación se procede a indicar los pasos a seguir en la evaluación para cada una de las fases:

- Requisitos en la Ingeniería del Dominio:
  1. Seleccionar las métricas que se evalúan sobre los diagramas de casos de uso, la descripción del caso de uso y los diagramas de clase.
  2. Identificar las métricas base, las derivadas y las dependencias entre ellas.
  3. Identificar los atributos que son evaluados con las métricas seleccionadas.
  4. Identificar los artefactos a medir.
  5. Para cada atributo, tomar las medidas de forma que se realice primero la medición de las métricas base y después las derivadas, teniendo en cuenta sus dependencias.
- Diseño en la Ingeniería del Dominio:
  1. Seleccionar las métricas que se evalúan sobre los modelos de características.
  2. Identificar las métricas base, las derivadas y las dependencias entre ellas.
  3. Identificar los atributos que son evaluados con las métricas seleccionadas.
  4. Identificar los artefactos a medir.
  5. Para cada atributo, tomar las medidas de forma que se realice primero la medición de las métricas base y después las derivadas, teniendo en cuenta sus dependencias.

4. Ejecución de la evaluación

Se ha realizado la evaluación de las métricas establecidas para la fase de Requisitos, dejando el resto de métricas para la fase de Diseño, tal y como se ha especificado en la descripción de cada métrica.

4.1. Ejecución y anotación de las medidas

Siguiendo el plan de evaluación, se han seleccionado las métricas a medir sobre los artefactos creados en la fase de requisitos. Los resultados se muestran en la Tabla 7.12.

4.2. Comparación con los criterios

Los resultados obtenidos de la medición han sido comparados con los valores de aceptación de cada una de las métricas. En la Tabla 7.12 se muestra una corta conclusión del valor de cada métrica de forma individual.

Tabla 7.12. Anotación de las medidas y comparación con los criterios (Ing. Dominio).

Métrica	Artefacto evaluado	Resultado	Conclusión
Puntos de variación en un caso de uso	Caso de uso: Browse Catalog	PV = 1/5 = 0.2	El valor de los puntos de variación del caso de uso es bajo, con lo que el caso de uso realiza unas funcionalidades concretas y no recoge mucha variabilidad en él.
	Caso de uso: Confirm Shipment	PV = 0/5 = 0	El valor de los puntos de variación del caso de uso es nulo, con lo que el caso de uso realiza unas funcionalidades muy concretas y no recoge variabilidad en él.
Variabilidad en las clases de un diagrama de clases	Diagrama de clases	V = 1/16 = 0.0625	El valor obtenido está fuera de los rangos aceptados. El resultado muestra que hay muy poca variabilidad y que la mayoría de los productos finales utilizarán la totalidad de las clases del diagrama de clases.

4.3. Aseguramiento de los resultados

Los resultados respecto a la variabilidad de los casos de uso son bajos, por lo que éstos realizan funcionalidades muy concretas. Es un resultado aceptable puesto que cada caso de uso podría verse como un activo, ya que sus funciones son independientes entre sí. Pueden verse como cajas negras que realizan una función muy concreta, con lo que su mantenibilidad es buena. No obstante, sería preferible aumentar su variabilidad ya que actualmente no contemplan casos con fallos o con respuestas del usuario no esperados.

El diagrama de clases tiene una variabilidad muy baja, de tal forma que puede considerarse un artefacto común a toda la LPS.

7.3.1.2. Diseño en la Ingeniería del Dominio

El **modelo de características** que describe la LPS Comercio Electrónico es mostrado en la Figura 7.3.

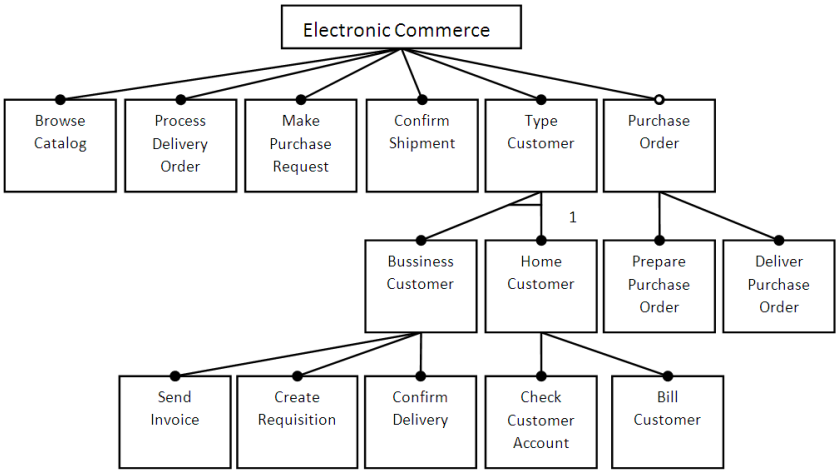

Figura 7.3. Modelo de características de la LPS Comercio Electrónico.

4. Ejecución de la evaluación

4.1. Ejecución y anotación de las medidas

Siguiendo el plan de evaluación, se han seleccionado las métricas a aplicar en la fase de diseño. Los resultados se muestran en la Tabla 7.13.

4.2. Comparación con los criterios

Los resultados obtenidos de la medición han sido comparados con los valores de aceptación de cada una de las métricas. En la

Tabla 7.13 se muestra una corta conclusión del valor de cada métrica de forma individual.

Tabla 7.13. Anotación de las medidas y comparación con los criterios (Ing. Aplicación).

Métrica	Artefacto evaluado	Resultado	Conclusión
Ratio de activos por característica	Cantidad de activos. Modelo de características.	X = 11/11 = 1	El valor es aceptado
Cubrimiento funcional de un activo	Activo: Browse Catalog	FC = ((4/4)*1)/11=0.09	El valor del cubrimiento funcional es muy bajo. El activo realiza pocas funcionalidades en relación a las funcionalidades de la LP.
	Activo: Confirm Shipment	FC = ((4/4)*11)/1=0.09	El valor del cubrimiento funcional es muy bajo. El activo realiza pocas funcionalidades en relación a las funcionalidades de la LP.
Cubrimiento de la variabilidad	Activo: Browse Catalog. Modelo de características.	CV = 1/11 = 0.09	El cubrimiento de la variabilida es bajo, pero entra en los valores aceptados.
	Activo: Confirm Shipment. Modelo de características.	CV = 1/11 = 0.09	El cubrimiento de la variabilida es bajo, pero entra en los valores aceptados.
Aplicabilidad acumulativa	Activo: Browse Catalog. Modelo de características.	CA = 0.5 * 1 + 0.5 * 0.09 = 0.59	El activo es lo suficientemente aplicable y es rentable introducirlos en la LPS.
	Activo: Confirm Shipment. Modelo de características.	CA = 0.5 * 1 + 0.5 * 0.11 = 0.59	El activo es lo suficientemente aplicable y es rentable introducirlos en la LPS.
Número de variabilidades en la LPS	Modelo de características	V = (2+1)/11 = 0.17	La LPS ofrece muy poca variabilidad. El valor obtenido no entra en los límites considerados aceptables.

4.3. Aseguramiento de los resultados

De los resultados se desprende que la variabilidad es muy baja, por lo que se están haciendo esfuerzos para llevar a cabo el desarrollo mediante línea de productos pero en realidad se crea un conjunto muy reducido de posibles productos. El ratio de activos por característica es 1, por lo que cada activo representa una característica. Tal vez podría verse la posibilidad de descomponer las características en más activos para poder reutilizar esos mismos activos para varias características.

7.3.2. Evaluación en la Ingeniería de la Aplicación

En la fase de la Ingeniería de la Aplicación se procede a la creación de un producto final a partir de unos requisitos dados y la reutilización de los activos establecidos en la fase de la Ingeniería del Dominio.

De este modo se desea crear un sistema de comercio electrónico para la empresa CompraNet. La web de la empresa debe mostrar el catálogo a los potenciales clientes y permitirles realizar búsquedas. La web no contiene la posibilidad de que los clientes creen cuentas, así que cada vez que un usuario desee comprar un artículo deberá facilitar sus datos y un número de tarjeta de crédito válida. Tras realizar una compra por parte del cliente y realizar las correspondientes validaciones (la existencia de suficientes artículos y la validez de la tarjeta de crédito), el proveedor envía los artículos al cliente a través de un servicio de mensajería y la factura de compra a través de correo postal.

7.3.2.1. Análisis y Requisitos en la Ingeniería de la Aplicación

A partir de la descripción del producto, se ha detectado la necesidad de los siguientes casos de uso:

- Browse catalog
- Process Delivery Order
- Make Purchase Request
- Check Customer Account
- Bill Customer
- SendInvoice

En la Figura 7.4 se muestra el diagrama de casos de uso del sistema de comercio electrónico de CompraNet. Nótese que el caso de uso SendInvoice no está disponible en la LPS.

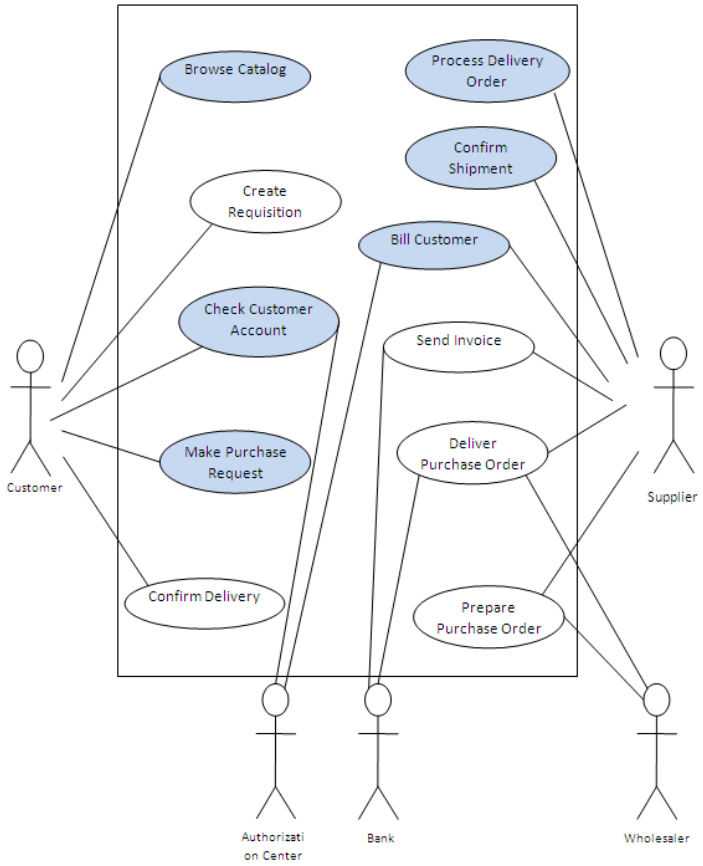

Figura 7.4. Diagrama de casos de uso del producto.

En esta ocasión, el evaluador ha considerado especificar en la fase de requisitos sólo lo referente a esta fase en cuanto a evaluación. Estos son los pasos que ha seguido.

1. Establecer los requisitos de evaluación

1.1. Establecer el propósito de la evaluación

El propósito de la evaluación es evaluar la característica de calidad Idoneidad Funcional, en concreto las subcaracterísticas Adecuación del producto en relación con la LPS.

1.2. Identificar los tipos de productos a ser evaluados

El producto a evaluar es el diagrama de casos de uso.

1.3. Especificar el modelo de calidad

Siguiendo los requisitos de evaluación de las dos actividades anteriores (*1.1. Establecer el propósito de la evaluación* y *1.2. Identificar los tipos de productos a ser evaluados*) se ha seleccionado del Modelo de Calidad propuesto una instancia del mismo con los atributos de las

subcaracterísticas Adecuación. El Modelo de Calidad a utilizar en la Ingeniería de la Aplicación es el mostrado en la Tabla 7.18.

Tabla 7.14. Modelo de Calidad para el producto ComprarNet.

1. Funcionalidad	
Subcaracterísticas	**Atributos**
1.1. Adecuación	1.1.2.1. Adecuación de los requisitos funcionales

2. Especificar la evaluación

En este paso se seleccionan las métricas para evaluar los atributos de calidad que se desean conocer, se establecen los niveles de ratio de las métricas y se establecen los criterios para el aseguramiento.

2.1. Seleccionar las métricas

En base a los atributos de calidad escogidos y los artefactos que van a evaluarse se realiza la selección de las métricas, las cuales se recogen en la Tabla 7.19.

Tabla 7.15. Correspondencia entre atributos y métricas para CompraNet.

Atributo	Métrica
1.1.2.1. Adecuación de los requisitos funcionales	Ratio del número de requisitos funcionales del producto disponibles en la LPS.

En la Tabla 7.20 se encuentra una descripción más detallada de la métrica.

2.2. Establecer los niveles de ratio de las métricas

Los posibles valores están comprendidos desde el 0 hasta el 1 pudiendo tomar valores continuos entre ellos.

2.3. Establecer los criterios para el aseguramiento

El valor considerado como aceptable es mayor que 0.7.

Tabla 7.16. Descripción de la métrica Ratio del número de requisitos funcionales del producto disponibles en la LPS.

Atributo	**Adecuación de los requisitos funcionales**			
Métrica	Ratio del número de requisitos funcionales del producto disponibles en la LPS.			
Descripción	Contar el número de requisitos funcionales del producto que se encuentran en la LPS sin tener que modificarlos y dividirlo entre el número total de requisitos funcionales del producto.			
Operacionalización				
Artefacto	**Fase**	**Fórmula**	**Umbrales**	**Valores aceptados**
Requisitos del producto. Requisitos de la LPS.	RD	X = A/B A = Número de requisitos funcionales del producto. B = Número de requisitos de la LPS.	0 - 1	X > 0.7

4. Ejecución de la evaluación

4.1. Ejecución y anotación de las medidas

Siguiendo el plan de evaluación se procede a realizar la medición y a anotar la medida en la Tabla 7.17.

4.2. Comparación con los criterios

Los resultados obtenidos de la medición han sido comparados con los valores de aceptación de cada una de las métricas. En la Tabla 7.17 se muestra una corta conclusión del valor de la métrica.

Tabla 7.17. Anotación de las medidas y comparación con los criterios (Ing. Dominio).

Métrica	**Artefacto evaluado**	**Resultado**	**Conclusión**
Ratio del número de requisitos funcionales del producto disponibles en la LPS.	Requisitos del producto. Requisitos de la LPS.	X = 5/6 = 0.83	El producto tiene un alto grado de adecuación con la LPS.

4.3. Aseguramiento de los resultados

El producto a construir tiene un grado de adecuación considerable para ser construido con la LPS de comercio electrónico. Por lo tanto, se pasa a la siguiente fase de desarrollo del producto final.

7.3.2.2. Diseño en la Ingeniería de la Aplicación

La especificación de requisitos está muy ligada a la selección de las características. Los requisitos indican que deben seleccionarse los activos

- Browse catalog
- Process Delivery Order
- Make Purchase Request
- Confirm Shipment
- Home Customer (el cual incorpora Check Customer Account, Bill Customer)

La visualización del modelo de características con las características seleccionadas se muestra en la Figura 7.5.

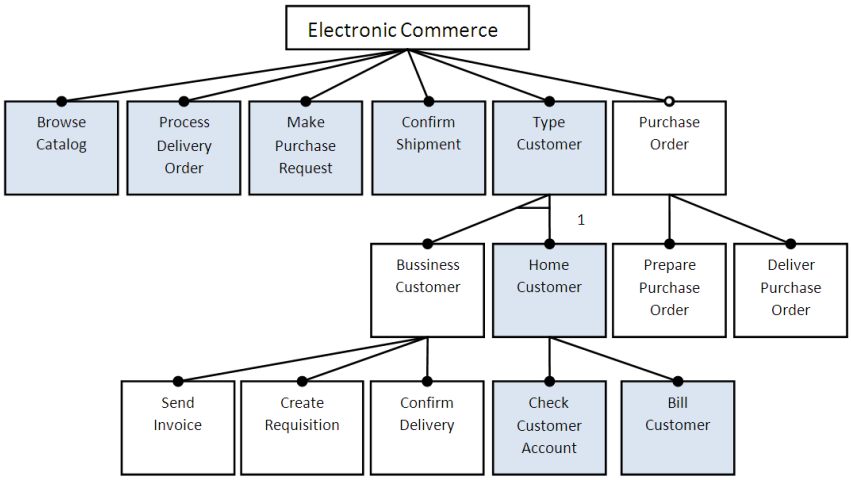

Figura 7.5. Selección de características.

Como puede observarse, ha sido necesario incorporar la característica Confirm Shipment aunque no estuviera incluida en la especificación. Por otra parte, tampoco está incluida la funcionalidad del envío postal de la factura, por lo tanto pasa a considerarse una característica propia del producto.

La característica propia se ha denominado Send Invoice y para su diseño se pretende considerar la reutilización de algún activo.

A continuación se describe el proceso de evaluación.

1. Establecer los requisitos de evaluación

1.1. Establecer el propósito de la evaluación

El propósito de la evaluación es evaluar la característica de calidad Mantenibilidad, en concreto la subcaracterística Reusablidad.

1.2. Identificar los tipos de productos a ser evaluados

El producto a evaluar es el modelo de características.

1.3. Especificar el modelo de calidad

Siguiendo los requisitos de evaluación de las dos actividades anteriores (*1.1. Establecer el propósito de la evaluación* y *1.2. Identificar los tipos de productos a ser evaluados*) se ha seleccionado del Modelo de Calidad propuesto una instancia del mismo con los atributos de Reusabilidad. El Modelo de Calidad a utilizar en la Ingeniería de la Aplicación es el mostrado en la Tabla 7.18.

Tabla 7.18. Modelo de Calidad para el producto ComprarNet.

7. Mantenibilidad		
Subcaracterísticas		**Atributos**
7.2. Reusabilidad	7.2.3. Cubrimiento de un producto	7.2.3.1. Nivel de cubrimiento de un producto con la LPS
		7.2.3.2. Reutilización en características propias

2. Especificar la evaluación

En este paso se seleccionan las métricas para evaluar los atributos de calidad que se desean conocer, se establecen los niveles de ratio de las métricas y se establecen los criterios para el aseguramiento.

2.1. Seleccionar las métricas

En base a los atributos de calidad escogidos y los artefactos que van a evaluarse se realiza la selección de las métricas, la cual se recoge en la Tabla 7.19.

Tabla 7.19. Correspondencia entre atributos y métricas para CompraNet.

Atributo	**Métrica**
7.2.3.1. Nivel de cubrimiento de un producto con la LPS	Ratio de activos reutilizados para crear un nuevo producto.
7.2.3.2. Reutilización en características propias	Estimación del ratio de funciones reutilizadas para crear un nuevo activo.

En Tabla 7.20 y Tabla 7.21 se encuentra una descripción más detallada de cada métrica indicando el artefacto evaluado.

2.2. Establecer los niveles de ratio de las métricas

Los posibles valores de las métricas están comprendidos desde el 0 hasta el 1 conteniendo valores continuos. La información de cada métrica se muestra en una tabla.

2.3. Establecer los criterios para el aseguramiento

Los valores establecidos como aceptables para los atributos evaluados son los mostrados en Tabla 7.20 y Tabla 7.21.

Tabla 7.20. Descripción de la métrica Número de activos en una LPS.

Atributo	**Nivel de cubrimiento de un producto con la LPS**			
Métrica	Ratio de activos reutilizados para crear un nuevo producto.			
Descripción	Contar el número de activos directamente reutilizados para formar un producto.			
Operacionalización				
Artefacto	**Fase**	**Fórmula**	**Umbrales**	**Valores aceptados**
Modelo de características. Requisitos del producto.	DD	X = A/B A = Número de activos utilizados por el producto sin modificarlos. B = Número de activos necesarios para crear el producto (reutilizados o creados).	0 - 1	X > 0.7

Tabla 7.21. Descripción de la métrica Número de activos en una LPS.

Atributo	**Reutilización en características propias**			
Métrica	Estimación del ratio de funciones reutilizadas para crear un nuevo activo.			
Descripción	Cantidad de componentes de un activo que pueden ser reutilizados para crear un activo propio de un producto final.			
Operacionalización				
Artefacto	**Fase**	**Fórmula**	**Umbrales**	**Valores aceptados**
Componentes de los activos con funcionalidades similares. Componentes nuevos del activo propio.	DD	X = A/B A = Número de componentes reutilizables de un activo existente para crear un activo propio. B = Número de componentes nuevos del activo propio.	0 - 1	No se requiere ningún valor inicial. Se trataría de maximizar el valor.

4. Ejecución de la evaluación

4.1. Ejecución y anotación de las medidas

Siguiendo el plan de evaluación, se han seleccionado las métricas a aplicar en la fase de diseño. Los resultados se muestran en la

4.2. Comparación con los criterios

Los resultados obtenidos de la medición han sido comparados con los valores de aceptación de cada una de las métricas. En la Tabla 7.13 se muestra una corta conclusión del valor de cada métrica de forma individual.

Tabla 7.22. Anotación de las medidas y comparación con los criterios (Ing. Dominio).

Métrica	Artefacto evaluado	Resultado	Conclusión
Ratio de activos reutilizados para crear un nuevo producto.	Modelo de características. Requisitos del producto.	X = 6/7 = 0.85	El nivel de reutilización es aceptable.
Estimación del ratio de funciones reutilizadas para crear un nuevo activo.	Componentes de los activos con funcionalidades similares. Componentes nuevos del activo propio.	X = 3/5 = 0.6 (Reutilización de Confirm Shipment)	Es un nivel razonable de reutilización.

4.3. Aseguramiento de los resultados

El nivel de reutilización para crear este producto se considera alto. Por una parte, está prácticamente construido con activos que provienen de la LPS. Y por otra, ha sido posible la reutilización de un activo con funcionalidades similares para crear una característica propia.

7.4. Conclusiones

Se ha aplicado el método para la evaluación de la calidad utilizando el modelo de calidad propuesto en este mismo trabajo. Las primeras conclusiones son que es un método aplicable y que es capaz de utilizarse para la evaluación de la calidad en líneas de productos software. El método es general y puede ser aplicado a cualquier desarrollo de líneas de productos software. Además, es flexible permitiendo ejecutar las actividades de evaluación en cualquier fase del desarrollo de la línea de productos software. Es una ventaja no tener un método demasiado restrictivo, ya que, de lo contrario podría malgastarse tiempo en tareas de evaluación que posteriormente no serán efectivas. Por ejemplo, en este caso se ha considerado especificar la evaluación de sólo los requisitos en la

fase de requisitos y estudiar la idoneidad funcional del producto final con la línea de productos. Si se hubiese considerado especificar la evaluación para todas las fases del desarrollo, podría ocurrir que ese producto no tuviera suficientes requisitos comunes con la línea de productos, y fuese más económico en costes y tiempo realizarlo mediante un desarrollo de software tradicional.

Finalmente se observa que la calidad y la completitud de la descripción de la línea de producto en la que se aplique el método condiciona fuertemente la calidad de la evaluación realizada, ya que cuando más especificados estén los requisitos, los activos, la arquitectura, etc. más atributos de calidad pueden ser medidos.

Capítulo 8

Conclusiones

Las conclusiones del trabajo son mostradas en este capítulo. Se recuerdan los objetivos propuestos al principio del trabajo y se comprueba si se han cumplido y en qué grado. Además, se presentan los artículos generados a partir de este trabajo de investigación. Por último, la investigación actual y los trabajos futuros son expuestos.

8.1. Contribuciones

Los objetivos que se han planteado en este trabajo son:

1. Conocer el estado actual en métodos y técnicas para la evaluación de la calidad en el ámbito de líneas de productos software.
2. Conocer los atributos de calidad que los investigadores han considerado relevantes para la evaluación de la calidad en las líneas de productos software.
3. Conocer las métricas que han sido propuestas para evaluar la calidad en las líneas de productos software.
4. Proponer un Modelo de Calidad para líneas de productos software que recoja el estado actual, cubra las deficiencias encontradas en el estado del arte, y que siga los estándares de calidad más actuales.
5. Proponer un método para evaluar la calidad de líneas de productos software que utilice el Modelo de Calidad propuesto.

Respecto al primero objetivo, se ha realizado una revisión sistemática para reunir el conocimiento actual sobre los métodos y técnicas para la evaluación de la calidad en LPS. Como se introdujo en el Capítulo 1, una revisión sistemática es un método objetivo y replicable para obtener información relevante en un área de estudio específica. El estudio es relevante tanto para investigadores como profesionales de la industria. Los resultados permiten localizar los métodos y técnicas disponibles para la evaluación de LPS, identificar las ventajas e inconvenientes de cada método en función de las necesidades de los usuarios (por ejemplo, en función de la representación utilizada para representar la variabilidad), y descubrir las carencias existentes para direccionar los esfuerzos de investigación.

Para el segundo y tercer objetivos se ha realizado otro estudio siguiendo la metodología de la revisión sistemática para conocer no sólo los atributos de calidad propuestos por los investigadores y profesionales de la industria sino también las métricas que han sido utilizadas por los mismos en el ámbito de las LPS. De nuevo, este estudio permite conocer el estado actual del área y es de interés tanto para investigadores como profesionales de la industria.

Ambos trabajos han supuesto un punto de partida indispensable para la consecución de los objetivos cuarto y quinto. Por una parte, el conocimiento de los métodos y técnicas existentes ha permitido direccionar el trabajo posterior al comprobar que no existe ningún método ni técnica que recoja todas las características deseables para la evaluación de LPS. Por otra parte, el conocimiento de los atributos de calidad y

métricas ha supuesto un punto de partida para la definición del Modelo de Calidad propuesto. En definitiva, conocer el estado actual permite llegar a la consecución de un trabajo más completo y de mayor calidad ya que (1) no se gastan esfuerzos en proponer trabajos que ya existen en la literatura, (2) se aprende sobre los trabajos propuestos que no han tenido éxito, y (3) es posible incorporar o mejorar trabajos ya propuestos intentando mejorar sus debilidades.

El Modelo de Calidad propuesto para LPS es la consecución del cuarto objetivo. Construir un modelo de calidad es bastante complejo y es usual que estos modelos descompongan la calidad del producto software jerárquicamente en una serie de características, subcaracterísticas y atributos que pueden usarse como una lista de comprobación de aspectos relacionados con la calidad. Se han desarrollado varios modelos de calidad para diferentes productos y procesos software pero ninguno de ellos es específico para las LPS.

El modelo de calidad propuesto en ese trabajo sigue el estándar de calidad SQuaRE e incorpora los resultados obtenidos de la revisión sistemática sobre atributos y métricas de calidad. Se trata de un modelo de calidad para LPS genérico, es decir, puede utilizarse para evaluar cualquier línea de producto software y tiene en cuenta criterios para satisfacer las necesidades de los desarrolladores, mantenedores, adquisidores y usuarios finales. Además, se ha recogido y propuesto un conjunto de métricas para la evaluación de los atributos del modelo de calidad. Para cada métrica se han indicado posibles operacionalizaciones indicando sobre qué artefacto(s) se debería realizar la evaluación y en qué fase(s) del ciclo de vida de la LPS.

El sexto objetivo ha sido materializado con Un Método para Evaluar la Calidad de las LPS que hace uso del Modelo de Calidad propuesto anteriormente. La definición del método es genérica, por lo que puede ser aplicado para evaluar cualquier LPS. No obstante, una de las metas que se desea alcanzar es realizar la evaluación siguiendo un enfoque dirigido por modelos. Es por ello que se presenta brevemente el plan de producción utilizando dicho método y el modelo de calidad. Una de las principales aportaciones del método, es que la calidad no es vista como un proceso externo al desarrollo del software, sino que forma parte del mismo proceso de desarrollo de software. Así, el modelo de calidad es visto como una vista más del sistema y se comporta como un artefacto "activo" durante todo el ciclo de vida.

Sintetizando, las contribuciones tienen un doble resultado. Por una parte es un estudio del estado actual en el ámbito de las LPS del cual pueden beneficiarse tanto investigadores como la industria, ya que se ha realizado el estudio siguiendo una metodología clara, replicable y objetiva. Por otra parte, se ha propuesto un modelo de calidad y un método para la evaluación de LPS siguiendo los últimos estándares de calidad.

8.2. Publicaciones relacionadas

El presente trabajo ha generado hasta el momento dos publicaciones:

- Sonia Montagud, Silvia Abrahão. Gathering Current Knowledge about Quality Evaluation in Software Product Lines. Proceedings of the **13th International Software Product Line Conference (SPLC 2009)**, August 24–28, San Francisco-CA, USA, J. D. McGregor and D. Muthing (Eds.), IEEE Press, pp. 91-100, 2009, ISBN 978-0-9786956-2-0.
- Sonia Montagud, Silvia Abrahão. Revisión Sistemática de Métricas de Calidad para Líneas de Productos Software. Antonio Vallecillo, Goiuria Sagardui (Eds.): **XIV Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2009)**, San Sebastián, Spain, Septiembre 8-11, 2009, pp. 275-280, ISBN 978-84-692-4211-7.

La primera publicación presenta un estudio sobre métodos y técnicas de evaluación de calidad para líneas de producto software que han sido propuestos en los últimos 10 años. **SPLC es la conferencia líder internacional sobre Líneas de Producto Software**. El comité científico es internacional. El proceso de revisión es externo por pares. El porcentaje de aceptación es del 25%. La editorial es internacional de prestigio. La edición del 2009 ha sido organizada por el Software Engineering Institute (SEI) de la Universidad Carnegie Mellon (USA). La conferencia aparece listada por el DBLP.

La segunda publicación presenta la primera versión del Modelo de Calidad para líneas de producto software que ha sido obtenido mediante una revisión sistemática que ha reunido los atributos y métricas de calidad para LPS que han sido propuestos en la literatura desde el año 1996. El conocimiento de esta información permite detectar los atributos considerados más relevantes por los investigadores y las métricas propuestas para medirlos. **JISBD es la conferencia nacional más relevante en el ámbito de la Ingeniería del Software** y un foro de discusión entre investigadores españoles. El trabajo ha sido presentado en una sesión específica sobre líneas de producto software.

Actualmente, estamos trabajando en la redacción de un artículo para la revista Information and Software Technology que recogerá la última versión del Modelo de Calidad para LPS presentado en esta tesina y su instanciación a un proceso de desarrollo basado en multimodelos.

8.3. Conclusiones y trabajos futuros

El modelo de calidad y el método de evaluación propuestos conforman una aproximación para la evaluación de calidad de líneas de productos software. Esta aproximación, a diferencia de las propuestas hasta ahora, ha tenido en cuenta aspectos concretos de las LPS (como la

variabilidad, la composicionalidad, la evaluación de activos individuales, la evaluación de la arquitectura, la evaluación durante todo el ciclo de vida, etc.) para todas las características de calidad que son consideradas de relevancia para los productos software. Además, la propuesta se ha realizado siguiendo el reconocido estándar de calidad SQuaRE. No obstante, se sigue investigando tanto en el modelo de calidad como en el método.

Con respecto al modelo de calidad, se pretende validar teórica y empíricamente las métricas del modelo. Además, se pretende definir dos vistas más del modelo de calidad propuesto (centrada en calidad de proceso y calidad en uso) y estudiar las relaciones entre ellas. El modelo de calidad propuesto se centra en la evaluación del producto, y existe un interés en evaluar también la calidad del proceso que se utiliza para obtener los productos así como la calidad en uso (calidad de los productos utilizados en un contexto específico).

Con respecto al método, se pretende definir la estrategia de transformaciones de modelos y desarrollar una herramienta destinada a que los usuarios finales puedan seleccionar los productos y los atributos de calidad que crean convenientes, de manera que esta selección guie el plan de producción del producto final. Este trabajo constituye uno de los objetivos principales del proyecto de investigación MULTIPLE (Multimodeling Approach for Quality-Aware Software Product Lines, con referencia TIN2009-13838) en el que se está trabajando actualmente.

Además, se pretende validar el modelo de calidad y el método propuesto tanto teórica como empíricamente mediante la realización de experimentos controlados y casos de estudio en entornos industriales. Actualmente diversas empresas se han interesado por el modelo de calidad y el método propuestos. En concreto, la empresa Rolls–Royce participa actualmente en el proyecto MULTIPLE y ha manifestado claramente su interés por este trabajo.

Finalmente, se espera que el trabajo presentado evolucione favorablemente, de manera que nuevas líneas de investigación en las que actualmente no se ha pensado, puedan surgir.

Bibliografía

[1] Abdelmoez, W., Nassar, D.M., Shereschevsky, M., Gradetsky, N., Gunnalan, R., Ammar, H.H., Yu, B., Mili, A.: Error Propagation In Software Architectures. In 10th International Symposium on Software Metrics (METRICS), Chicago, Illinois, USA, 2004.

[2] Ajila, S. A., Dumitrescu, R. T.: Experimental use of code delta, code churn, and rate of change to understand software product line evolution. Journal of Systems and Software 80, pp.74-91, 2007.

[3] Alves de Oliveira Junior, E., Gimenes, I. M. S., Maldonado, J. C.: A Metric Suite to Support Software Product Line Architecture Evaluation. In XXXIV Conferencia Latinamericana de Informática (CLEI), Santa Fé, Argentina, pp.489-498, 2008.

[4] Aldekoa, G., Trujillo, S., Sagardui, G., Díaz, O.: Quantifying Maintanibility in Feature Oriented Product Lines, Athens, Greece, pp.243-247, 2008.

[5] M. A. Babar. "Evaluating Product Line Architectures: Methods and Techniques". 14th Asia-Pacific Software Engineering Conference, 2007, pp. 13.

[6] J. Bayer, O. Flege, and C. Gacek. Creating product line architectures. In F. van der Linden, editor, Software Architectures for Product Families, International Workshop IW-SAPF-3, volume 1951 of Lecture Notes in Computer Science, pages 210–216. Springer, 2000.

[7] D. Benavides, S. Segura, P. Trinidad and A. Ruiz-Cortés. "FAMA: Tooling a Framework for the automated analysis of feature models". In 1st Internacional Workshop on Variability Modelling of Software Intensive Systems, 2007, pp.129-134.

[8] Benavides, D., Trinidad, P. And Ruiz-Cortés, A. Automated Reasoning on Feature Models. In 17th International Conference in Advanced Information Systems Engineering (CAiSE), pages 491-503, 2005.

[9] Boehm, B. W., Brown, J. R., Kaspar, H., Lipow, M., McLeod, G., and Merritt, M., Characteristics of Software Quality, North Holland, 1978.

[10] Boehm, Barry W., Brown, J. R, and Lipow, M.: Quantitative evaluation of software quality, International Conference on Software Engineering, Proceedings of the 2nd international conference on Software engineering, 1976.

[11] J. Bosch. "Design and use of software architectures: adopting and evolving a product line approach". ACM Press/Addison-Wesley Publishing Co., USA, 2000.

[12] Calero, C., Ruiz, J., Piattini, M.: Classifying web metrics using the Web Quality Model. Online Information Review, OIR - 29, 3 Emerald Literari. United Kingdom.

[13] CAFÉ Project website. http://www.esi.es/Projects/Cafe, 2003.

[14] Capilla, R. And Dueñas, J.C. Modeling Variability with Features in Distributed Architectures. In 4th International Workshop on Software Product-Family Engineering, pages 319-329, 2002.

[15] Clements, P., and Northrop, L., "Software Product Lines: Practices and Patterns", Addison Wesley, ISBN: 0-201-70332-7, 2001.

[16] J. Dehlinger and R.R. Lutz. "PLFaultCAT: A Product-Line Software Fault Tree Analysis Tool". In Automated Software Engineering, 2006, pp.169-193.

[17] C. Del Rosso. "The process and the lessons learned from performance tuning of a product family software architecture for mobile phones". In 8th European Conference on Software Maintenance and Reengineering, pp.270-275.

[18] T. J. Dolan. Architecture Assessment of Information-System Families: a practical perspective. PhD thesis, Tech. Univ. Eindhoven, Netherlands, 2001.

[19] Dromey, R. G., "Concerning the Chimera [software quality]", IEEE Software, no. 1, pp. 33-43, 1996.

[20] Dromey, R. G., "A model for software product quality", IEEE Transactions on Software Engineering, no. 2, pp. 146-163, 1995.

[21] ESAPS project website. http://www.esi.es/Projects/Esaps, 2001.

[22] Etxeberria, L. and Goiuria, S. Evaluation of Quality Attribute Variability in Software Product Families. In ICW on the Engineering of Computer Based Systems, 2008.

[23] L. Etxeberria, G. Sagardui and L. Belategi. "Quality aware Software Product Line Engineering" Journal of the Brazilian Computer Society (JBCS), vol.14, no.1, Mar 2008.

[24] L. Etxeberria and G. Sagardui. "Variability Driven Quality Evaluation in Software Product Lines". In Software Product Line Conference, 2008, pp.243-252.

[25] FAMILIES project website. http://www.esi.es/Projects/Families, 2005.

[26] N. Fenton and S. L. Pfleeger. Software Metrics: A Rigorous and Practical Approach. In 2nd Edition. International Thomson Computer Press, 1996.

[27] J. L. Fleiss. Statistical Methods for Rates and Proportions. Wiley, New York, USA, 1981.

[28] Gallina, B. and Guelfi, N. A Product Line Perspective for Quality Reuse of Development Framework for Distributed Transactional Applications. In Internacional Computer Software and Applications Conference, 2008.

[29] Ganesan, D., Knodel, J., Kolb, R., Haury, U., Meier, G.: Comparing Costs and Benefits of Different Test Strategies for a Software Product Line: A Study from Testo AG. In 11th International Software Product Line Conference, Kyoto, Japan, pp.74-83, September 2007.

[30] G.C. Gannod and R.R. Lutz. An Approach to Architectural Analysis of Product Lines. In 22nd International Conference on Software Engineering, 2000, pp.548-557.

[31] F. Garcia, F. Ruiz, C. Calero, M.F. Bertoa, A. Vallecillo, B. Mora, and M. Piattini. Effective Use of Ontologies in Software Measurement. The Knowledge Engineering Review, Vol. 00:0,1-24, 2008. Cambridge University Press.

[32] Gómez, O., Oktaba, H., Piattini, M., García, F.: A Systematic Review Measurement in Software Engineering: State-of-the-Art in Measures. In First International Conference on Software and Data Technologies (ICSOFT), Setúbal, Portugal, pp.11-14. September, 2006.

[33] González-Baixauli, B., Laguna, M.A. and Sampaio do Prado Leite, J.C. Using Goal-Models to Analyze Variability. In VaMoS, pages 101-107, 2007.

[34] Grady, R. B., Practical software metrics for project management and process improvement, Prentice Hall, 1992.

[35] A. Helferich, G. Herzwurm and S. Schockert. "QFD-PPP: Product Line Portfolio Planning using Quality Function Deployment". In Software Product Lines Conference, 2005, pp.162-173.

[36] J.S. Her, J.H. Kim, S.H. Oh, S.Y. Rhew and S.D. Kim. "A framework for evaluating reusability of core asset in product line engineering". Information and Software Technology, 2007, vol.49(7), pp.740-760.

[37] Z. Huang, R.R. Raje, A.M. Olson, M. Auguston, B.R. Bryant, C. Burt and C. Sun. "Unified Approach for System-Level Generative Programming". In 5th International Conference on Algorithms and Architectures for Parrallel Processing (ICA3PP), 2002, pp.136-142.

[38] IEEE Standard for a software quality metrics methodology, IEEE Std. 1061-1998, 1998.

[39] Inoki, M., Fukazawa, Y.: Software Product Line Evolution Method Based on Kaizen Approach. In 22nd Annual ACM Symposium on Applied Computing, Korea, 2007.

[40] ISO/IEC 25000:2005. Software Engineering. Software product Quality Requirements and Evaluation (SQuaRE).

[41] Jacobson, I., Booch, G., and Rumbaugh, J., The Unified Software Development Process, Addison Wesley Longman, Inc., 1999.

[42] M. Jaring and J. Bosch. "Architecting Product Diversification - Formalizing Variability Dependencies in Software Product Family Engineering". IN 4th International Conference on Quality Software, 2004, pp.154-161.

[43] Jarzabek, S, Yang, B. and Sam, S. Addressing Quality Attributes in Domain Analysis for Product Lines. IEE Proceedings Software, IEE and British Computer Society, vol. 153, No. 2, pages 61-73, 2006.

[44] 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.

[45] T. Kim, I.Y. Ko, S.W. Kang and D.H. Lee. "Extending ATAM to assess product line architecture". In 8th IEEE International Conference on Computer and Information Technology, pp. 790-797, 2008.

[46] B. Kitchenham. "Guidelines for Performing Systematic Literature Reviews in Software Engineering". Version 2.3, EBSE Technical Report, Keele University, UK.

[47] B. Kitchenham. "Procedures for Performing Systematic Reviews". Joint Technical Report Software Engineering Group, Keele University, UK and Empirical Software Engineering, National ICT Australia Ltd, Australia, 2004.

[48] Kitchenham, B. and Pfleeger, S. L., "Software quality: the elusive target [special issues section]", IEEE Software, no. 1, pp. 12-21, 1996.

[49] M. Khurum, T. Gorschek and K. Petterson. "Systematic Review of Solutions Proposed for Product Line Economics". In 2nd International Workshop on Management and Economics of Software Product Lines, Limerick, Ireland, 2008, pp.386-393.

[50] R. Kolb, I. John, J. Knodel, D. Muthing, U. Haury and G. Meier. "Experiences with Product Line Development of Embedded Systems at Testo AG". In 10th International Software Product Line Conference, 2006.

[51] Kuusela, J. And Savolainen, J. Requirements engineering for product lines. In 22nd International Conference on Software Engineering (ICSE), pages 61-69, 2000.

[52] McCall, J. A., Richards, P. K., and Walters, G. F., "Factors in Software Quality", Nat'l Tech.Information Service, no. Vol. 1, 2 and 3, 1977.

[53] M. Matinlassi, E. Niemelä, and L. Dobrica. Quality-driven architecture design and quality analysis method: A revolutionary initiation approach to a product line architecture. Technical Report VTT-PUBS-456, VTT Electronics, 2002.

[54] D. Mellado, E. Fernandez-Medina and M. Piatiini. "Security Requirements Variability for Software Product Lines". In 3rd International Conference on Availability, Reliability and Security (ARES), 2008, pp.1413-1420.

[55] D. Needham and S. Jones. "A Software Fault Tree Metric". 2006. Journal of Systems and Software, 2007, vol.80(9), pp.1530-1540.

[56] Niemelä, E. And Immonen, A. Capturing quality requirements of product family architecture. Information and Software Technology, vol. 49, pages 1107-1120, 2007.

[57] F. G. Olumofin and V. B. Mišić. "A holistic architecture assessment method for software product lines". Information and Software Technology 49, 2007, pp. 309-323.

[58] O. Osatretin Edwin. Testing in Software Product Lines. School of Engineering, Master Thesis, Blekinge Institute of Technology, Sweden. March 2007.

[59] D. Parnas. On the design and development of program families. IEEE Transactions (ICSSEA 2000), volume 3, 2001.

[60] K. Pohl, G. Böckle, and F. van der Linden. Software Product Line Engineering: Foundations, Principles, and Techniques. Springer, 2005.

[61] G. Raghavan. "Improving software quality in product families through systematic reengineering". ECSQ 2002.

[62] Rational Software Inc., RUP - Rational Unified Process, www.rational.com, 2003.

[63] Sinnema, M., Deelstra, S., Nijhuis, J. and Bosch, J. COVAMOF: A Framework for Modeling Variability in Software Product Families. In 3rd International Conference on Software Product Lines (SPLC), pages 197-213, 2004.

[64] M. Sinnema, J. S. van der Ven and S. Deelstra. "Using Variability Modeling Principles to Capture Architectural Knowledge". In ACM SIGSOFT Software Engineering Notes, 31(5), 2006.

[65] E. D. de Souza Filho, R. de Oliveira Cavalcanti, D. F. S. Neiva, T.H.B. Oliveira, L. Barachisio Lisboa, E.S. de Almeida and S. R. de Lemos Meira. "Evaluating Domain Design Approaches Using Systematic Review". In 2nd European Conference on Software Architecture, Cyprus, 2008, pp.50-65.

[66] S. Thiel. On the definition of a framework for an architecting process supporting product family development. In PFE '01: Revised Papers from the 4th International Workshop on Software Product-Family Engineering, pages 125–142, London, UK, 2002. Springer-Verlag.

[67] M. Tom, W. Alan W. A Product-Focused Approach to Software Certification. Computer 41(2):91-93, 2008.

[68] P. Trinidad, D. Benavides, A. Durán, A. Ruiz-Cortés and M. Toro. "Automated error analysis for the agilization of feature modeling". Jouranl of Systems and Software, 2008, vol.81(6), pp.883-896.

[69] A. Van der Hoek, E. Dincel and N. Medvidovic. "Using service utilization metrics to assess the structure of product software architectures". In 9th International Symposium on Software Metrics, 2003, pp.298.

[70] F. Van der Linden and J. Müller. Creating architectures with building blocks. IEEE Software, 12(6):51-60, 1995.

[71] F. Van der Linden, K. Schmid and E. Rommes. Software Product Lines in Action. Springer, 2007.

[72] Zhang, H., Jarzabek, S. And Yang, B. Quality Prediction and Assessment for Product Lines. In 15th International Conference on Advanced Information Systems Engineering (CAiSE), pages 681-695, 2003.

Anexos

A. Artículos clasificados en la Revisión Sistemática de Métodos y Técnicas para evaluar la calidad de Líneas de Producto Software

1. J. Bayer. "Design for Quality". PFE 2003.
2. D. Benavides, A. Ruiz-Cortés, M. A. Serrano, C. Montes de Oca Vázquez. "A first approach to build product lines of multi-organizational Web based systems (MOWS)". I2CS 2004.
3. S. Bhattacharya, D. E. Perry. "Predicting Emergent Properties of Component Based Systems". ICCBS 2007.
4. B. Boehm, A. Winsor Brown, R. Madachy and Y. Yang. "A Software Product Line Life Cycle Cost Estimation Model". ISESE 2004.
5. R. Capilla, J.C. Dueñas. "Modeling variabilty with features in distributed architectures". PFE 2002.
6. C. Catal, B. Duni. "A Conceptual Framework to Integrate Fault Prediction Sub-process for Software Product Lines". TASE 2008.
7. J. Dehlinger, R.R. Lutz. "PLFaultCAT: A Product-Line Software Fault Tree Analysis Tool". ASE 2006.
8. C. Del Rosso. "The process and the lessons learned from performance tuning of a product family software architecture for mobile phones". CSMR 2004.
9. E. Dincel, N. Medvidovic, A. van der Hoek. "Measuring product line architectures". PFE 2001.
10. L. Etxeberria, G. Sagardui. "Variability Driven Quality Evaluation in Software Product Lines". SPLC 2008.
11. L. Etxeberria, G. Sagardiu. "Evaluation of Quality Attribute Variability in Software Product Families". IEEE ECBS 2008.
12. B. Gallina, N. Guelfi. "A Product Line Perspective for Quality Reuse of Development Framework for Distributed Transactional Applications". QUORS 2008.
13. D. Ganesan, J. Knodel, R. Kolb, U. Haury, G. Meier. "Comparing Costs and Benefits of Different Test Strategies for a Software Product Line: A Study from Testo AG". SPLC 2007.
14. G.C. Gannod, R.R. Lutz. An Approach to Architectural Analysis of Product Lines. ICSE 2000.
15. B. Geppert, D. M. Weiss. "Goal-Oriented Assessment of Product-Line Domains". METRICS 2003.
16. A. Helferich, G. Herzwurm, S. Schockert. "QFD-PPP: Product Line Portfolio Planning using Quality Function Deployment". SPLC 2005.
17. J.S. Her, J.H. Kim, S.H. Oh, S.Y. Rhew, S.D. Kim. "A framework for evaluating reusability of core asset in product line engineering". IST 2007.
18. Z. Huang, R.R. Raje, A.M. Olson, M. Auguston, B.R. Bryant, C. Burt, C. Sun. "Unified Approach for System-Level Generative Programming". ICA3PP 2002.
19. M. Jaring, J. Bosch. "Architecting Product Diversification - Formalizing Variability Dependencies in Software Product Family Engineering". QSIC 2004.
20. T. Kim, I.Y. Ko, S.W. Kang, D.H. Lee. "Extending ATAM to assess product line architecture". IEEE CIT 2008.

21. T. Kishi, N. Noda, T. Katayama. "Architectural Design for Evolution by Analyzing Requirements on Quality Attributes". APSEC 2001.
22. J. Knodel, I. John, D. Ganesan, M. Pinzger, F. Usero, J. L. Arciniegas and C. Riva. "Asset Recovery and Their Incorporation into Product Lines". WCRE 2005.
23. R. Kolb, I. John, J. Knodel, D. Muthing, U. Haury, G. Meier. "Experiences with Product Line Development of Embedded Systems at Testo AG". SPLC 2006.
24. H. Krüger, R. Mathew, M. Meisinger. "From scenarios to aspects: exploring product lines". ACM SIGSOFT 2005.
25. F. Loesch, E. Ploedereder. "Optimization of Variability in Software Product Lines". SPLC 2007.
26. R. R. Lutz, G.C. Gannod. "Analysis of a software product line architecture: an experience report". JSS 2003.
27. M. Matinlassi, E. Niemelä, L. Dobrica. "Quality-driven architecture design and quality analysis method: A revolutionary initiation approach to a product line architecture". TR VTT 2002.
28. Mellado, E. Fernandez-Medina, M. Piatiini. "Security Requirements Variability for Software Product Lines". ARES 2008.
29. V. B. Mišic. "Measuring the Coherence of Software Product Line Architectures". SERP 2003.
30. D. Needham, S. Jones. "A Software Fault Tree Metric". JSS 2006.
31. E. Niemelä, A. Immonen. "Capturing quality requirements of product family architecture". IST 2007.
32. F. G. Olumofin, V. B. Mišic. "A holistic architecture assessment method for software product lines". IST 2007.
33. K. Pohl, A. Reuys. "Considering variabilities during component selection in product family development". PFE 2001.
34. G. Raghavan. "Improving software quality in product families through systematic reengineering". ECSQ 2002
35. M. Sinnema, J. S. van der Ven, S. Deelstra. "Using Variability Modeling Principles to Capture Architectural Knowledge". ACM SIGSOFT 2006.
36. J. Savolainen, J. Kuusela. "Consistency management of product line requirements". RE 2001.
37. P. Trinidad, D. Benavides, A. Durán, A. Ruiz-Cortés, M. Toro. "Automated error analysis for the agilization of feature modeling". JSS 2008.
38. A. Van der Hoek, E. Dincel, N. Medvidovic. "Using service utilization metrics to assess the structure of product software architectures". METRICS 2003.
39. H. Zhang, S. Jarzabek, B. Yang: "Quality Prediction and Assessment for Product Lines". CAiSE 2003.

B. Resultados y Artículos seleccionados en la revisión sistemática de métricas y atributos de calidad.

A continuación se muestran los resultados de la revisión sistemática y los artículos seleccionados.

Métrica	Característica Calidad	Fase Ciclo Vida	Artefacto Software Evaluado				Valid.		Herr.	Ref.
			Arq. LP	Activ.	Arq. Pr.	Prod.	T	E		
Coherence	Funcionalidad	ID Diseño	x							9
A Software Fault Tree Metric	Funcionalidad	IA Diseño			x					10
Level of Coverage	Funcionalidad	ID Diseño	x	x						7
Level of Coverage	Funcionalidad	ID Diseño	x	x						7
Error Propagation	Fiabilidad	ID Diseño	x					x	x	1
Maturity	Fiabilidad	ID Pruebas		x						11
Overall Understandability (OU)	Usabilidad	ID Diseño	x	x			x			14
Interface Complexity	Usabilidad	ID Dis. ó Rea.		x						11
Provided Service Utilization (PSU)	Eficiencia	ID Diseño		x						13
Required Service Utilization (RSU)	Eficiencia	ID Diseño		x						13
Compound PSU (CPSU)	Eficiencia	ID Diseño	x							13
Compound RSU (CRSU)	Eficiencia	ID Diseño	x							13
Percentatge of product-specific components per product	Eficiencia	ID Diseño			x				x	6
Percentatge of non-generic reusable components per product	Eficiencia	ID Diseño			x				x	6
Percentatge of slightly generic reusable components per product	Eficiencia	ID Diseño			x				x	6
Percentatge of very generic reusable components per product	Eficiencia	ID Diseño			x				x	6
Average RSU	Eficiencia	ID Diseño		x						5
Average per Product Family Architecture	Eficiencia	ID Diseño	x							5
Tamaño binario	Eficiencia	IA Pruebas		x		x			x	12
Performance	Eficiencia	IA Pruebas				x			x	12
Functional Coverage (FC)	Funcionalidad	ID Diseño	x				x			14
Architectural Commonality (AC)	Eficiencia	ID Diseño	x				x			14
Non-functional Coverage (NC)	Funcionalidad	ID Diseño	x				x			14
Coverage of Variability (CV)	Eficiencia	ID Diseño	x	x			x			14
Cumulative Applicability (CA)	Eficiencia	ID Diseño	x	x			x			14
Reusability (RE)	Eficiencia	ID Diseño	x	x			x			14
CompVariant	Eficiencia	ID Diseño			x					3
CompVP	Eficiencia	ID Diseño			x					3
CompVariability	Eficiencia	ID Diseño			x					3
CompPL	Eficiencia	ID Diseño			x					3
Efficiency	Eficiencia	Evolución	x			x				2
Component Reuse Rate	Eficiencia	IA Diseño			x				x	16

Métrica	Característica Calidad	Fase Ciclo Vida	Artefacto Software Evaluado				Valid.		Herr.	Ref.
			Arq. LP	Activ.	Arq. Pr.	Prod.	T	E		
(CRR)										
Reuse Benefit Rate (RBR)	Eficiencia	IA Realización				x			x	16
Clone Coverage	Eficiencia	ID Dis. ó Rea.		x						15
Reusability	Eficiencia	ID Diseño	x							11
Modularity	Portabilidad	ID Diseño	x							11
Average PSU	Mantenibilidad	ID Diseño		x						5
Cost of testing a single product	Mantenibilidad	IA Diseño		x					x	6
Relative cost to test a non-generic component	Mantenibilidad	IA Diseño		x					x	6
Relative cost to test a slightly generic component	Mantenibilidad	IA Diseño		x					x	6
Relative cost to test a very generic component	Mantenibilidad	IA Diseño		x					x	6
Relative cost to test adaptations to a non-generic component	Mantenibilidad	IA Diseño		x					x	6
Relative cost to test adaptations to a slightly generic component	Mantenibilidad	IA Diseño		x					x	6
Relative cost to test adaptations to a very generic component	Mantenibilidad	IA Diseño		x					x	6
Complejidad ciclomática	Mantenibilidad	IA Pruebas		x		x			x	12
Component Compliance (CC)	Mantenibilidad	ID Diseño		x			x			14
Size of code in the product line	Mantenibilidad	Evolución	x							2
Number of modules	Mantenibilidad	Evolución	x							2
Code churn (changes in PL layer)	Mantenibilidad	Evolución	x							2
Source of change	Mantenibilidad	Evolución	x	x		x				2
Product line growth	Mantenibilidad	Evolución	x							2
Impact of change	Mantenibilidad	Evolución	x							2
Adjust product line growth	Mantenibilidad	Evolución	x	x						2
Code churn (for product layer)	Mantenibilidad	Evolución				x				2
Changes on product line	Mantenibilidad	Evolución	x			x				2
Code churn on product line	Mantenibilidad	Evolución	x			x				2
UseCaseVP	Mantenibilidad	ID Diseño	x							3
UseCaseAlternativeOR	Mantenibilidad	ID Diseño	x							3
UseCaseAlternativeXOR	Mantenibilidad	ID Diseño	x							3
UseCaseOptional	Mantenibilidad	ID Diseño	x							3
UseCaseMandatory	Mantenibilidad	ID Diseño	x							3
UseCaseNumVariantsAltOR	Mantenibilidad	ID Diseño	x							3
UseCaseNumVariantsAltXOR	Mantenibilidad	ID Diseño	x							3
UseCaseNumVariantsOptional	Mantenibilidad	ID Diseño	x							3
UseCaseNumVariantsMandatory	Mantenibilidad	ID Diseño	x							3
ClassVP	Mantenibilidad	ID Diseño	x							3
ClassAlternativeOR	Mantenibilidad	ID Diseño	x							3
ClassAlternativeXOR	Mantenibilidad	ID Diseño	x							3
ClassOptional	Mantenibilidad	ID Diseño	x							3

Métrica	Característica Calidad	Fase Ciclo Vida	Artefacto Software Evaluado				Valid.		Herr.	Ref.
			Arq. LP	Activ.	Arq. Pr.	Prod.	T	E		
ClassMandatory	Mantenibilidad	ID Diseño	x							3
ClassNumVariantsAltOR	Mantenibilidad	ID Diseño	x							3
ClassNumVariantsAltXOR	Mantenibilidad	ID Diseño	x							3
ClassNumVariantsOptional	Mantenibilidad	ID Diseño	x							3
ClassNumVariantsMandatory	Mantenibilidad	ID Diseño	x							3
UseCaseTotalVP	Mantenibilidad	ID Diseño	x							3
UseCaseTotalAlternativeOR	Mantenibilidad	ID Diseño	x							3
UseCaseTotalAlternativeXOR	Mantenibilidad	ID Diseño	x							3
UseCaseTotalOptional	Mantenibilidad	ID Diseño	x							3
UseCaseTotalMandatory	Mantenibilidad	ID Diseño	x							3
UseCaseTotalVariabilities	Mantenibilidad	ID Diseño	x							3
ClassTotalVP	Mantenibilidad	ID Diseño	x							3
ClassTotalAlternativeOR	Mantenibilidad	ID Diseño	x							3
ClassTotalAlternativeXOR	Mantenibilidad	ID Diseño	x							3
ClassTotalOptional	Mantenibilidad	ID Diseño	x							3
ClassTotalMandatory	Mantenibilidad	ID Diseño	x							3
ClassTotalVariabilities	Mantenibilidad	ID Diseño	x							3
ComponentTotalVariabilities	Mantenibilidad	ID Diseño	x							3
UseCaseTotalPLVariabilities	Mantenibilidad	ID Diseño	x							3
ClassTotal PLVariabilities	Mantenibilidad	ID Diseño	x							3
PLTotalVariability	Mantenibilidad	ID Diseño	x							3
ComponentVariable	Mantenibilidad	ID Diseño	x							3
Structure Similarity Coefficient (SSC)	Mantenibilidad	ID Diseño	x						x	16
Architecture Variability (AV)	Mantenibilidad	ID Diseño	x						x	16
PLA-IFG Cyclomatic Complexity	Mantenibilidad	ID Diseño	x						x	16
Maintainability Index (MI)	Mantenibilidad	IA Pruebas				x			x	4
Tracking Degradation	Mantenibilidad	IA Pruebas	x	x	x	x				8
Observability	Mantenibilidad	ID Dis. ó Rea.		x						11
Customizability	Mantenibilidad	ID Dis. ó Rea.		x						11
Effectiveness of Tailoring (ET)	Portabilidad	ID Diseño		x			x			14
Tailorability of Closed variability (TC)	Portabilidad	ID Diseño		x			x			14
Tailorability of Open variability (TO)	Portabilidad	ID Diseño		x			x			14
Tailorability (TL)	Portabilidad	ID Diseño		x			x			14
Self Completeness	Portabilidad	ID Dis. ó Rea.		x						11
Size of product code	Mantenibilidad	Evolución				x				2

Articulos seleccionados

1. Abdelmoez, W., Nassar, D.M., Shereschevsky, M., Gradetsky, N., Gunnalan, R., Ammar, H.H., Yu, B., Mili, A.: Error Propagation In Software Architectures. In 10th International Symposium on Software Metrics (METRICS), Chicago, Illinois, USA, 2004.
2. Ajila, S. A., Dumitrescu, R. T.: Experimental use of code delta, code churn, and rate of change to understand software product line evolution. Journal of Systems and Software 80, pp.74-91, 2007.
3. Alves de Oliveira Junior, E., Gimenes, I. M. S., Maldonado, J. C.: A Metric Suite to Support Software Product Line Architecture Evaluation. In XXXIV Conferencia Latinamericana de Informática (CLEI), Santa Fé, Argentina, pp.489-498, 2008.
4. Aldekoa, G., Trujillo, S., Sagardui, G., Díaz, O.: Quantifying Maintanibility in Feature Oriented Product Lines, Athens, Greece, pp.243-247, 2008.
5. Dincel, E., Medvidovic, N., Van der Hoek, A.: Measuring Product Line Architectures. In 4th International Workshop on Product Family Engineering (PFE), Bilbao, Spain, 2001.
6. Ganesan, D., Knodel, J., Kolb, R., Haury, U., Meier, G.: Comparing Costs and Benefits of Different Test Strategies for a Software Product Line: A Study from Testo AG. In 11th International Software Product Line Conference, Kyoto, Japan, pp.74-83, September 2007.
7. Inoki, M., Fukazawa, Y.: Software Product Line Evolution Method Based on Kaizen Approach. In 22nd Annual ACM Symposium on Applied Computing, Korea, 2007.
8. Johansson, E., Höst, R.: Tracking Degradation in Software Product Lines through Measurement of Design Rule Violations. In 14th International Conference on Software Engineering and Knowledge Engineering, Ischia, Italy, pp.249-254, 2002.
9. Mišić, V. B.: Measuring the Coherence of Software Product Line Architectures. Technical Report TR 06/03. University of Manitova, Canada, June 2006.
10. Needham, D., Jones, S.: A Software Fault Tree Metric. In 22nd International Conference on Software Maintenance (ICSM), Philadelphia, Pennsylvania, USA, 2006.
11. Rahman, A.: Metrics for the Structural Assessment of Product Line Architecture. Master Tesis on Software Engineering, Thesis no. MSE-2004:24. School of Engineering, Blekinge Institute of Technology, Sweden, 2004.
12. Siegmund, N., Rosenmüller, M., Kuhlemann, M., Kästner, C., Saake, G.: Measuring Non-functional Properties in Software Product Lines for Product Derivation. In 15th Asia-Pacific Software Engineering Conference, Beijing, China, 2008.
13. Van der Hoek, A., Dincel, E., Medidović, N.: Using Services Utilization Metrics to Assess the Structure of Product Line Architectures. In 9th International Software Metrics Symposium (METRICS), Sydney, Australia, 2003.
14. Sun Her, J., Hyeok Kim, J., Hun Oh, S., Yul Rhew, S., Dong Kim, S.: A framework for evaluationg reusability of core asset in product line engineering. Information and Software Technology 49, pp. 740-760, 2007.
15. Yoshimura, K., Ganesan, D., Muthig, D.: Assessing Merge Potencial of Existing Engine Control Systems into a Product Line. In International Workshop on Software Engineering for Automative Systems, Shangai, China, pp.61-67, 2006.
16. Zhang, T., Deng, L., Wu, J., Zhou, Q., Ma, C.: Some Metrics for Accesing Quality of Product Line Architecture. In International Conference on Computer Science and Software Engineering (CSSE), Wuhan, China, pp.500-503, 2008.

C. Relaciones entre los atributos del Modelo de Calidad y métricas para su evaluación

1. Funcionalidad

Atributo	Métrica	Descripción
Completitud de la implementación funcional	Functional implementation completeness	Funciones implementadas que son apropiadas para las tareas especificadas.
Correctitud funcional	Functional implementation coverage	Funciones implementadas correctamente que son apropiadas para las tareas específicas.
Precisión computacional	Computational accuracy	Contar el número de funciones que han implementado los requisitos precisos contra el número de funciones con requisitos precisos especificados.
Nivel de cubrimiento de los activos	Número de activos especificados en la LPS.	Contar el número de activos que están especificados en la LPS frente al número total de activos necesarios.
	Número de activos implementados en la LPS.	Contar el número de activos que están implementados en la LPS frente al número total de activos necesarios.

2. Fiabilidad

Atributo	Métrica	Descripción
Ratio de llamadas a funciones satisfactorias.	Ratio de llamadas a funciones satisfactorias de un activo.	Establecer un número N determinado de llamadas a cada función. Realizar N llamadas a cada función. Obtener el número porcentaje de llamadas satisfechas correctamente frente al número de llamadas totales.
	Ratio de llamadas a funciones satisfactorias de un producto.	Calcular el Ratio de llamadas a funciones satisfactorias de cada activo del producto y obtener la media.
	Ratio de llamadas a funciones satisfactorias de la familia de productos.	Calcular el Ratio de llamadas a funciones satisfactorias de cada activo de la familia de productos y obtener la media.
Failure avoidance	Ratio de fallos evitados en un activo.	Contar el número de reglas de fallos evitadas de un activo y compararlo con el número de reglas de fallos a ser considerados de ese activo.
	Ratio de fallos evitados en un producto.	Calcular el Ratio de fallos evitados en un activo para todos los activos del producto y obtener la media.
	Ratio de fallos evitados	Calcular el Ratio de fallos evitados en

Atributo	Métrica	Descripción
	en la familia de productos.	un activo para todos los activos de la familia de productos y obtener la media.
Restorability	Restauración de un activo.	Contar el número de requisitos de restauración implementados y compararlo con el número de requisitos de restauración especificados en los requisitos. (Ejemplos de requisitos de restauración: puntos de control en la base de datos, puntos de control en las transacciones, funciones de deshacer, funciones de rehacer).
	Restauración de un producto.	Calcular la Restauración de un activo, para cada activo que compone el producto y calcular la media.
	Restauración de la familia de productos.	Calcular la Restauración de un activo, para cada activo que compone la familia de productos y calcular la media.

3. Eficiencia

Atributo	Métrica	Descripción
Response time.	Tiempo de respuesta de una función de un activo.	Calcula el tiempo de respuesta al realizar llamadas a una función de un activo.
	Porcentaje de tiempo de respuesta de un activo.	Calcula la media de los tiempos de respuesta de todas las funciones de un activo.
Throughput time.	Throughput time de un activo.	Calcula el número de tareas realizadas por un activo por unidad de tiempo.
	Throughput time de un producto.	Calcula el número de tareas realizadas por un producto por unidad de tiempo.
Turnaround time.	Turnaround time.	Calcula el tiempo empleado en realizar un conjunto de tareas.
I/O Utilization	I/O Utilization del producto.	Estima o calcula la utilización de I/O para la aplicación.
I/O Utilization Message Density	I/O Utilization Message Density del producto.	Cuenta el número de errores pertenecientes a los fallos de I/O y mensajes de peligro (warnings) y los compara con el número estimado de líneas de código responsables en el sistema de llamadas.
Memory utilization.	Memory utilization.	Estimación de la memoria requerida.
Memory utilization message density	Memory utilization message density	Contar el número de mensajes de error pertenecientes a los fallos de memoria y mensajes de error (warnings) y compararlo con el número estimado de

Atributo	Métrica	Descripción
		líneas de código responsables en el sistema de llamadas.
Transmission Utilization	Transmission Utilization	Estima los requisitos de utilización de fuentes de transmisión contra la estimación de volúmenes de transmisión.

4. Usabilidad

Atributo	Métrica	Descripción
Completitud de la descripción del activo	Completitud de la descripción del activo	Mide como de fácil, eficiente y correcta la descripción de un activo puede ser comprendida por los usuarios, donde la descripción de un activo incluya la especificación, manual de usuario y cualquier documentación describiendo el activo.
Capacidad de demostración	Capacidad de demostración	Mide cuantas funciones de un activo o producto pueden ser explicadas al usuario con una demostración.
Uniformidad entre las interfaces de usuario de la LPS	Cantidad de interfaces de usuario similares	Cuenta el número de interfaces que son iguales para cada diseño de interfaz.
	Tipos de diseño de interfaces de usuario	Cuenta el número de tipos de diseños de interfaces que son distintos.
Funciones evidentes para un usuario	Ratio de funciones evidentes para un usuario en un activo o producto.	Cuenta el número de funciones en un activo o un producto que no requieren de explicación adicional para el entendimiento de un usuario.
Facilidad de ayuda	Existencia de activos concretos para la facilidad de ayuda	Cuenta el número de activos que han sido diseñados/implementados para funciones concretas de facilidad de ayuda.
	Aseguramiento que se incluye en los productos el activo de facilidad de ayuda	Mide el grado en el que se asegura que un producto final contendrá o contiene los activos específicos de

Atributo	Métrica	Descripción
		facilidad de ayuda.

5. Seguridad

Atributo	Métrica	Descripción
Seguridad de las contraseñas.	Grado de seguridad mínimo impuesto para una contraseña	Seguridad mínima que el sistema impone al usuario para crear una contraseña.
	Grado de seguridad mínimo de una contraseña creada automáticamente.	Seguridad mínima que utiliza el sistema para crear una contraseña generada aleatoriamente.
	Ratio de la seguridad de las contraseñas.	Ratio de las seguridades impuestas por el sistema para las contraseñas.
Encriptación de datos.	Ratio de encriptación de los datos.	Ratio de la cantidad de datos encriptados contra el número total de datos.
Alcanzabilidad de información privada sin permisos.	Alcanzabilidad de información privada sin permisos.	Asegurar la no existencia de un acceso a información privada por un usuario no autorizado.
Encriptación de los activos.	Ratio de encriptación de un activo.	Calcula la cantidad de información encriptada con respecto al total de información de un activo.
	Ratio de encriptación de los activos de un producto.	Calcula el ratio de la encriptación de un activo de cada uno de los activos que componen un producto.
	Ratio de encriptación de la familia de productos.	Calcula el ratio de la encriptación de un activo de cada uno de los activos que forman la familia de productos.
Historial de acciones.	Existencia de un historial de acciones en un activo.	Comprueba si se ha tenido en cuenta la creación de un historial de las acciones en el activo.
	Ratio de existencias de un historial de acciones en un producto.	Calcula la cantidad de activos en el producto que han tenido en cuenta la creación de un historial de acciones con respecto al número de activos que forman el producto final.
	Ratio de existencias de un historial de acciones en la familia de productos.	Calcula la cantidad de activos de la familia de productos que han tenido en cuenta la creación de un historial de acciones con respecto al número de activos que forma la familia de productos.
Acceso al historial de acciones.	Acceso al historial de acciones.	Asegura que no existe ninguna función, transición o estado que permita que un usuario pueda acceder al historial de acciones.
Encriptación de la	Encriptación de la	Calcula la cantidad de información del

Atributo	Métrica	Descripción
información del historial	información del historial.	historial encriptada con respecto a la cantidad total de la información del historial.
Modificación externa de un activo.	Modificación de un activo por parte de otro activo.	Asegura que no existe ninguna función en el sistema que pueda modificar el comportamiento de un activo.
	Modificación de un activo por parte de un usuario.	Asegura que no existe ninguna función, transición o estado que permita a un usuario modificar el comportamiento de un activo.
Información personal requerida.	Cantidad de información requerida para la autenticación.	Suma la cantidad de información requerida para la autenticación.
	Calidad de la información requerida para la autenticación.	Evalúa si el sistema comprueba que la información introducida es correcta.
Finalización de sesión.	Finalización de sesión de un usuario.	Evalúa si el sistema comprueba si la sesión es cerrada cuando el usuario finaliza su uso.

6. Compatibilidad

Atributo	Métrica	Descripción
Functional inclusiveness	Functional inclusiveness	Contar el número de funciones cubiertas por nuevo software que produce resultados similares y compararlo con el número de funciones del software antiguo.
Available co-existence	Available co-existence	Contar el número de entidades con las cuales el producto puede co-exisitir cas está especificado y comparar con el número de entidades en el entorno de producto que requieren co-existencia.
Formato de datos	Data exchangeability	Cuenta el número de formatos de interfaces que han sido implementados correctamente como en la especificación y lo compara con el número de formato de datos que han sido intercambiados como en la especificación.

7. Mantenibilidad

Atributo	Métrica	Descripción
Component replaceability	Component Compliance	Mide el grado en el cual los componentes en un activo pueden ser reemplazados con nuevas

Atributo	Métrica	Descripción
		actualizaciones sin complicaciones.
Functional commonality	Functional Coverage	Mide la media de la comunalidad de cada característica funcional en una activo. La comunilidad de cada característica funcional puede ser mediad por el cálculo del grado de miembros utilizando cada característica funcional.
Non-functional commonality	Architectural Commonality	Mide la comunalidad de los requisitos no funcionales que son mantenidos por la arquitectura de la LP. Mide el ratio de aplicaciones que comparten la arquitectura de la LP de un activo.
	Non-functional Coverage	Mide la comunalidad de los otros requisitos no funcionales que no pueden ser direccionados por la arquitectura de la LP. Mide la media de las comunalidades de cada característica no funcional que no está cubierta por la arquitectura de la LP.
	Non-functional Commonality	Mide la comunalidad de los requisitos no funcionales (tanto los mantenidos por la LP como los que no)
Variability richness	Coverage of Variability	Mide cuantos puntos de variación incluidos en el alcance de la LP son realmente implementados en el activo.
Applicability	Cumulative Applicability	Mide como de aplicable es un activo para desarrollar varias aplicaciones.
Tailorability	Effectiveness of Tailoring	Mide cuantos puntos de variación pueden ser eficientemente y correctamente resueltos, esto es, como de bien los puntos de variación son resueltos.
	Tailorability of Closed variability	Mide el grado en el cual los puntos de variación cerrados en DRM (Decision Resolution Model) pueden ser válidamente resueltos sin efectos adversos o fallos.
	Tailorability of Open variability	Mide el grado en el cual los puntos de variación abiertos en DRM pueden ser válidamente resueltos sin efectos adversos o fallos.
	Tailorability	Mide la capacidad de que el activo sea posiblemente hecho a medida adheriendo los requisitos específicos de la aplicación.

8. Transferabilidad

Atributo	Métrica	Descripción
Adaptability of data structures	Adaptabilidad de las estructuras de información de un activo.	Contar el número de estructuras de información del activo que son operables y que no están limitadas después de la adaptación y compararlo con el número de total de estructuras de información requeridas para la capacidad de adaptación.
	Ratio de la adaptabilidad de las estructuras de información de un producto.	Ratio de la Adaptabilidad de las estructuras de información de los activos de un producto.
	Ratio de la adaptabilidad de las estructuras de información de la familia de productos.	Ratio de la Adaptabilidad de las estructuras de información de los activos de la familia de productos.
Organizational Environment daptability	Organizational Environment daptability	Contar el número de funciones implementadas las cuales son capaces de lograr los resultados requeridos en múltiples organizaciones y entornos de negocio y compararlo con el número de funciones que requieren tener la capacidad de adaptación al entorno organizacional.
Hardware Environment Adaptability	Hardware Environment Adaptability	Contar el número de funciones implementadas las cuales son capaces de lograr los resultados requeridos en múltiples entornos hardware especificados y compararlo con el número de funciones que requieren tener la capacidad de adaptación a entornos hardware.
System software Environmental adaptability (OS, concurrent application)	System software Environmental adaptability	Contar el número de funciones implementadas que son capaces de lograr los resultados requeridos en múltiples sistemas software especificados y compararlos con el número de funciones que requieren tener la capacidad de adaptación a entornos de sistemas software.
Ease of setup retry	Ease of setup retry	Contar el número de operaciones de reinstalación implementadas y compararlo con el número de operaciones de reinstalación requeridas.
Installation effort	Installation effort	Contar el número de pasos de la instalación automatizados que estén implementados y compararlo con el número de pasos de instalación

Atributo	Métrica	Descripción
		establecidos.
Installation flexibility	Capacidad de la instalación para ser flexible y personalizable.	Contar el número de operaciones de instalación personalizables implementadas y compararlo con el número de operaciones de instalación con la capacidad de personalización requeridas.

D. Operacionalización de las métricas

1. Funcionalidad

Métrica	Artefacto(s)	Fase(s)	Descripción
Completitud de la implementación funcional	Evalúa activos utilizando modelos de requisitos.	Realización en dominio y aplicación.	A= Número de funciones no implementadas. B= Número de funciones descritas en la especificación de requisitos. M= 1-(A/B)
Cubrimiento de la implementación funcional	Evalúa activos utilizando modelos de requisitos.	Realización en dominio y aplicación.	A= Número de funciones no implementadas o implementadas incorrectamente. B= Número de funciones descritas en la especificación de requisitos. M= 1-(A/B)
Precisión computacional	Activos	Implementación dominio.	X = A/B A = Número de funciones en las cuales los requisitos precisos especificados han sido implementados, as confirmed in evaluation. B = Número de funciones para las cuales los requisitos de precisión especificados debían ser implementados.
Número de activos especificados en la LPS	Arquitectura de la LPS. Activos.	Diseño del dominio	X = A/B A = Número de activos que están especificados en la LPS. B = Número total de activos necesarios.
Número de activos implementados en la LPS	Arquitectura de la LPS. Activos.	Realización del dominio	X = A/B A = Número de activos que están implementados en la LPS. B = Número total de activos necesarios.

2. Fiabilidad

Métrica	Artefacto(s)	Fase(s)	Descripción
Ratio de llamadas a funciones satisfactorias de un activo.	Activo	Realización dominio, aplicación (si es creado en fase de aplicación)	X = A/(F*N) N = Número de llamadas que se van a realizar a cada función. F = Número de funciones del activo. A = Número de llamadas satisfechas

Métrica	Artefacto(s)	Fase(s)	Descripción
			correctamente.
Ratio de llamadas a funciones satisfactorias de un producto.	Producto, activos	Realización aplicación	X = Sumatorio de R de los activos del producto / N R = Ratio de llamadas a funciones satisfactorias de un activo. N = Número de activos del producto.
Ratio de llamadas a funciones satisfactorias de la familia de productos.	Familia de productos, activos	Realización dominio.	X = Sumatorio de R de los activos de la familia de productos / N R = Ratio de llamadas a funciones satisfactorias de un activo. N = Número de activos de la familia de productos.
Ratio de fallos evitados en un activo.	Activo.	Diseño, Realización dominio (o aplicación si es una característica propia).	X = A/B A = Número de reglas de fallos que son eludidas en diseño/código. B = Número de reglas de fallos a ser consideradas.
Ratio de fallos evitados en un producto.	Producto final, activos.	Diseño, realización aplicación.	X = Sumatorio de R de los activos del producto / N R = Ratio de fallos evitados en un activo N = Número de activos del producto.
Ratio de fallos evitados en la familia de productos.	Familia de productos, activos.	Diseño, realización dominio.	X = Sumatorio de R de los activos de la familia de productos / N R = Ratio de fallos evitados en un activo. N = Número de activos de la familia de productos.
Restauración de un activo.	Activo.	Realización dominio (o aplicación si es una característica propia).	X = A/B A = Número de requisitos de restauración implementados. B = Número de requisitos de restauración en las especificaciones.
Restauración de un producto.	Producto final, activos.	Realización aplicación.	X = Sumatorio de R de los activos del producto / N R = Restauración de un activo. N = Número de activos del producto.
Restauración de la familia de productos.	Familia de productos, activos.	Realización dominio.	X = Sumatorio de R de los activos de la familia de productos / N R = Restauración de un activo. N = Número de activos de la familia de productos.

3. Eficiencia

Métrica	Artefacto(s)	Fase(s)	Descripción
Tiempo de respuesta de una función de un activo.	Activo	Realización, pruebas dominio (o aplicación si es una característica propia)	X = tiempo de respuesta de una función.
Porcentaje de tiempo de respuesta de un activo.	Activo	Realización, pruebas dominio (o aplicación)	X = Sumatorio de T de las funciones del activo / N T = Tiempo de respuesta de una función de un activo. N = Número de funciones del activo.
Throughput time de un activo.	Activo	Realización dominio (o aplicación)	X = Número de tareas realizadas por un activo / T T = Unidad de tiempo.
Throughput time de un producto.	Producto, activos.	Pruebas aplicación	X = Número de tareas realizadas por un producto / T T = Unidad de tiempo.
Turnaround time.	Producto	Pruebas aplicación	X = Tiempo empleado en realizar un conjunto de tareas.
I/O Utilization del producto.	Producto	Pruebas aplicación	X = Número de buffers.
I/O Utilization Message Density del producto.	Producto	Pruebas aplicación	X = A/B A = Número de I/O relacionados con mensajes de error. B = Número de líneas de código directamente relacionadas a llamadas al sistema.
Memory utilization.	Producto	Pruebas aplicación	X = Tamaño en bytes de la memoria requerida.
Memory utilization message density	Producto	Pruebas aplicación	X = A/B A = Número de memoria relacionada con mensajes de error. B = Número de líneas de código directamente relacionados a las llamadas al sistema.
Transmission Utilization	Producto	Pruebas aplicación	X = bits/ tiempo Estimate the Transmission resource utilization requirements by estimating the transmission volumes.

4. Usabilidad

Métrica	Artefacto(s)	Fase(s)	Descripción
Completitud de la descripción del activo	Activos	Pruebas del dominio o aplicación.	OU = A/B A = Número de elementos comprensibles. B = Número total de elementos.
Capacidad de demostración	Activo, producto final.	Realización pruebas. Dominio, Aplicación.	CD = A/B A = Número de funciones con capacidad de demostración. B = Número total de funciones.
Cantidad de interfaces de usuario similares	Arquitectura LPS, activos, producto final.	Realización, Pruebas. Dominio, Aplicación.	A = Número de interfaces de usuario similares
Tipos de diseño de interfaces de usuario	Arquitectura LPS, activos, producto final.	Realización, Pruebas. Dominio, Aplicación.	A = Tipos de interfaces de usuario similares
Ratio de funciones evidentes para un usuario en un activo o producto.	Activo, producto final.	Realización pruebas. Dominio, Aplicación.	CD = A/B A = Número de funciones evidentes para un usuario. B = Número total de funciones.
Existencia de activos concretos para la facilidad de ayuda	Arquitectura LPS.	Diseño dominio, aplicación	Comprueba la existencia de activos concretos para la facilidad de ayuda.
Aseguramiento que se incluye en los productos el activo de facilidad de ayuda	Especificación requisitos, arquitectura producto, implementación producto, producto terminado.	Requisitos, diseño, realización, pruebas en aplicación.	X = A/B A = Número de activos para la facilidad de ayuda introducidos. B = Número de activos considerados necesarios.

5. Seguridad

Métrica	Artefacto(s)	Fase(s)	Descripción
Grado de seguridad mínimo impuesto para una contraseña	Modelo requisitos.	Requisitos, Realización	X = A * ((B+C+D+E)/4) A= Longitud de la contraseña B= Obligación de utilizar minúsculas. C= Obligación de utilizar mayúsculas. D= Obligación de utilizar números.

Métrica	Artefacto(s)	Fase(s)	Descripción
			E= Obligación de utilizar símbolos.
Grado de seguridad mínimo de una contraseña creada automáticamente.	Modelo requisitos.	Requisitos, Realización	X = A * ((B+C+D+E)/4) A= Longitud de la contraseña B= Obligación de utilizar minúsculas. C= Obligación de utilizar mayúsculas. D= Obligación de utilizar números. E= Obligación de utilizar símbolos.
Ratio de la seguridad de las contraseñas.	Modelo requisitos.	Requisitos, Realización	X = (A + B)/2 A = Grado de seguridad mínimo impuesto para una contraseña. B = Grado de seguridad mínimo de una contraseña creada automáticamente.
Ratio de encriptación de los datos.	Implementación	Realización dominio, aplicación	X = A/B A = Cantidad de datos encriptados. B = Cantidad de datos totales
Alcanzabilidad de información privada sin permisos.	Modelo de requisitos, implementación	Requisitos, realización.	X = A + B A = Número de funciones en el sistema que acceden a información privada sin previo registro del usuario. B = Número de transiciones sin autenticación previa.
Ratio de encriptación de un activo.	Activo	Realización dominio	X = A/B A = Cantidad de información encriptada de un activo. B = Cantidad de información total de un activo.
Ratio de encriptación de los activos de un producto.	Activo	Realización aplicación	X = Sumatorio de todos los A del producto / N A = Ratio de encriptación de cada activo que compone el producto. N = Número de activos del producto.
Ratio de encriptación de la familia de productos.	Activo	Realización dominio	X = Sumatoria de todos los A de la familia de productos / N A = Ratio de encriptación de cada activo que compone la familia de productos. N= Número de activos que componen la familia de productos.
Existencia de un historial de acciones en un	Activo	Requisitos y realización	Si existe el historial -> X=1 Si no existe el historial -> X=0

Métrica	Artefacto(s)	Fase(s)	Descripción
activo.		del dominio	
Ratio de existencias de un historial de acciones en un producto.	Activos y Producto final	Requisitos y realización de la aplicación	X= A/B A = Sumatorio de "Existencia de un historial de acciones en un activo" de cada activo del producto. N = Número de activos que componen el producto.
Ratio de existencias de un historial de acciones en la familia de productos.	Arquitectura de la LP y activos	Requisitos, realización del dominio	X= A/B A = Sumatorio de "Existencia de un historial de acciones en un activo" de cada activo de la familia de productos. N = Número de activos que componen la familia de productos.
Acceso al historial de acciones sin autorización.	Arquitectura de la LP y activos.	Requisitos,	X = B/A A = Número de funciones que acceden al historial. B = Numero de transiciones que pueden acceder al historial sin autorización.
Encriptación de la información del historial.	Producto final	Realización, test aplicación.	X = A/B A = Cantidad de información del historial encriptada. B = Cantidad de información del historial total.
Modificación de un activo por parte de otro activo.	Activos, arquitectura LPS.	Diseño, realización, pruebas. Dominio, aplicación.	X = B/A A = Número de funciones que pueden impactar en el activo V B = Funciones con impacto tratadas
Cantidad de información requerida para la autenticación.	Activo.	Requisitos, diseño, realización, pruebas. Dominio, aplicación.	X = Numero de "datos" requeridos para la autenticación. Entendemos por "datos", los strings a introducir (por ejemplo, si se pide usuario y contraseña, X es 3)
Calidad de la información requerida para la autenticación.	Activo.	Requisitos, diseño, realización, pruebas. Dominio, aplicación.	X = A/B A = Número de "datos" requeridos para la autenticación que son comprobados. B = Número de "datos" requeridos para la autenticación.
Finalización de sesión de un usuario.	Modelos requisitos, test	Requisitos y test, dominio y	X = Número de funciones/casos en los que el sistema cierra la sesión de un usuario (por finalización, desuso, cierre de la aplicación, apagado del

Métrica	Artefacto(s)	Fase(s)	Descripción
		aplicación.	ordenador, etc.).

6. Compatibilidad

Métrica	Artefacto(s)	Fase(s)	Descripción
Functional inclusiveness	Activos	Realización dominio, evolución.	X = A/B A = Número de funciones cubiertas por el nuevo software que produce resultados similares, confirmados en la revisión. B = Número de funciones en el software antiguo.
Available co-existence	Arquitectura de la LP, activos, producto final.	Diseño, Realización del dominio (y aplicación para características propias)	X = A/B A = Número de entidades con las cuales los productos pueden co-existir como está especificados. B = Número de entidades en la familia de productos que requieren co-existencia.
Data exchangeability	Activos	Realización Dominio y Aplicación	X = A/B A= Número de formatos de interfaz de información que han sido implementados correctamente en la especificación. B= Número de formatos de datos que son intercambiados como en la especificación.

7. Mantenibilidad

Métrica	Artefacto(s)	Fase(s)	Descripción
Component Compliance	Activo	Diseño o realización del dominio o aplicación.	CC = A/B A = Número de componentes reemplazables. B = Número total de componentes en el activo.
Functional Coverage	Activo, línea de productos, conocer los productos finales que vamos a crear	Diseño, implementación dominio (y también puede usarse en la aplicación o en la evolución por si queremos añadir la nueva	FC = S/N A = Número de aplicaciones utilizando la característica funcional i B = Número total de aplicaciones en la línea de productos. S = Sumatorio de A/B para cada i, tal que i=1 hasta N. N = Número total de características funcionales.

Métrica	Artefacto(s)	Fase(s)	Descripción
		característica).	
Architectural Commonality	Arquitectura LPS.	Diseño dominio.	AC = A/B A = Número de aplicaciones que pueden compartir la arquitectura de la LP. B = Número total de aplicaciones en la línea de productos.
Non-functional Coverage	Arquitectura LPS.	Diseño dominio.	NC = C/N A = Número de aplicaciones utilizando la característica no funcional i B = Número total de aplicaciones en la línea de productos. N = Número de características no funcionales que no son mantenidas por la arquitectura de la LP. C = Sumatorio de A/B para cada i, tal que i=1 hasta N
Non-functional Commonality	Arquitectura LPS.	Diseño dominio.	NFC = A * AC + B * NC AC = Architectural Commonality A = Peso dado a AC NC = Non-functional Coverage B = Peso dado a NC
Coverage of Variability	Arquitectura LPS.	Diseño dominio.	CV = A/B A = Número de puntos implementados en el activo. B = Número de puntos de variación incluidos en el alcance de la LP
Cumulative Applicability	Arquitectura LPS.	Diseño dominio.	CA = A * FC + B*NFC + C*CV A= Peso de FC FC B = Peso de NFC NFC C = Peso de CV CV
Effectiveness of Tailoring	Arquitectura LPS. Producto final	Diseño dominio, aplicación.	ET = A/B A = Número de puntos de variación resueltos efectivamente. B = Número total de puntos de variación.
Tailorability of Closed variability	Arquitectura LPS.	Diseño dominio.	TC = A = Número de variantes válidas para VPi (una variante válida es la variante la cual puede resolver (es decir, enlazar) un punto de variación

Métrica	Artefacto(s)	Fase(s)	Descripción
			sin efectos adversos o fallos). SA = Sumatorio de todos los A, para i=1 hasta N B = Número total de variantes para VP i SB = Sumatorio de todos los B, para i =1 hasta N VPi = is ith closed variation point among effectively resolvable variation points.
Tailorability of Open variability	Arquitectura LPS.	Diseño dominio.	TO = A/B A = Número de puntos variación abiertos válidos. B = Número total de puntos de variación abiertos.
Tailorability	Activo	Diseño, realización. Dominio, aplicación.	TL = ET * (A* TC + B * TO) ET TC A = Peso de TC. Ratio of closed variation points among the total effectively evolvable variation points. TO B = Peso de TO. Ratio of open variation points among the total effectively evolvable variation points. Nota: A + B debe ser igual a 1

8. Transferabilidad

Métrica	Artefacto(s)	Fase(s)	Descripción
Adaptabilidad de las estructuras de información de un activo.	Activo	Diseño, realización. Dominio, aplicación.	X = A/B A = Contar el número de estructuras de información del activo que son operables y que no están limitadas después de la adaptación B = Número total de estructuras de información requeridas para la capacidad de adaptación.
Ratio de la adaptabilidad de las estructuras de información de un producto.	Producto, activos.	Diseño, realización, pruebas. Dominio, aplicación.	X= A/B A = Sumatorio de "Adaptabilidad de las estructuras de información de un activo" de cada activo del producto. N = Número de activos que componen el producto.
Ratio de la adaptabilidad	Modelo de características,	Diseño, realización.	X= A/B

Métrica	Artefacto(s)	Fase(s)	Descripción
de las estructuras de información de la familia de productos.	activo.	Dominio.	A = Sumatorio de "Adaptabilidad de las estructuras de información de un activo" de cada activo de la familia de productos. N = Número de activos que componen la familia de productos.
Organizational Environment daptability	Activo.	Diseño, Realización. Dominio.	X = A/B A = Número de funciones implementadas las cuales son capaces de lograr los resultados requeridos en múltiples organizaciones y entornos de negocio B = Número de funciones que requieren tener la capacidad de adaptación al entorno organizacional.
Hardware Environment Adaptability	Activo. Producto.	Diseño, Realización. Dominio. Aplicación.	X = A/B A = Número de funciones implementadas las cuales son capaces de lograr los resultados requeridos en múltiples entornos hardware especificados B = Número de funciones que requieren tener la capacidad de adaptación a entornos hardware.
System software Environmental adaptability	Activo. Producto.	Diseño, Realización. Dominio. Aplicación.	X = A/B A = Número de funciones implementadas que son capaces de lograr los resultados requeridos en múltiples sistemas software especificados B = Número de funciones que requieren tener la capacidad de adaptación a entornos de sistemas software.
Ease of setup retry	Activo. Producto.	Diseño, Realización. Dominio. Aplicación.	X = A/B A= Número de operaciones de reinstalación implementadas. B= Número total de operaciones de reinstalación requeridas.
Installation effort	Activo. Producto.	Diseño, Realización. Dominio. Aplicación.	X = A/B A = Número de pasos de la instalación automatizados que estén implementados. B = Número de pasos de instalación requeridos.
Capacidad de la instalación para ser flexible y	Activo. Producto.	Diseño, Realización. Dominio.	X = A/B A = Número de operaciones de instalación personalizables

Métrica	Artefacto(s)	Fase(s)	Descripción
personalizable.		Aplicación.	implementadas. B = Número de operaciones de instalación con la capacidad de personalización requeridas.