Análisis de metodologías de validación de una línea de producto de software: Caso aplicable a un Cajero automático

Analysis of validation methodologies of a software product line: Case applicable to an ATM

Luis Fernando Cadavid Zuluaga 1[1]*
Mauricio Toro Bermudez 2[2]

[1] Universidad EAFIT. Departamento de Informática y Sistemas. Medellín, Colombia. E-mail:lcadavid96@gmail.com.
[2] Universidad EAFIT. Departamento de Informática y Sistemas. Medellín, Colombia. E-mail:mtorobe@eafit.edu.co

RESUMEN:
El concepto de líneas de producto de software se asocia con el conjunto de sistemas de software que comparte entre sí una serie de características en común. En este orden, en la presente investigación se evalúan diferentes metodologías enfocadas en los procesos de validación de software, siendo estas centralizadas en las líneas de producto de software, para lo cual se analizan algunas de las principales metodologías principales, determinándose aspectos como el campo de aplicación, los atributos de calidad abordados y los criterios de validación, las fases del proyecto, los modelos y artefactos sobre el que se valida, las entradas y salidas consideradas, entre otros elementos. De acuerdo con los resultados del proceso investigativo, se selecciona la metodología ATAM (Architecture Tradeoff Analysis Method), considerando el hecho de que es en esencia una herramienta que posibilita el reuso de información, considera la evaluación de varios atributos y se centraliza en el modelado de objetivos. Esta herramienta fue aplicada al diseño de un software de cajero automático, en la que se analizó la priorización de los escenarios, encontrándose que se requiere de una prioridad alta en los módulos de validación de tarjeta y retiro de dinero.
Palabras clave: línea de productos de software, validación, atributos de calidad, ATAM.

ABSTRACT:
The concept of software product lines is associated with the set of software systems that share a series of characteristics in common. In this order, in this research different methodologies focused on software validation processes are evaluated, being centralized in the software product lines, for which some of the main methodologies are analyzed, determining aspects such as the field of application, the attributes of quality addressed and the validation criteria, the phases of the project, the models and artifacts on which it is validated, the inputs and outputs considered, among other elements. According to the results of the research process, the ATAM methodology is selected, considering the fact that it is in essence a tool that enables the reuse of information, considers the evaluation of several attributes and is centralized in the modeling of objectives. This tool was applied to the design of an ATM software, in which the prioritization of the scenarios was analyzed, finding that a high priority is required in the

modules of card validation and withdrawal of money.
Keywords: software product line, validation, quality attributes, ATAM.

INTRODUCCIÓN

Los métodos de validación, en los procesos de análisis, permiten asegurar la calidad de los productos; de allí se justifica la importancia de los métodos formales de validación [1], ya que estos contribuyen al cumplimiento de los estándares de calidad, siendo la base para la aplicación de técnicas en todas las fases de desarrollo de software [2].

En la corta historia de los sistemas de información se hace evidente que estos fueron diseñados para ayudar a las empresas en la búsqueda de procesos, respuestas ágiles y en la demostración de la adaptación y competitividad en el medio. Ahora bien, dado que estos sistemas de información cada día incrementan su complejidad y que la construcción de un software, en tiempos adecuados y bajo costos razonables, se ha convertido en un problema constante, resulta inminente la detección y corrección de errores, pues es más costoso detectar y corregir estos en las últimas fases, que al inicio del proyecto [3].

Para tener una mejor comprensión del concepto de validación, Uzcátegui, Ortega y Delgado [4] lo han definido como una versión ampliada del ciclo de vida completo del desarrollo de sistemas, que incluyen tareas o pasos para cada fase, funciones desempeñadas en cada tarea, productos resultantes, normas de calidad y técnicas de desarrollo que se utilizan en cada tarea. Por su parte Aristides Dasso y Ana Funes definen que “you are validating when you are answering the question: Are we building the right product?"(Validas cuando estás respondiendo a la pregunta: ¿Estamos construyendo el producto correcto?) [5].Teniendo en cuenta estas definiciones y la existencia de los métodos de validación, es posible tomar y aplicar estos presupuestos teóricos a una línea de productos software, la misma que se desarrollará a lo largo de este artículo.

Complementando la descripción anterior, la validación del software se asocia con la comprobación que el software hace lo que el usuario espera; es decir que el software cumple las expectativas del cliente [6]. Es así que la validación se considera como la confirmación mediante exanimaciones y presentación de evidencias objetivas de que los requerimientos particulares de un uso que se hayan especificado se han cumplido [2].

Respecto al paradigma de las familias de producto software, durante el diseño y la gestión de los artefactos reutilizables, se analiza cuál es la variante que mejor se acomoda a las necesidades del usuario o a un producto de una familia. Las familias de productos, según el Software Engineering Institute (SEI) en Carnegie Mellon University [7], ha sido descrita como, “un conjunto de sistemas de software que comparten un conjunto común de características que satisfacen las necesidades específicas de un determinado segmento de mercado o misión, que se desarrolla a partir de un conjunto común de componentes” [8]. Esto indica que son grupos de productos con un conjunto de semejanzas y diferencias, donde cada uno es derivado de una plataforma con requisitos comunes a toda la familia. Ahora bien, un defecto en un producto genera fallas en la familia de productos, por lo que en este artículo quedará claro que es necesario asegurar los productos de la familia garantizando su mantenimiento, lo que se logra por medio de la aplicación de métodos de validación que permitan validar el diseño de los

artefactos, a su vez el modelo que representa la familia y, por supuesto, de los productos derivados a partir de esta.

Y es que la evolución de la ingeniería en familias de productos, demanda desafíos para el aseguramiento de la calidad y es de vital importancia garantizarlo en los productos que conforman dicha familia. Para esto, resulta relevante la capacidad de mejoramiento y la reutilización sistemática para derivar productos específicos, lo cual conlleva al desarrollo de familias de productos software, en la creación de artefactos de arquitectura, diseño y en general, todos los componentes reutilizables que se puedan usar en el proceso de desarrollo de software. Postulado que se evidencia en este artículo, al considerar que con el paso del tiempo se dan avances en la ingeniería de software, que aseguran nuevos desarrollos de mejor calidad [9].

Así, validar una familia de productos consiste en garantizar la satisfacción de las necesidades para las cuales fue creado cada uno de los productos de la familia, elementos variables y activos del núcleo que deben ser objeto de validación. Algunos métodos han alcanzado un nivel de madurez alto, en particular, aquellos que presentan ventajas para aplicar conceptos de calidad de software y evaluaciones arquitectónicas en el desarrollo de software tradicional. Algunos de estos son ALPSM (Architecture Level Prediction of Software Maintance), ABD (Architecure Based Design), SAAM (Software Architecture Analysis Method) , FAAM (Family-Architecture Assessment Method), entre otros; los cuales serán presentados más adelante [10]. En el caso explícito de este estudio, se escogió ATAM (Architecture Tradeoff Analysis Method) como metodología para la ejemplificación del proceso de validación de una línea de producto de software, puesto que se trata en sí de las pocas que se enfocan en el reuso de la información y considera la evaluación de diferentes atributos, centrando su interés en el modelado de objetivos.

Finalmente es de destacar que en las diferentes fases del ciclo de vida de una familia de productos, se puede lograr que los procesos de análisis aseguren el cumplimiento de calidad de todos los productos de una misma familia; en ese sentido, la validación permitirá soportar actividades de la ingeniería de software como la obtención de requisitos, diseño del sistema, diseño de objetos, entre otros, siendo en el caso de este artículo enunciado la forma cómo se comportan al adaptarlos a modelos con variabilidad, asegurando la calidad de los productos reutilizables dentro las familias de productos.

ANÁLISIS TEÓRICO

Línea de producto de software-LPS

Son múltiples los conceptos asociados con la línea de producto de software, siendo uno de los más abordados el descrito por Clements, quien lo establece como “el conjunto de sistemas software, que comparten un conjunto común de características (features), las cuales satisfacen las necesidades específicas de un dominio o segmento particular de mercado, y que se desarrollan a partir de un sistema común de activos base (core assets) de una manera preestablecida” [11].

De acuerdo a lo anterior, son dos las características fundamentales de la LPS, siendo la primera

el hecho de que no tiene como objetivo el desarrollo de un producto, sino el de un conjunto de los mismos; los cuales poseen una característica en particular. La segunda característica se asocia con la búsqueda de satisfacer las necesidades específicas de una parte del mercado, siendo estas necesidades solventadas a partir de un conjunto de activos reutilizables [12].
Ampliando lo descrito por *Clements y Northrop* respecto a LPS, [11], *Krueger* [13] alude a las mismas como las técnicas de ingeniería que se usan para conformar un portafolio de software similares, por medio de activos de software que se utilizan en un medio común de producción. Ahora bien, en el caso de Gomma [14], este considera a la LPS como una familia de sistemas de sofrware que poseen una funcionalidad común y a su vez una variable [15].
Complementando lo dispuesto anteriormente, el Instituto de Ingeniería de Software de la Universidad Carnegie Mellon, describe a la LPS como: "un conjunto de sistemas de software que comparten un conjunto común y gestionado de características que satisfacen las necesidades específicas de un segmento de mercado particular o misión, y que son desarrolladas de forma prescrita a partir de un conjunto común de elementos clave" [16]. Bajo esta perspectiva, la reutilización permite la validación de los objetos y sus características, para lo cual alude a abstracciones para generar nuevos modelos [10].

Línea de producto de software-LPS

En lo que respecta a los elementos principales de la LPS; estos se clasifican en los activos de software, las decisiones de productos, el proceso de producción y los productos del software. En el caso de los activos de software, estos se refieren a la colección de partes del software como requisitos, diseños, componentes, casos de pruebas, entre otros; quienes en sí se configuran y constituyen de forma prescrita con el objetivo de producir las líneas de producto de software [17].
Por su parte en los modelos y decisiones de los productos del software se describen los aspectos, variables y opcionales de los productos de la línea, en el cual cada uno es definido de acuerdo a una serie de decisiones. En el proceso de producción se establecen los mecanismos, herramientas o pasos para llegar a componer y configurar los productos a partir de los activos de entrada, siendo las decisiones del producto usados para establecer cuales activos de entrada se deben incorporar y la forma de configurar los puntos de variación de esos activos. Por último, la salida corresponde al conjunto de todos los productos que son diseñados por la línea de productos [15].
Ahora bien, desde una estructura mucho más elaborada (Figura 1) se define que el proceso de ingeniería de líneas de productos de software se encuentra compuesta por dos etapas: la ingeniería del dominio y la ingeniería de la aplicación. Cada una de estas etapas se constituye de subprocesos como por ejemplo en la ingeniería de aplicación se abordan aspectos que más adelante se profundizarán, entre los que se encuentran los requisitos, diseño, implementación y pruebas, siendo este último de interés del presente artículo, el cual se orienta a evaluar diferentes técnicas o herramientas para la validación del software.
En la etapa de ingeniería del dominio, lo que se hace es elaborar los componentes reutilizables, determinando similitudes y diferencias. A esta también se le conoce como el modelo de dominio que está constituido de toda clase de componentes de software con capacidad

suficiente para asegurar la flexibilidad que habilite la construcción de una amplia gama de productos. Así mismo, en esta etapa se incorporan subprocesos como la administración del producto encargada de definir los componentes que se incluyen en el producto, para lo cual debe evaluar los objetivos y estrategias requeridas por el mercado [18].

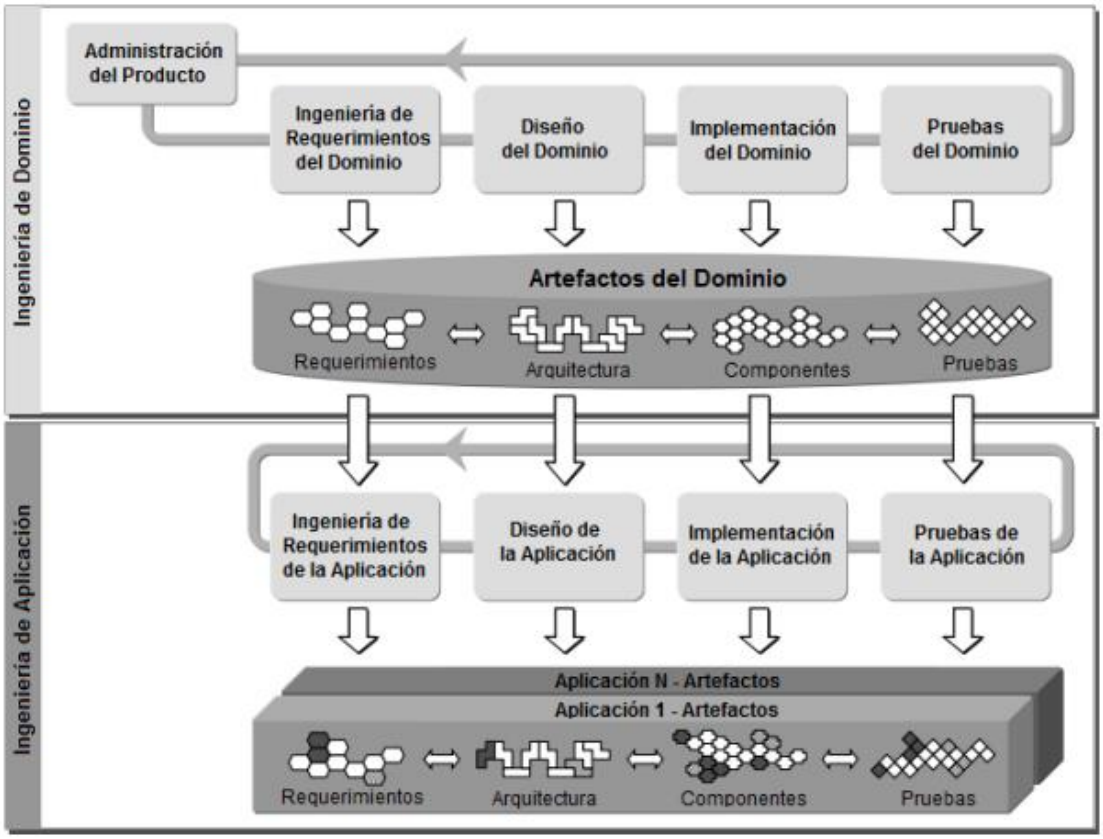

Figura 1. Estructura de la Línea de Producto de software.

Otro concepto intrínseco al proceso de diseño del modelo del dominio es la variabilidad, siendo este un concepto relacionado con las características comunes y cambiantes de los productos. Dicha variabilidad se caracteriza a partir del establecimiento de una serie de interrogantes como ¿qué es lo que varía? En el cual se busca establecer los elementos variables de un producto a los cuales se le denomina sujetos variables, el ¿por qué varía? Que determina las causas de la variación y el ¿cómo varía? Que permite establecer las diferentes formas que toman un elemento o propiedad [19].

En el caso de la ingeniería de aplicación, en esta se construye cada uno de los productos relacionados con la línea de productos por medio de la plataforma dispuesta en la ingeniería de dominio. En dicho orden, el objetivo principal es obtener un mayor grado de reutilización de los componentes [20]. Respecto a los subprocesos de la ingeniería de aplicación, estas etapas corresponden a:

- Requisitos: Se recibe los requisitos de entrada del dominio junto con la hoja de ruta de administración del producto. Así mismo se desarrolla la especificación de requerimientos que sirven para conseguir la implementación que se busca.

- Diseño: Este recibe la especificación de los requisitos a partir de lo cual se diseña la arquitectura para el producto, seleccionando y configurando las partes que indispensables de la arquitectura del dominio.
- Implementación: Se encarga de la creación del producto, para lo cual es indispensable seleccionar y configurar cada uno de los componentes de la plataforma del dominio [21].
- Pruebas: En este punto se valida la aplicación, estableciendo que su funcionamiento sea consistente con la especificación del requerimiento, por lo cual recibe como entrada la aplicación ya constituida junto con los elementos de prueba del dominio, produciéndose finalmente un informe con los resultados de pruebas ejecutadas [22].

En el caso explícito de esta investigación, el subproceso corresponde al de pruebas de la etapa de aplicación, siendo el objetivo principal la evaluación de las metodologías de validación de líneas de producto de software contextualizado a un ejemplo correspondiente al funcionamiento de un cajero automático, es un en un ejemplo sencillo ya que por las diferentes operaciones que pueden realizar, estas varían dependiendo de las necesidad de cada banco.

Validación de software

Los procesos de validación de software son entendidos desde diferentes posturas, siendo en ocasiones confundidos erróneamente con la de verificación. Bajo esta premisa, la verificación alude principalmente a la determinación de si los productos en cierta fase del desarrollo del software cumplen los requisitos dispuestos en la fase anterior, mientras que la validación evalúa si al final del proceso de desarrollo se logra dar cumplimiento a las necesidades de los clientes a través del software [23].

Ahora bien, bajo cualquiera de estas dos perspectivas, la realización de las verificaciones o las validaciones de las mismas trascienden al uso de dos técnicas: la evaluación estática y la dinámica. La primera de estas estudia los diferentes modelos que constituyen el software, buscando para tal efecto posibles fallas de los mismos, pudiéndose aplicar tanto a los modelos de análisis como a los de diseño y de código. Por su parte, la técnica de evaluación dinámica lo que busca es detectar fallas entre la salida esperada y la real; por lo cual en ocasiones se le conoce como prueba de software o testing, aplicándose casi siempre sobre el código [24].

Existen otros modelos basados en la evaluación del diseño arquitectónico, los requerimientos de atributos de calidad como lo es por ejemplo ATAM, por medio del cual se busca entender una arquitectura desde el punto de vista de la modificabilidad en el que se requiere entender como observar y medir ese atributo, así como evaluar de qué manera se impactan dichos atributos ante las diferentes decisiones arquitectónicas.

En este orden, ATAM permite caracterizar los atributos por medio del conocimiento de los atributos de calidad, como desempeño, modificabilidad, disponibilidad, usabilidad y seguridad. Ahora bien, para poder validar o evaluar los atributos se debe caracterizar los estímulos externos, las decisiones arquitectónicas y las respuestas, en donde los primeros corresponden a eventos que hacen que la arquitectura responda o cambie. Por su parte, las decisiones se asocian con elementos de la arquitectura como componentes, conectores y sus propiedades

que tienen un impacto directo sobre los atributos, al igual que sobre la latencia y rendimiento [25].

Atributos de calidad

Los atributos de calidad son características no funcionales que se contemplan como deseadas en un sistema de software. Aun así, en todos los sistemas de software no se consideran la totalidad de estos atributos o cualidades, puesto que algunos son más relevantes que otros, ello dependiendo del sistema. De acuerdo con el Instituto Internacional de Norma ISO, los atributos de calidad se asocian con la totalidad de las características de un producto o servicio que se relaciona con su capacidad de satisfacer las necesidades enunciadas o implícitas [26].
Dentro de los atributos de calidad que se abordan en esta investigación se encuentran:

- Funcionalidad: La funcionalidad se relaciona con la existencia de un conjunto de funciones (servicios) para satisfacer necesidades. Ya que es una característica fundamental en todo software; ella determina la capacidad del producto del software para proveer funciones que cumplan con necesidades específicas o implícitas, cuando el software es utilizado bajo ciertas condiciones.
- Fiabilidad: Es la capacidad para mantener el nivel de desempeño bajo condiciones enunciadas y períodos de tiempo enunciados.
- Mantenibilidad: Es el esfuerzo necesario para la ejecución modificaciones especificadas.
- Portabilidad: Es la capacidad de transferir un sistema de un ambiente a otro.
- Utilizabilidad: esfuerzo necesario para el uso eficaz del sistema.
- Eficiencia: relación entre rendimiento y recursos utilizados. En esencia, se trata del atributo que dice cómo el sistema utiliza el equipo (procesador, espacio en disco, memoria, ancho de banda, etc). Si un sistema consume muchos recursos su rendimiento se ve afectado y por ende el usuario [27].

ESTADO DE LA CIENCIA

La ingeniería de software se ha orientado en la búsqueda de métodos que ayuden a evaluar los productos desde su construcción, el objetivo de los métodos es mejorar la productividad en el desarrollo y la calidad del producto software, ya que uno de los problemas en el desarrollo es la dificultad para predecir el comportamiento del software esperado [28].
En la actualidad se presentan propuestas de métodos [29] , metodologías [30], aseguramiento [31], modelos de calidad [32] y métricas [33], cuyo objetivo es garantizar el correcto desempeño de los productos bajo cualquier situación y el aseguramiento del cumplimiento de los requisitos de los usuarios. Todos han ido aumentado su grado de aceptación a la hora emplear métodos efectivos. Una de las propuestas son los métodos de validación, los cuales han ido en creciente demanda, y en el diseño de nuevas metodologías de validación [34], entre los objetivos está evaluar el diseño de arquitecturas, evaluación de arquitecturas y de requisitos

a fin de evaluar diferentes atributos de calidad.
Ahora bien, los métodos de validación se han extendido en los últimos años como un instrumento análitico y algunos de ellos han alcanzado un nivel de madurez que exponen ventajas para aplicar conceptos de calidad de software y evaluaciones arquitectónicas, estos métodos pueden clasificarse en dos grandes grupos, uno para aquellos que pueden representar un solo sistema, que se derivan del desarrollo tradicional y el otro para familias de productos, cuya particularidad es la optimización de procesos y recursos de elementos comunes entre los productos de la familia.
En las familias de productos ha surgido los procesos de reutilización, aspecto que se ha logrado debido al soporte de una mayor variabilidad de los sistemas de software, lo cual les otorga a los sistemas la habilidad de cambio o de personalización [35]. Debido a la complejidad en la gestión de la variabilidad, en estudios recientes surgieron métodos de validación que ayudan a garantizar la calidad de una familia de productos, de los cuales muy pocos manejan variabilidad tales como: ESSAMI que es una combinación de análisis y reutilización [36], FAAM (se centraliza en la evaluación de arquitecturas en los aspectos estratégicos de la familia) [37], SACAM que se enfoca en un proceso de selección de una arquitectura, ATAM en donse se evalúa las decisiones arquitectónicas en comparación con los atributos de calidad [38], ABD que es un método iterativo que incluye la realización de un mini-proceso ATAM en cada iteración [39], ALPSM que define un perfil de mantenimiento basado en su arquitectura [40], SAAM enfocado en la evaluación de atributos de calidad [11], QAW que desarrolla un análisis de la arquitectura con atributos críticos de calidad [41], PASA que realiza una evaluación del rendimiento de arquitecturas de software [42], y finalmente SBAR que ejecuta la evaluación de los atributos de calidad [43].
Complementando lo dispuesto anteriormente, [44] establecen que existe multiples riesgos en el proceso de reutilización de una arquitectura de software, la cual no se detecta al momento de definir el nuevo producto sino hasta cuando se busca integrar las diferentes partes del producto y no se logra un producto final conforme a sus requisitos. Ante esto surge la necesidad imperante de contar con una arquitectura inicial correcta, extensibles y reutilizable a nivel de todos los productos dela línea de producto de software.
Para dar solvencia a tal problemática surge lo que los autores deniminan la “arquitectura de referencia” en el que se describe un marco de trabajo generalizable para el rango de los productos en la Línea de Producto. En este se considera elementos de entradas, de proceso y de salida, siendo entre los pasos constitutivos de la arquitectura de referencia los siguientes: 1) la identificación de la variabilidad en los tipos de aplicación, 2) la identificación de la variabilidad en el estilo arquitectónico, 3) la identificación de la variabilidad en las estrategias de despliegue, 4) la identificación de la variabilidad en los aspectos técnicos, 5) la construcción del modelo de variabilidad de la arquitectura de referencia y la 6) documentación de la arquitectura de referencia.

Fuente: [44]

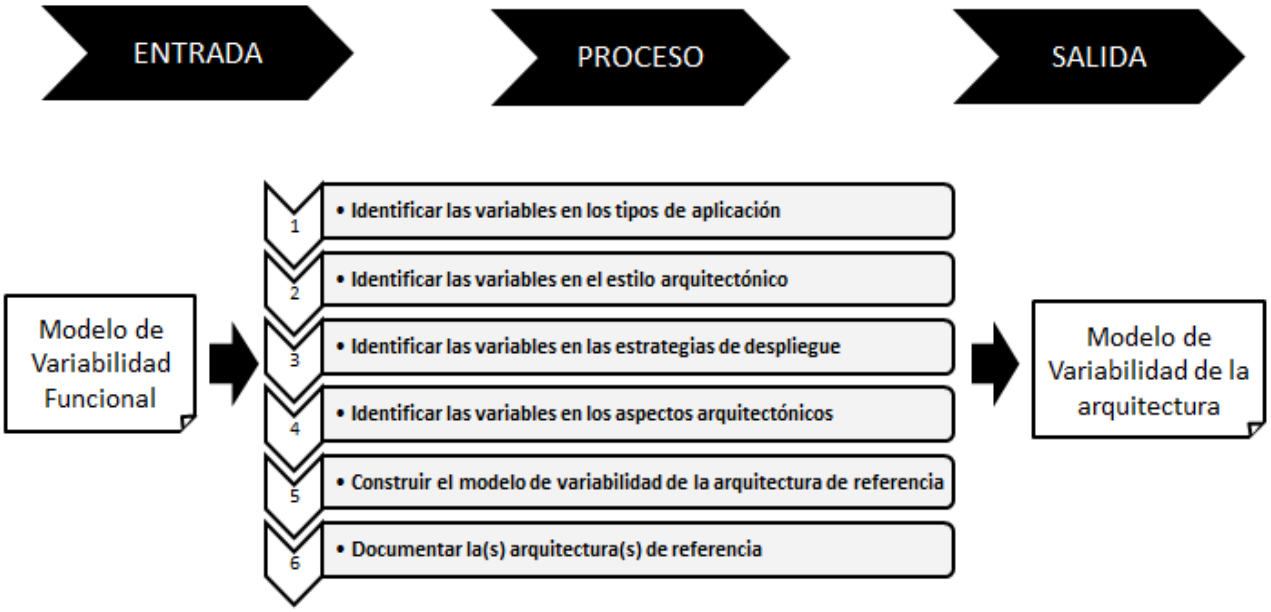

Figura 2. Proceso para definir la arquitectura de referencia.

Una vez abordado en el estado de la ciencia que permite considerar los avances que se tiene en torno a la implementación de herramientas de validación de las LPS, se procede a establecer la metodología sobre la cual se sustenta la presente investigación.

METODOLOGÍA

El estudio implementado corresponde a uno de índole cualitativo, en el cual se analiza las características de las diferentes metodologías de validación de software asociado a las LPS. En dicho orden, se considera lo dispuesto por la norma NTC-ISO 9000:2015, la cual relaciona las actividades de validación como la confirmación, mediante evidencia objetiva de que se han cumplido los requisitos para una aplicación específica prevista [26].

En dicho orden y considerando el objeto de estudio, se parte realizando una revisión sobre diferentes metodologías de validación de LPS, estableciendo dentro de dicho estudio bibliográfico las de mayor abordaje. Dentro de los documentos evaluados se destacan los estudios realizados por [7] [10] [14][17][18] [35][38][44][45], a partir de lo cual se determinaron aspectos claves como:

- Tipo de criterios arquitectónico estructural evaluado. componentes, conectores, estilos arquitectónicos, patrones, uso de puertos, propiedades, interfaces y vistas.
- Levantamiento de requisitos de calidad. Se determina el tipo de mecanismo de levantamiento de los requisitos de calidad usado, ya sea a través de la generación de escenarios, el análisis de casos de uso, la construcción de árboles de utilidad, el modelado de objetivos o el levantamiento previo de requisitos.

- Enfoques de evaluación de requisitos. Se determina si la evaluación de los enfoques se realiza por medio del análisis de escenarios, el razonamiento basado en la experiencia o el análisis de factores.
- Atributo de calidad evaluado: Se determina el tipo de atributo evaluado, ya sea de funcionalidad, modificabilidad, la mantenibilidad, la conveniencia del diseño, la interoperabilidad, el desempeño, la disponibilidad, el rendimiento, la seguridad, la calidad, la escalabilidad, la factibilidad de uso y la capacidad de prueba.
- Generación de arquitectura: Se determina el uso de herramientas como el análisis de escenarios, el análisis de casos de uso, el análisis de vistas arquitectónicas, la generación de especificaciones, la identificación de estilos y patrones arquitectónicos y estimaciones cuantitativas.
- Posición de expertos. Se identifican como cinco los grupos identificables: en el que todas las actividades son grupales, en el que la mayoría son grupales, en el que algunas son grupales, en las que todas las actividades son grupales o individuales y las que están sin determinar.
- Objetivos. Está orientado hacia el levantamiento de requisitos, el diseño de arquitectura y la evaluación de la arquitectura.

Los métodos, posterior a ser caracterizados son evaluados y analizados, determinándose el que mejor se acopla a los requerimientos de una LPS, explícitamente para un caso de aplicación que se relaciona con el funcionamiento de un cajero automático; considerándose el hecho de que se acopla a las características de una LPS y es de fácil entendimiento. Las principales características de cada uno de las metodologías fueron resumidas a través de tablas, de forma que su comprensión y la manera de compararlos con otras metodologías se facilitaran. Es así que se organizaron en tres tablas principales en donde cada fila definía el método y en las columnas los parámetros a evaluar, usándose para tal fin un conjunto de siglas que se describen a lo largo de la investigación.
El cajero automático se encarga de entregar a los usuarios los mismos servicios que un cajero humano, pero de manera autómata, eficiente y prolongada ya que funciona día y noche. Por ende, el sistema que ha de crearse debe ser capaz de manejar datos donde el porcentaje de error que presente sea mínimo, además debe resguardar los datos ingresados y mostrarlos por pantalla, teniendo siempre en cuenta qué datos pueden ser o no mostrados.

RESULTADOS

Para establecer la metodología más propicia a implementar en el proceso de validación de software en una línea de producto, se debe partir del análisis de las mismas, proceso que se realizó por etapas.

Etapa 1. Definición de los métodos: En el caso de la metodología ABD (Architecure Based Design), se enfoca en el diseño de la arquitectura centralizado en tres pilares [39]: la descomposición funcional del problema basado en aspectos de bajo acoplamiento y alta

cohesión , el desarrollo de los requisitos de calidad y negocio, considerando la elección de estilos arquitectónicos adecuados y el uso de patrones de interacción de los elementos de la arquitectura con la infraestructura [45].

Respecto a ALPSM (Architecture Level Prediction of Software Maintance), este se trata de un método de predicción de la capacidad de mantenimiento del software durante el diseño de la arquitectura, para lo cual considera como parámetros la especificación de los requisitos, el diseño de la arquitectura, la experiencia de los ingenieros del software y a su vez datos históricos como entradas y salidas del esfuerzo promedio para una tarea de mantenimiento [40] [45].

Respecto a ATAM (Architecture Tradeoff Analysis Method), puede usarse para crear definiciones preliminares de la arquitectura, analizar las decisiones arquitectónicas cuando no existe código, analizar las diferentes arquitecturas para el desarrollo de un sistema e incluso analizar sistemas ya existentes, de allí que sea la evolución de SAAM [27]. Ahora, dentro de las características básicas que describen a ATAM se destaca el hecho de que es un método orientado a escenarios, evaluando la arquitectura los requisitos [41], para lo cual solicita de una descripción detallada de las estructuras del sistema. Así mismo y uno de los puntos claves es que su ejecución o aplicación requiere de pocos recursos y tiempo corto, además de considerar no solo los requisitos técnicos, sino también aspectos sociales y de negocio [46] [47].

FAAM (Family-Architecture Assessment Method) es una metodología que centra su accionar en torno a la interoperabilidad y la extensibilidad, la cual además involucra a las partes interesadas de la familia de productos en el proceso de creación de productos [48]. Una de las características principales de FAAM radica en el hecho de que presenta descripción explícita de los stakeholders, haciendo principal énfasis en el redimensionamiento de las tareas de cada uno bajo un contexto de evaluación de arquitecturas de la familia de sistemas de información [45].

SAAM (Software Architecture Analysis Method) se enfoca en la validación de la arquitectura en desarrollo como para sistemas de software ya implementados. En este orden, debe mencionarse que SAAM no se enfoca en la evaluación de los atributos en función de la calidad de manera abstracta sino explícitamente de acuerdo a los requisitos. Otra de las características que describen a este método es el consumo de pocos recursos, debiéndose aplicar antes de que el sistema se implemente, ello aunque no se tenga razón de los detalles finales del sistema, es decir, se necesita de una descripción clara pero no muy detallada de la arquitectura. Por último, debe mencionarse que se trata de un método que es conducido por escenarios, por lo cual es indispensable definir el contexto cuando la calidad de la arquitectura es función del mismo [48] [49].

PASA (Performance Assessment of Software Architectures) por su parte es un método de evaluación del desempeño de arquitectura de software que usa los principios y técnicas de SPE (software process engineering que se relaciona con la definición, implementación, medición y mejora de los procesos de Ingeniería del Software) con el objetivo de identificar áreas potenciales de riesgo dentro de la arquitectura para lo que considera parámetros como el rendimiento y los objetivos de calidad. Respecto a la aplicación de esta metodología, se centraliza en nueve pasos que corresponden en su orden a la revisión del proceso, revisión de la arquitectura, identificación de casos de uso crítico, selección de escenarios de rendimiento clave, identificación de objetivos de desempeño, clarificación y discusión sobre la arquitectura,

análisis de la arquitectura, identificación de alternativa y presentación de resultados [50].
QAW (Quality Attribute Workshps) por su parte proporciona un método para la determinación de un sistema crítico basado en el análisis de los atributos de calidad, al igual que disponibilidad, rendimiento, seguridad, interoperabilidad y modificabilidad. En este orden, QAW complementa la arquitectura ATAM, puesto que identifica los atributos relevantes de calidad y permite establecer los requisitos del sistema antes de la existencia de una arquitectura de software [41].
SACAM (Software Architecture Comparison Analysis Method) es por su parte un método de evaluación temprana que se basa en el análisis de escenarios. Para tal objetivo sigue una serie de pasos que incluye en primera medida la preparación o la determinación de las entradas disponibles como candidatos a la arquitectura y objetivos de negocio; en la etapa dos se realiza la clasificación de los criterios, siendo en la tercera etapa en donde se determina las vistas arquitectónicas, las tácticas, los estilos y los patrones [51]. la cuarta etapa se determina las vistas arquitectónicas cada uno de los candidatos y se detectan los indicadores que soportan los escenarios de atributos de calidad. En la etapa cinco, se le asigna a cada criterio una puntuación basado en la evidencia proporcionada en el paso anterior. Finalmente, se resumen los resultados del análisis de validación [51].

Etapa 2. Criterios arquitectónicos: Para establecer el método a aplicar en el proceso de validación del software propuesto (cajero automático), se consideraron un conjunto de elementos o criterios arquitectónicos descritos por autores como Thiel, Babar, Grimán, entre otros [52] [53] [54]. En la Tabla 1, se muestra algunos conceptos arquitectónicos estructurales que son ampliamente aceptados y evaluados en los métodos de validación de software, siendo considerados entre ellos elementos como componentes (CO), conectores (CT), estilos arquitectónicos (EA), patrones (PA), uso de puertos (PT), propiedades (PR), interfaces (IT) y vistas (VT).

Tabla 1. Conceptos arquitectónicos en las metodologías analizadas.

Metodología/Criterio	CO	CT	PT	VT	IT	PR	EA	PA
ABD	X							
ALMA	X							
ALPSM	X							
QAW	X			X				
ASSAM	X							
ATAM	X							
CBAM							X	
CBSP	X	X						
FAAM	X					X		
MECABIC	X		X		X			
PASA								X
PRESKRIPTOR	X	X				X		
QUADRAD	X	X		X				x

SAAM	X							
SACAM	X	X						

Los métodos seleccionados dentro del análisis fueron considerados basados en el análisis del estado del arte, determinándose los de mayor aplicabilidad en el campo de la validación de software, junto con la diversificación de los criterios de evaluación que cada uno de ellos establece, esto con el objetivo de realizar una comparación de las características que cada uno ofrece. Debe mencionarse que no todas las metodologías dispuestas en la tabla anterior fueron mencionadas, se seleccionaron las de mayor aplicabilidad en el campo de la validación y aquellas que son implementables en términos de las líneas de producto de software.

Considerando las bases de conocimiento, se establece que muchos de los métodos poseen un alto grado de dependencia, esto respecto a la posición de los expertos; elemento que lleva a que se trate en cierta medida de validaciones subjetivas. En este orden, son cinco los grupos identificables: en el que todas las actividades son grupales (TG), en el que la mayoría son grupales (MG), en el que algunas son grupales (AG), en las que todas las actividades son grupales o individuales (TGI) y las que están sin determinar (SD).

Ahora bien, uno de los aspectos fundamentales que atañen a este proceso investigativo es utilizar una metodología aplicable a las líneas de productos de software, que se sustenta explícitamente en el reuso de información. De los métodos anteriormente descritos, tan solo ABD, FAAM y ATAM incluyen actividades relacionadas con los procesos de reuso, siendo orientados en sí al desarrollo del método y al uso del conocimiento adquirido respecto a los mecanismos y soluciones arquitectónicas.

Respecto al proceso de levantamiento de los requisitos de calidad, se utilizan principalmente mecanismos como la generación de escenarios (GE), el análisis de casos de uso (CU), construcción de árboles de utilidad (AU), el modelado de objetivos (MO), el levantamiento previo de requisitos (LPR) y otros.

Respecto a los enfoques de evaluación de los requisitos de calidad, son tres las categorías consideradas: la basada en el análisis de escenarios (AE) que busca sistematizar de forma explícita los requisitos de calidad, el razonamiento basado en la experiencia (RE) que es una herramienta intuitiva y subjetiva sobre el diseño y el análisis de factores (AF).

Dentro del proceso de evaluación de los atributos de calidad, se hace la clasificación en función al parámetro, existiendo en dicho orden asociados a la funcionalidad (F), la modificabilidad (MO), la mantenibilidad (MA), la conveniencia del diseño (CD), la interoperabilidad (IT), el desempeño (D), la disponibilidad (DP), el rendimiento (R), la seguridad (S), la calidad (C), la escalabilidad (E), la facilidad de uso (FC) y la capacidad de prueba (CP).

En cuanto a los mecanismos de generación de arquitectura, en este se busca es encontrar las posibles soluciones centralizados en torno a herramientas como el análisis de escenarios (AE), el análisis de casos de uso (ACU), el análisis de vistas arquitectónicas (AVA), los cuestionarios (C), la generación de especificaciones (GE), la identificación de estilos y patrones arquitectónicos (PA) y estimaciones cuantitativas (EC).

Respecto a los objetivos generales asociados a los procesos de validación de software, estos se enfocan en tres ejes: el levantamiento de requisitos (LR), el diseño de arquitectura (DA) y la evaluación de la arquitectura (EA). En el caso del levantamiento de requisitos, son dos las ramas de estudio, la identificación de requisitos y la de riesgos. Para el caso del diseño de

arquitectura, los objetivos específicos se subdividen en la derivación arquitectónica, la identificación de la arquitectura inicial y la transformación arquitectónica. Por último, en la evaluación de la arquitectura que es un eje central de análisis, se aborda la evaluación de un atributo (SAAM, ALMA, entre otros), evaluación de varios atributos (ATAM, ASAAM, QAW, etc) y la evaluación de dominio específico (FAAM).

Etapa 3. Consolidación y selección del método : Los aspectos anteriormente descritos son consolidados en la Tabla 2 y Tabla 3. Las tablas 2 y 3 se construyeron considerando las metodologías de validación de software que fueron anteriormente descritas o que fueron mencionadas en la Tabla 1; de allí se clasifican tres criterios de evaluación con el objetivo de mostrar de manera más organizada la información. En el caso de la Tabla 2 se definen los principales parámetros de calidad, tipo de actividad y levantamiento de requisitos de cada una de las metodologías de validación de software. A cada uno de los parámetros de evaluación se le asignó una sigla de identificación que se relaciona con la característica y la cual se define a lo largo de los párrafos anteriores. En el caso de la Tabla 3, en esta se presentan los requisitos de calidad que cada metodología presenta, en conjunto con el mecanismo de análisis y el objetivo general del proceso de validación de cada metodología.

Tabla 2. Parámetros de calidad, actividad y requerimientos en las metodologías analizadas.

Metodología/Parámetro	Atributo de calidad	Actividad Colectiva	Levantamiento de Requerimiento
ABD	F	SD	GE
ALMA	MO	AG	MO
ALPSM	MA	SD	GE
ASSAM	F	SD	GE
ATAM	MO	MG	
CBAM	F	MG	LPR
CBSP	F	AG	LPR
FAAM	IT	TGI	CU
MECABIC	C	TG	
PASA	D	TG	GE; CU
PRESKRIPTOR	F	TG	MO
QAW	DP, R, S, IT, MO	TG	GE
QUADRAD	S, F, R, MO	TG	GE
SAAM	F, MO	AG	GE
SACAM	MO, DP, FC, ,CP, D	SD	GE

Tabla 3. Parámetros de requisitos de calidad, mecanismos de análisis y objetivos en las metodologías analizadas.

Metodología/Parámetro	Evaluación Requisitos de Calidad	Mecanismos de análisis	Objetivos
ABD		AE, ACU	DA
ALMA	RE	AE	EA
ALPSM		AE	EA
ASSAM	AF	AE	EA
ATAM	AE	AE	EA
CBAM	AE	EC	DA
CBSP		AE, ACU	DA
FAAM	AE	AVA	EA
MECABIC		C	EA
PASA		IE, PA	EA
PRESKRIPTOR		GE	DA
QAW		AE	LR
QUADRAD	AE	AE	DA
SAAM		AE	EA
SACAM		AE	EA

Como se demuestra a lo largo del trabajo, existe una amplia asociación o dependencias de los métodos, aspectos que deben de esperarse si se consideran que estos parten como elementos de evolución de sus antecesores. Ahora, esto quiere decir que lo que muchos métodos lo que hacen es incorporar algunas características de sus antecesores y mejorar las falencias, por lo cual su diferencia radica en ocasiones en aspectos la posición de los expertos; elemento que lleva a que se trate en cierta medida de validaciones fundamentadas en métodos subjetivos.

Basado en lo anteriormente expuesto, se establece usar como método para la validación de una línea de producto de software ATAM, considerando el hecho de que es en esencia una de las pocas metodologías que se describieron que posibilitan el reuso de información junto con ABD y FAAM, considera la evaluación de varios atributos y se centraliza en el modelado de objetivos.

Etapa 4. Aplicación del método: En concordancia, a continuación, se definen las diferentes fases del proceso de validación de una línea de producto de software usando la metodología ATAM, para lo cual se presenta un ejemplo ilustrativo de las diferentes fases de aplicación de la metodología para el caso del funcionamiento de un cajero automático.
Fase 1-Presentación del Método ATAM: En el presente estudio se establece la evaluación arquitectónica del diseño del software asociado al funcionamiento de un cajero automático

bajo la consideración de las LPS.
Fase 2- Objetivos del negocio: El objetivo de negocio se relaciona con el proceso de reuso de software enfocado en la prestación del servicio de cajero automático.
Fase 3-Presentación de la arquitectura: En el ejemplo fuente de estudio, se estableció el diseño de un software que funciona como interfaz de un cajero automático, el cual permite opciones generalizables como el retiro de dinero, la consulta de saldo, el cambio de clave, la consignación en efectivo o cheque, las transferencias entre cuentas del mismo banco, la realización de pagos como impuesto predial y servicios públicos y la selección del idioma. En este orden, los actores que componen el sistema corresponden a los clientes (primario) y el sistema del banco central (externo), quien se encarga de proveer y recibir la información de las transacciones y las cuentas.

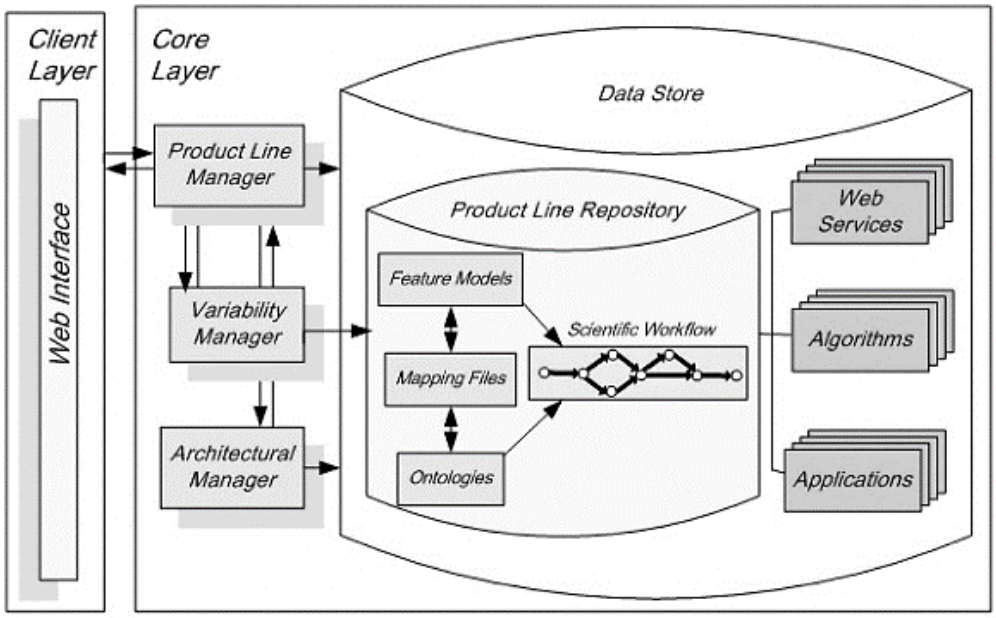

Figura 3. Identificación de la arquitectura. Fuente: (Pohl, Böckle, & Van der Linden, 2005)

De acuerdo a lo expuesto en la Figura 3, son tres elementos trascendentales que constituyen la arquitectura de sistema, correspondiendo estos al cliente, el cajero automático que incorpora la interfaz de usuario (Core Layer) y el servidor central que se encarga de controlar las peticiones por medio de la comunicación con la base de datos (Data Store).
Considerando lo dispuesto por [44], un elemento intrínsecamente asociado con la arquitectura de referencia es la evaluación de los aspectos técnicos. Para ello se alude a los modelos de variabilidad de la arquitectura que contiene las diferentes configuraciones arquitectónicas que soportará la línea de producto, las cuales se pueden describir con atributos asociados con cada variación arquitectónica, tal y como se muestra en la Tabla 4, en el que se dispone los atributos de calidad de cada una de las representaciones del modelo de variabilidad ortogonal descrito en la Figura 4 [44].

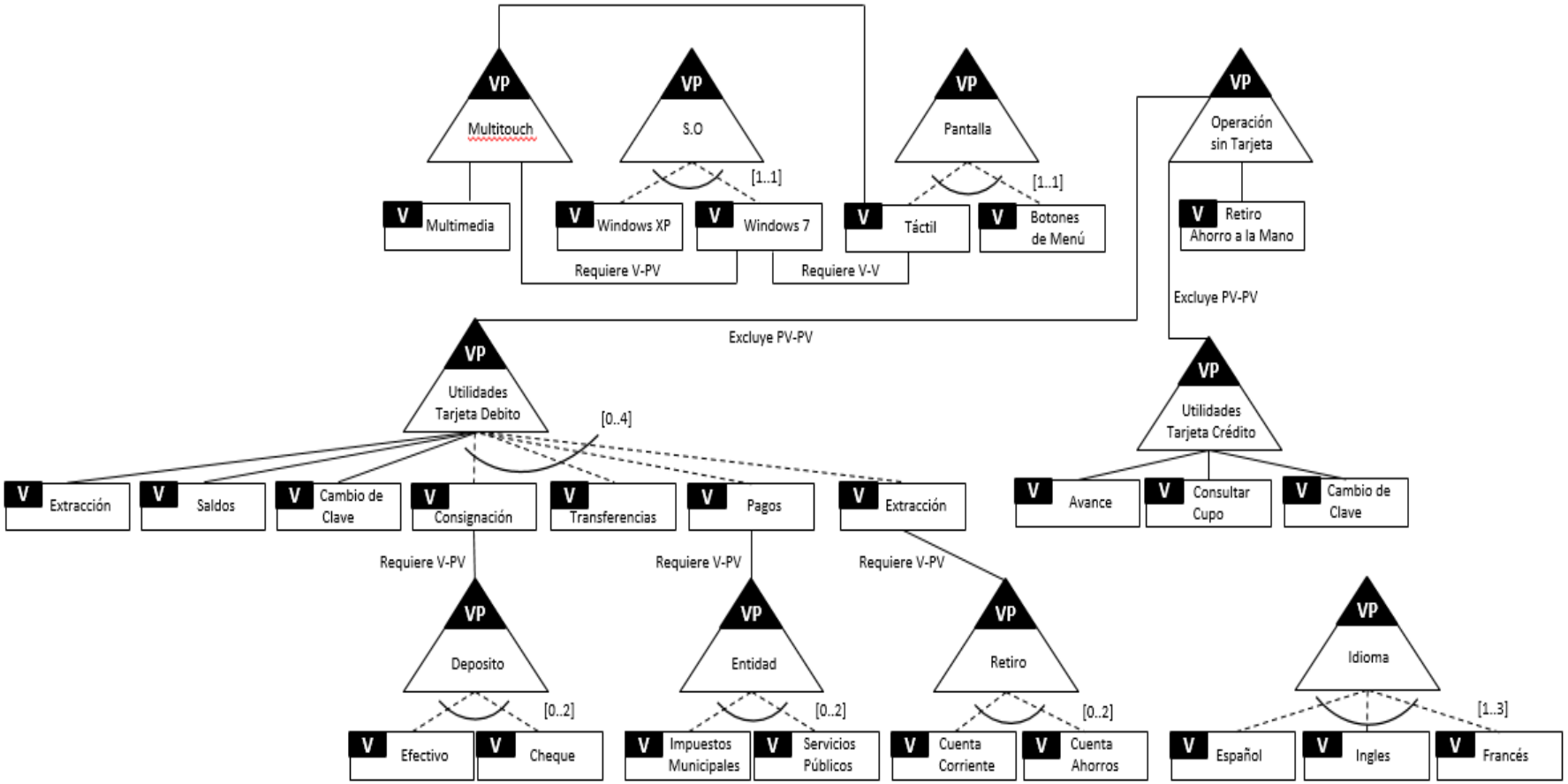

Figura 4. Representación de las características del modelo con notación en OVM (Modelo de Variabilidad Ortogonal)

A través del modelo de variabilidad ortogonal, se analiza los puntos de variación que son representados por los triángulos y sus variantes con el rectángulo. En el caso de las líneas punteadas, estas se relacionan con las variantes opcionales (que pueden llegar a ser omitidas en algunos productos), pero las líneas continuas se asocian con variantes obligatorias (presente en todos los productos) [25].

Bajo la consideración de las LPS, el cajero automático tiene nueve puntos de variación, correspondientes a Multitouch, S.O, Pantalla, Operación sin tarjeta, utilidades de tarjeta débito, utilidades de tarjeta crédito, Depósito, Entidad, Retiro e Idioma.

Fase 4- Factores del negocio: Para la evaluación de los factores de negocio, se analizan los distintos módulos, el atributo de calidad asociado, el proceso de negocio relacionado, los elementos de la arquitectura y el escenario más relevante al interior de la arquitectura.

En lo que respecta a los casos de usos propios del funcionamiento del cajero automático se destacan: CU01 – Validar tarjeta; CU02 – Retirar dinero; CU03 – Consignar dinero; CU04 – Consultar saldo; CU05 – Realizar transferencias; CU06 – Realizar pagos; CU07 – Cambiar clave; CU08 – Cambiar idioma, CU09-Entregar comprobante.

Relacionado con cada caso de uso se desprende una serie de escenarios los cuales son descritos a continuación:

- Esc 1-Utilidades de tarjeta débito/Extracción
- Esc 2-Utilidades de tarjeta débito/Saldos
- Esc 3- Utilidades de tarjeta débito/Cambio de clave
- Esc 4- Utilidades de tarjeta débito/Consignación/Depósito/Efectivo
- Esc 5- Utilidades de tarjeta débito/Consignación/Depósito/Cheque
- Esc 6- Utilidades de tarjeta débito/transferencias
- Esc 7- Utilidades de tarjeta débito/pagos/Entidad/Impuestos municipales
- Esc 8- Utilidades de tarjeta débito/pagos/Entidad/Servicios públicos
- Esc 9- Utilidades de tarjeta débito/Extracción/Retiro/Cuenta corriente
- Esc 10- Utilidades de tarjeta débito/Extracción/Retiro/Cuenta de ahorros
- Esc 11- Utilidades tarjeta crédito/avance.
- Esc 12- Utilidades tarjeta crédito/consultar cupo.
- Esc 13- Utilidades tarjeta crédito/cambio de clave
- Esc 14- Idioma/español.
- Esc 15- Idioma/inglés
- Esc 16- Idioma/francés.
- Esc 17-Operación sin tarjeta/Retiro ahorro a la mano.
- Esc 18- Multitouch/Multimedia
- Esc 19- Multitouch/Táctil
- Esc 20- S.O/Windows XP
- Esc 21- S.O/Windows 7
- Esc 22- Multitouch/Windows 7/Táctil
- Esc 23- Pantalla/Táctil
- Esc 24- Pantalla/Botones de Menú

En la Tabla 4 se presenta para cada caso de uso el atributo asociado al mismo, el proceso de negocio relacionado y el escenario en el cual se realiza el reuso de la línea de software.

Tabla 4. Aplicación del ATAM sobre la arquitectura del cajero automático

Caso de uso	Atributo asociado	Proceso de negocio relacionado	Escenario
CU01	Funcionabilidad	Pantalla Utilidad con tarjeta débito Utilidad con tarjeta débito	(1 a 13); 23 y 24
CU02	Funcionalidad Eficiencia	Retiro Utilidades tarjeta crédito Operación sin tarjeta	(9 a 11), 17
CU03	Funcionabilidad Eficiencia	Depóstio	4, 5
CU04	Funcionabilidad Eficiencia	Utilidades tarjeta débito Utilidades tarjeta crédito	2 y 12
CU05	Fiabilidad Eficiencia	Utilidades tarjeta débito	6
CU06	Fiabilidad Eficiencia	Entidad	7 y 8
CUO7	Funcionabilidad Eficiencia	Utilidades tarjeta débito Utilidades tarjeta crédito	3 y 13
CU08	Mantenibilidad Portabilidad	Idioma	14 a 16
CU09	Mantenibilidad	Pantalla Multitouch	23 a 24

Fase 5- Árbol de utilidad: Bajo la perspectiva anterior, se realiza la clasificación de cada escenario basado en los atributos de calidad. Para tal evento se consideraron dos elementos descriptores: A1-La importancia del escenario en relación al éxito del sistema y A2-El grado de dificultad para el logro del escenario. Cada uno de estos parámetros consideran tres niveles: Alto (A), Medio (M) y Bajo (B) tal y como se muestra en la Tabla 5.

Tabla 5. Asociación de puntuación por escenarios

Atributo Asociado	Escenario	Puntuación (A1,A2)
Funcionalidad	Esc1	(A, B)
	Esc 2	(M, B)
	Esc 3	(B, A)
	Esc 4	(A, M)
	Esc 5	(A, A)
	Esc 6	(B, M)
	Esc 7	(B, M)
	Esc 8	(B, M)
	Esc 9	(A,A)
	Esc 10	(A,A)
	Esc 11	(A,A)
	Esc 12	(A, M)
	Esc 13	(B, A)
	Esc 17	(M, M)
	Esc 23	(M,M)
	Esc 24	(M, M)
Eficiencia	Esc 9	(M,A)
	Esc 10	(M,A)
	Esc 11	(M, M)
	Esc 17	(M, M)
	Esc 5	(M, A)
	Esc 4	(M, M)
	Esc 2	(B, B)
	Esc 12	(M, B)
	Esc 6	(B, B)
	Esc 7	(B, B)
	Esc 8	(B, B)
	Esc 3	(B, M)
	Esc 13	(B, M)
Mantenibilidad	Esc 14	(B,B)
	Esc 15	(B,B)
	Esc 16	(B,B)
	Esc 23	(M,M)
	Esc 24	(M, M)
Portabilidad	Esc 14	(B,B)
	Esc 15	(B,B)
	Esc 16	(B,B)

Fase 6-Priorización de escenarios: Basados en los elementos anteriormente dispuestos, se procede a realizar una priorización de escenarios; ello basado principalmente en atributos como la funcionalidad (relacionado a su vez con la seguridad) y la eficiencia (tiempo de respuesta) (Estas definiciones se encuentran en el marco teórcio), tal y como se muestra en la Tabla 6.

Tabla 6. Priorización de escenarios

Módulo o Punto de Variación VP	Prioridad
Lectura de tarjeta	Alta
Retiro de Dinero	Alta
Consignación	Media
Entrega de comprobante	Media
Cambio de Clave	Media
Pagos	Baja
Transferencias	Baja
Consultas	Baja
Selección de idioma	Baja

Fase 7-Informe: De acuerdo a los resultados expuestos del proceso investigativo del software de un cajero automático, se establece que la priorización del diseño del mismo se debe centralizar en dos elementos claves: los procesos de lectura de tarjeta que se requiere en la mayoría de los procesos, el retiro de dinero que encarece en cierta medida los parámetros de seguridad y el uso de herramientas de autenticidad. Una menor priorización del diseño se debe enfocar en los procesos de consignación, la entrega de comprobantes y el cambio de clave, aspectos que en cierta medida son menos frecuentes en comparación a los primeros.

CONCLUSIONES

Respecto a las líneas de producto de software, son dos los parámetros fundamentales que los caracterizan, el primero es que se enfocan en el desarrollo de un conjunto de productos con una características en particular y el segundo es la satisfacción de las necesidades explicitas del mercado por medio de activos reutilizables.

Ahora, una de las complejidades sobre las líneas de producto de software radica en los procesos de validación del software, puesto que no todas las metodologías existentes son aplicables a estos particulares diseños. En dicha medida, el análisis de las distintas metodologías de validación de software se centralizaron en la determinación de los atributos de calidad abordados, el requerimiento del tipo de actividad colectiva, los aspectos sobre el levantamiento de requisitos, los requisitos de calidad, los mecanismos de análisis y los objetivos específicos que define cada una de estas.

De las metodologías descritas se tiene que no todas son aplicables a las diferentes ramas del diseño, como caso explícito en la línea de productos de software (LPS) en donde los métodos ABD, FAAM y ATAM son de los pocos que se relacionan con los procesos de reuso, orientados en sí al desarrollo del método y al uso del conocimiento adquirido respecto a los mecanismos y soluciones arquitectónicas.

En el caso explícito de selección de ATAM como metodología para la ejemplificación del proceso de validación de una línea de producto de software, se debe que se trata en sí de las pocas que se enfocan en el reuso de la información y considera la evaluación de diferentes

atributos, centrando su interés en el modelado de objetivos. Para el caso del ejemplo con el cajero automático se abordaron seis fases: presentación del modelo, establecimiento de los objetivos del negocio, la presentación de la arquitectura usando connotación OVM (Modelo de Variabilidad Ortogonal), factores del negocio, árbol de utilidad a partir de los cuales finalmente se llega a la priorización de escenarios en función de los atributos de calidad. Como resultado de dicho proceso, fueron cuatros los tipos de atributos de calidad asociados al proceso evaluativo: funcionalidad, eficiencia o desempeño, mantenibilidad y la portabilidad o facilidad de uso. De los módulos o puntos de variación VP analizados de acuerdo a los diferentes escenarios, se estableció que la priorización se debe dar sobre la lectura de las tarjetas y los retiros de dinero.

Para el tema de empresas de desarrollo de software se puede implementar las metodologías de validación de líneas de producto de software con el objetivo de reutilizar los elementos de diseño ya propuestos, de manera que se reducen el uso de recursos; los cuales deben estar acompañados indiscutiblemente de un proceso de validación como el propuesto en este artículo.

Para futuros trabajos se recomienda evaluar el proceso no sólo de evaluación de metodologías sino describir desde el punto de partida que corresponde al diseño de una línea de producto de software la sistemática considerando herramientas como ATAM y comparándolas con otras existentes, de manera que se le permita al lector evaluar el grado de evolución e implementación de la misma.

AGRADECIMIENTOS

Le agradezco a Dios por guiarme y acompañarme a lo largo de mi vida, a todas las personas que con su conocimiento científico y humano colaboraron en la realización de este trabajo de investigación, en especial a mi tutor y director de tesis al profesor Mauricio Toro Bermúdez, los jurados Elizabeth Suescún Monsalve y Paola Vallejo Correa por sus aportes y comentarios.

REFERENCIAS

[1] L. Pauta y S. Moscoso, «Verificación y validación de software,» *Revista Killkana Técnica,* vol. 1, nº 3, pp. 25-32, 2017.

[2] R. Villegas, Ciclo de validación de una aplicación informática, Universitat oberta de Catalunya, 2013.

[3] L. Gimson, Metodologías ágiles y desarrollo basado en conocimiento, Universidad Nacional de La Plata, 2012.

[4] Uzcátegui, E., Ortega, D., & Delgado, D. (2009). Metodologías de desarrollo para sistemas de tiempo real. Un Estudio Comparativo. ScieloVenezuela, 13(50), 059-066..

[5] Dasso, A., & Funes, A. (July, 2006). Verification, Validation and Testing in Software Engineering. United States of America: Integrate book technology..

[6] Universidad de la República, Verificación y validación, Montevideo: Facultad de Ingeniería, 2015.

[7] R. Puerta, Soporte de trazabilidad en el desarrollo de líneas de producto de software, Madrid: Universidad Politécnica de Madrid, 2011.

[8] Northrop, L., & Clements, P. (2009). Software Engineering Institute (Carnegie Mellon).

[9] G. Solarte, L. Múñoz y B. Arias, «Modelos de calidad para procesos de software,» *Scientia Et Technica,* vol. 15, nº 42, pp. 375-379, 2009.

[10] C. Gaitán, «Líneas de Productos Software:Generando Código a Partir de Modelos y Patrones,» Scientia et Technica, vol. 22, nº 2, pp. 175-183, 2017..

[11] P. Clements, R. Kazman y M. Klein, Evaluating Software Architectures: Methods and Case Studies, Addison Wesley, 2002.

[12] Ó. Díaz, Líneas de producto de software, Universidad del País Vasco, 2010.

[13] D. Krueger, Software Product-Line Engineering, IEEE Transactions on Software Engineering, 2006.

[14] F. Gomma, An Evaluation of Aspect-Oriented Programming as a Product Line Implementation Technology, International Conference on Software Reuse, 2004.

[15] R. Casallas, Líneas de Producto de Software, Bogotá: Departamento de Sistemas y Computación, Universidad de los Andes, 2010.

[16] SEI, Software Engineering Institute, 2012.

[17] J. Montilva, Desarrollo de Software Basado en Líneas de Productos de Software, Venezuela: Universidad de los Andes, 2010.

[18] M. Laguna, «Desarrollo de Líneas de Productos: un Caso de Estudio en Comercio Electrónico,» *7th Latin American and Caribbean Conference for Engineering and Technology ,* pp. 2-5, 2009.

[19] Bosch y J, Design & Use of Software Architectures. Adopting and Evolving a Product-Line Approach., Addison-Wesley, 2000.

[20] M. Laguna, O. López y B. González, Gestión de la Variabilidad en Líneas de Productos, Instituto Tecnológico de Costa Rica, 2012.

[21] J. Ros, Instituto Tecnológico de Costa Rica, Costa Rica: Universidad de Murcia, 2009..

[22] Roger S. Pressman, "Ingeniería de Software. Un enfoque práctico", España, 1998.

[23] J. Drake y P. López, verificación y validación, Universidad de Cantabria, 2008.

[24] N. Juristo, A. Moreno y S. Vegas, Técnicas de evaluación de software, Madrid: Universidad Politécnica de Madrid, 2006.

[25] A. Delgado y A. Castro, «Evaluación de Arquitecturas de Software con ATAM (Architecture Tradeoff Analysis Method): un caso de estudio,» *Conference: Conference: VI Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento,* pp. 151-159, 2007.

[26] International Organization for Standardization, ISO 9001 Sistemas de calidad. Requisitos, 2015.

[27] J. Chaparro y J. Romero, Diseño de una arquitectura de software para el análisis de sentimientos aplicado a las opiniones de usuarios en Twitter sobre los servicios ofrecidos en el sistema de salud en Colombia, Santiago de Cali: Universidad de San Buenaventura Colombia, 2018.

[28] R. Kuhn, T. Chandramouli y R. W. Butler, «Cost Effective Use of Formal Methods in Verification and Validation,» *Foundations 02, a Workshop on Modeling and Simulation Verification and Validation for the 21st Century,* pp. 106-144, 2002.

[29] R. Kazman, P. Kruchten, R. L. Nord y J. E. Tomayko, «Integrating Software-Architecture-Centric Methods into the Rational Unified Process,» *Carnegie Mellon Software Engineering Institute,* 2004.

[30] J. Medina, Metodología y Herramientas UML para el Modelado y Análisis de Sistemas de Tiempo Real Orientados a Objetos, Universidad de Catabria, 2008.

[31] E. Diez, «Aseguramiento de la calidad en la construccion de sistemas basados en el conocimiento,» 2003.

[32] F. Scalone, «Estudio comparativo de los modelos y estandares de calidad de software,» *Universidad Tecnológica Nacional,* 2006.

[33] Ravindranath Pandian, C. (2003). Software Metrics: A Guide to Planning, Analysis, and Application. Taylor & Francis Group..

[34] Serna, E. Capítulo 1. Revista Virtual Universidad Católica. Editorial Eafit Pág. 180-184.

[35] M. A. Laguna, B. G. Baixauli y O. López, «Gestión de la Variabilidad en Líneas de Productos,» *In Pro c. of Conferencia Latinoamericana de Informática (CLEI'07,* 2007.

[36] L. Dobrica y E. Niemela, «A strategy for analysing product line software architectures,» *TECHNICAL RESEARCH CENTRE OF FINLAND,* nº 427, p. 124, 2000.

[37] T. J. Dolan, Architecture Assessment of Information-System, Ierland: Technische Universiteit Eindhoven, 2001.

[38] T. Kim, I. Ko, S. Kang y D. Lee, «Extending ATAM to assess product line architecture,» *In 8th IEEE International Conference on Computer and Information Technology,* pp. 790-797, 2008.

[39] J. Á. Pastor F., «Evaluación y desarrollo incremental de una arquitectura software de referencia para sistemas de teleoperación utilizando métodos formales,» *Universidad Politécnica de Cartagena, Dpto. de Tecnologías de la Información y de la Comunicación.,* 2002.

[40] P. Bengtsson y J. Bosch, Architecture Level Prediction of Software Maintenance, Amsterdam: in Proceedings of Third European Conference on Software Maintenance and Reengineering, 1999, pp. 139-147.

[41] M. R. Barbacci, R. Ellison, A. J. Lattanze, J. Stafford, C. Weinstock y W. Wood, Quality Attribute Workshops (QAWs), Third Edition (CMU/SEI-2003-TR-016, ADA418428) ed., Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003.

[42] L. Williams y C. Smith, Performance Evaluation of Software Architectures, Santa Fe: Proceedings of the Workshop on Software and Performance (WOSP98), 1998.

[43] P. Bengtsson y J. Bosch, Scenario-based Architecture Reengineering, 5th International Conference on Software Reuse, 1998.

[44] R. Mazo, M. Toro y L. Cobadela, Guía para la adopción industrial de líneas de productos de software, Medellín: Colección Académica EAFIT, 2018.

[45] F. Losavio y C. Guillen, «Comparación de métodos para la arquitectura del software: Un marco de referencia para un método arquitectónico unificado,» *Revista de la Facultad de Ingenieria ,* vol. 25, nº 1, pp. 71-87, 2010.

[46] A. Dolores y L. Silva, «Arquitectura de software para el sistema de visualización médica Vismedic,» *Revista Cubana de Informática Médica,* vol. 8, nº 1, pp. 75-86, 2016.

[47] A. Orellana y V. Velasteguí, Evaluación de la arquitectura de software de aplicaciones de producción, Quito: Universidad Politécnica Nacional, 2007.

[48] P. Shanmugapriya y R. Suresh, «Software Architecture Evaluation Methods – A survey,» *International Journal of Computer Applications,* vol. 49, nº 16, pp. 19-26, 2012.

[49] T. Mugurel, K. Dieter y O. Henk, Scenario-Based Software Architecture Evaluation Methods: An Overview, The Netherlands: Department Software Architectures, Technical University Eindhoven, 2008.

[50] L. Williams y C. Smith, «PASASM: A Method for the Performance Assessment of Software Architectures,» *Software Engineering Research and Performance Engineering Services.,* 2002.

[51] C. Stoermer, C. Verhoef y F. Bachmann, SCAM: The Software Architecture Comparison Analysis Method, 2003.

[52] S. Thiel, Framework to Improve the Architecture Quality of Software-Intensive Systems, Alemania: Universität Duisburg-Essen, 2005.

[53] M. Babar y I. Gorton, Comparison of scenariobased software architecture evaluation methods, Australia: Nacional ICT Australia Ltd. and University of New South Wales, 2004.

[54] A. Grimán, M. Pérez, L. Mendoza y F. Losavio, «Feature Analysis for Architectural Evaluation Methods,» *Journal of Systems & Software,* vol. 79, nº 6, pp. 871-888, 2006.

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

[56] J. Franco, Evaluación y desarrollo incremental de una arquitectura software de referencia para sistemas de teleoperación utilizando métodos formales, Cartagena, España: Universidad Politécnica de Cartagena, 2002.

[57] J. Miranda, Arche: Evaluación de arquitecturas y toma de decisiones. Aplicación al análisis de arquitecturas de software en el sector aeronaútico, Madrid: Universidad Nacional de Educación a Distancia, 2015.

[58] E. Albertos, Arquitecturas software para microservicios: una revisión sistemática de la literatura, Madrid: Universidad Politécnica de Madrid, 2018.

[59] Chung-Horng, Lung; Bot, Sonia; Kalai, Kalaichelvan; Rick, Kazman, Toronto: Centre for Advanced Studies Conf, 1997, pp. 144-154.

[60] W. Husnain y S. Ahmed, «A LITERATURE REVIEW AND RECOMMENDATIONS ON SOFTWARE ARCHITECTURE EVALUATION,» *International Journal of Reviews in Computing,* vol. 10, pp. 28-35, 2009.