UNIVERSIDAD CENTROCCIDENTAL
"LISANDRO ALVARADO"

PROCESO PARA LA INGENIERÍA DE LA APLICACIÓN EN LÍNEAS DE PRODUCTO DE SOFTWARE

SOL MIRELLA OVALLES HERNÁNDEZ

Barquisimeto, 2015

UNIVERSIDAD CENTROCCIDENTAL “LISANDRO ALVARADO”
DECANATO DE CIENCIAS Y TECNOLOGÍA
COORDINACIÓN DE POSTGRADO

PROCESO PARA LA INGENIERÍA DE LA APLICACIÓN EN LÍNEAS DE PRODUCTO DE SOFTWARE

Trabajo presentado para optar al grado de
Magister Scientiarum en Ciencias de la Computación

Por: SOL MIRELLA OVALLES HERNÁNDEZ

Barquisimeto, 2015

PROCESO PARA LA INGENIERÍA DE LA APLICACIÓN EN LÍNEAS DE PRODUCTO DE SOFTWARE

Por: SOL MIRELLA OVALLES HERNÁNDEZ

Trabajo de grado aprobado

_____	_____
(Jurado 1)	**(Jurado 2)**

_____

(Jurado 3)

Barquisimeto, ___ de _____ de 2015

Dedicatoria

A Dios Todopoderoso, por permitirme ser y estar.

A mi Hijo, que es la bendición más grande que Dios me ha concedido.

Y a los que ya no están, porque gracias a ellos puedo estar aquí.

ÍNDICE GENERAL

LISTA DE FIGURAS

LISTA DE TABLAS

UNIVERSIDAD CENTROCCIDENTAL "LISANDRO ALVARADO"
DECANATO DE CIENCIAS Y TECNOLOGÍA
COORDINACIÓN DE POSTGRADO

PROCESO PARA LA INGENIERÍA DE LA APLICACIÓN EN LÍNEAS DE PRODUCTO DE SOFTWARE

Autor: Ing. Sol Mirella Ovalles Hernández
Tutor: MSc. Edison Sira Carreño

RESUMEN

La reutilización en el ámbito de la ingeniería de software, apunta a mejoras en la productividad, reducción de tiempo y costos en los procesos involucrados; uno de los enfoques utilizados para ello es líneas de producto de software (LPS), el cual ha tenido éxito en el campo empresarial. Sin embargo, metodológicamente se observa que en alto grado LPS aun está en desarrollo en cuanto a las normas de calidad y especificidad en sus disciplinas, en especial las correspondientes a la ingeniería de la aplicación, en tal sentido, se considera que el asunto de la reutilización sigue vigente. El objetivo principal de esta investigación es proponer un proceso para la ingeniería de la aplicación con características de calidad (InALPro), abarcando las disciplinas de análisis y diseño de la aplicación. El estudio fue abordado bajo la modalidad de proyecto especial, apoyado en el paradigma investigativo de ciencia del diseño, se utilizó el enfoque de extensión de la ingeniería de método situacional para hacer extender el proceso InDoCaS, esto permitió obtener nuevos artefactos y actividades para la ingeniería de la aplicación, luego ser representadas con la notación SPEM. Con InALPro se logró definir una mejor forma de gestionar los requisitos, planificar la reutilización de los artefactos de la ingeniería del dominio, precisar normas de calidad y arquitectura de los productos de la LPS. Finalmente, se aplica InALPro en el dominio eLearning en la Universidad Politécnica Territorial del estado Portuguesa "Juan de Jesús Montilla", para ejemplificar la propuesta.

Palabras claves: reutilización de artefactos de software, línea de producto de software, InDoCaS, ingeniería de la aplicación, calidad de software, metodología de software, ciencia del diseño, ingeniería de método situacional.

INTRODUCCIÓN

La reutilización de software ha evolucionado desde sus inicios en la década de los sesenta, ésta consiste en la creación de aplicaciones usando activos de software existentes, estos activos no se refieren sólo a código, son todos los artefactos generados en el ciclo de vida del producto, por ejemplo requisitos, modelos de características, arquitecturas, funciones y entre otras. Este enfoque ha surgido como alternativa a la producción de aplicaciones individuales, las cuales representan mayor esfuerzo, costos y tiempo para ser producidas, debido a que cada sistema es construido casi desde cero; de igual forma presentan mayor complejidad en el mantenimiento y corrección de errores, originado por la naturaleza cambiante de los requisitos y la atención individualizada que cada sistema demandará.

En este propósito, ha surgido el enfoque de líneas de producto de software (LPS), el cual se fundamenta en el conocimiento de un dominio para generar una arquitectura que optimiza los aspectos comunes y gestiona los variables. El enfoque de LPS tiene dos fases principales, la ingeniería de dominio y la ingeniería de la aplicación, la primera construye la infraestructura común y la segunda, da forma específica a los productos de la línea. Las LPS apuntan a generar productos de calidad en menor tiempo, disminuyendo costos y errores en el proceso de desarrollo, influenciando positivamente la satisfacción de los usuarios y la productividad organizacional, al respecto el SEI[1] plantea que empresas como Hewlett-Packard, Cummins, Inc., General Motors, CelsiusTech, Rockwell-Collins, Motorola, Philips, Nokia, Boeing, Raytheon, y Salion, Inc., entre otras, han adoptado este paradigma en el dominio de aplicación de varios de sus productos.

La ingeniería de software sugiere que acoger un método para el desarrollo de software es una buena práctica, en este sentido Camarero, M. (2014), afirma que las metodologías para el desarrollo se han convertido en herramientas de vital importancia que influencian el éxito del producto. Hoy por hoy, existen varios

[1] Software Engineering Institute. Carnegie Mellon University.

procesos que guían el desarrollo bajo el enfoque de LPS, según Montilva, J. (2006) estos son el modelo TWIN, WATCH, ESPLEP y el modelo del SEI; en los anteriores se observan similitudes y diferencias, pero coinciden en la debilidad de no ser específicos en las actividades de las disciplinas de la ingeniería de la aplicación, es decir, no se detallan los pasos que deben cumplirse para alcanzar los objetivos de cada disciplina.

De igual manera, carecen de aspectos de calidad en sus pautas, lo que ocasiona problemas al evaluar el proceso y producto resultante. Así lo afirma Montagut, S. (2009), el aseguramiento de la calidad en líneas de productos es aún más importante que en el desarrollo de software tradicional, debido a que un error o decisión de diseño inadecuada podrían propagarse a varios productos de la línea.

No obstante, el proceso para la Ingeniería de Dominio basado en Calidad de Software (InDoCaS), es detallado en sus disciplinas y único de los estudiados en contemplar aspectos de calidad en su práctica (características de la norma ISO/IEC[2] 25010), también se puede mencionar que ha sido implementado en otras investigaciones como son las realizadas por Rivero, L. (2011), Gómez, E. (2012), Perez, J. y Sánchez, I. (2012) y Duarte, D. (2012). Una limitación de este enfoque es que no contempla actividades de las disciplinas de la ingeniería de la aplicación.

Varios de los problemas descritos, se presentan en la Universidad Politécnica Territorial del estado Portuguesa "Juan de Jesús Montilla" [3] en Acarigua, donde hay cierta insatisfacción de usuarios de las aplicaciones de software, esto motivado a que algunos sistemas, o parte de ellos, no cubren plenamente las necesidades de información existentes, algunas aplicaciones presentan errores y ausencia de documentación y los nuevos proyectos son entregados con retraso. Por otro lado, en la unidad de sistemas, los desarrolladores plantean que no utilizan metodologías ni estándares de calidad para el desarrollo, algunas técnicas de reutilización son inadecuadas, el personal es escaso y el trabajo abundante.

---

[2] ISO/IEC: International Organization for Standadization/International Electrotechnical Commission

[3] UPTP (www.iutep.tec.ve)

La situación anterior, se evidenció mediante la observación directa y entrevista a usuarios y desarrolladores de la UPTP, donde se puede inferir que el desarrollo de software es tradicional, enfocado en resultados y sin control del proceso, por tanto, se considera que parte de la solución a los problemas de la unidad de sistemas de la UPTP, es la implementación de una metodología de desarrollo basada en reutilización con características de calidad, para lo cual el paradigma de LPS promete ser uno de los más adecuados.

Por las consideraciones anteriores, se propone un proceso para la ingeniería de la aplicación en LPS con enfoque de calidad, denominado en lo sucesivo InALPro, dicho proceso se obtiene a través del empleo del enfoque de extensión de la ingeniería de método situacional, lo que permite definir los artefactos, actividades y actores de las disciplinas de análisis y diseño, dejando la implementación para trabajos futuros, adicionalmente, este proceso está enmarcado dentro de las normas de calidad ISO/IEC 25010, lo que le concede distinción respecto a modelos similares. Para la representación del proceso InALPro se utiliza la notación SPEM[4], ideal para documentar trabajos de este género.

Posteriormente, se aplica InDoCaS e InALPro sobre el dominio eLearning en la UPTP "Juan de Jesús Montilla" para ejemplificar la implementación de una LPS. El uso de estos dos procesos genera un método para LPS con enfoque de calidad. Con esta investigación se espera enriquecer el conocimiento académico sobre las LPS en especial su fase de ingeniería de la aplicación, como también el uso de la ingeniería de método situacional y la ciencia del diseño como enfoque investigativo.

El presente estudio está estructurado en cinco (5) capítulos:

El Capítulo I, contempla el planteamiento del problema, donde se explican las causas que dieron origen a esta investigación, los objetivos que se persiguen y el contexto dentro del cual se delimita la temática estudiada. Además, se presenta la justificación e importancia, así como el alcance y limitaciones del mismo.

[4] Software Process Engineering

En el Capítulo II, se presentan los antecedentes recientes relacionados al tema investigado, y las bases teóricas que fundamentan el estudio, como lo son la reusabilidad, las líneas de producto de software, los modelos de desarrollo relacionados a las LPS, la ingeniería de la aplicación, calidad de software, la ingeniería de método situacional, la ciencia del diseño, entre otros.

En el Capítulo III, se especifican los aspectos metodológicos como lo son: naturaleza, diseño y procedimiento de investigación.

En el Capítulo IV, se plantea la propuesta y la ejemplificación de su uso en el dominio eLearning en la UPTP "Juan de Jesús Montilla".

El Capítulo V, se encuentran las conclusiones y recomendaciones.

CAPITULO I

EL PROBLEMA

Planteamiento del Problema

El cambio es inherente al software, sino se adapta caduca, de acuerdo a Lehman, M.M. (1996) el *cambio es continuo*, de lo contrario el software llega a ser menos satisfactorio, es decir, el software debe estar a la par con la cambiante dinámica organizacional, la necesaria satisfacción del usuario y la disminución de la incertidumbre que rodea el proceso. A este tenor, Jones, C. (1996, Febrero) considera que los proyectos de software cambian rápidamente, en consecuencia la gestión de cambio eficiente es un reto para la industria del software.

Esta naturaleza cambiante del software, demanda la evolución de métodos de desarrollo que faciliten la adaptación a nuevas situaciones y diversidad de requisitos, en este sentido, la ingeniería del software ha proporcionado modelos o enfoques de desarrollo variados, así lo afirma Pressman, R., (2010), el proceso de software se ha aplicado durante años para ordenar y estructurar el desarrollo, de este modo han surgido modelos en cascada, incrementales, evolutivos, basados en componentes, proceso unificado, entre otros. Sin embargo, todos pretenden la misma finalidad: el desarrollo de software de calidad en el menor tiempo y costo posible.

Los enfoques de desarrollo centrados en un solo producto generan mayor esfuerzo, costos y tiempo en su diseño, lo que es poco rentable para la organización que lo utiliza, igualmente, la variabilidad de los requisitos exige enfoques versátiles a las nueva necesidades, en tal sentido, surgen nuevas disciplinas, técnicas, métodos, paradigmas que faciliten la producción de software de forma eficiente.

Entre estos paradigmas, se encuentran los orientados a reutilización, al respecto Sametinger, J. (1997, p. 10) cita a McIlroy M.D. (1968) quien planteaba beneficios de la reutilización desde esos días. Montilva, J., Arapé N. y Colmenares J. (2003) definen “La reutilización de software es un proceso de la Ingeniería de Software que conlleva al uso recurrente de activos de software en la especificación, análisis, diseño,

implementación y pruebas de una aplicación o sistema de software". Estos activos han evolucionado y hoy en día son todos los productos derivados del desarrollo de software que pueden ser utilizados nuevamente.

En ese mismo sentido, Sommerville, I., (2005), considera formas potenciales de soportar la reutilización: Patrones de diseño, Reutilización basada en generadores, Marcos de trabajo de aplicaciones y Reutilización de sistemas de aplicaciones, este último enfoque se divide a su vez en Reutilización de productos Commercial Off-The-Shelf (COTS) y líneas de producto de software, esto se ilustra en la Figura 1. Es decir, existen variados enfoques que abordan la reutilización, cada uno desde una perspectiva particular, con fortalezas y debilidades diferentes, lo que deja a criterio de cada organización la elección del más conveniente según sus necesidades.

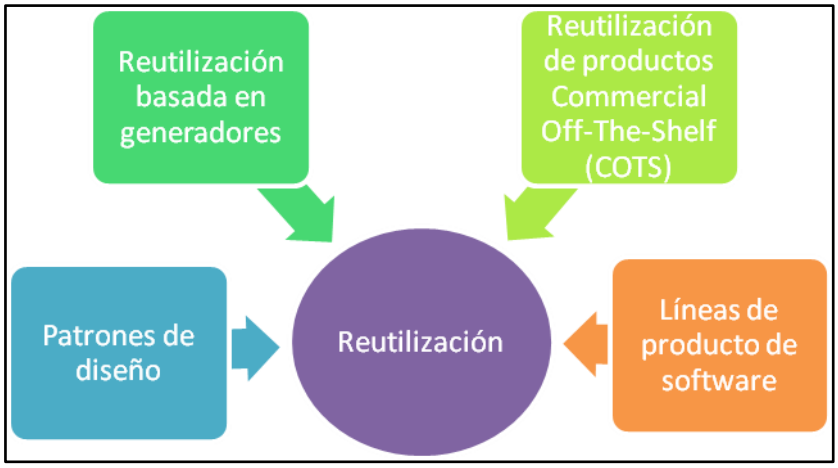


Figura 1. Formas potenciales de reutilización.
Fuente: Sommerville, I., (2005). Gráfico: Autora de la investigación (2015).

De los enfoques anteriores, el de LPS es uno de los más promisorios, Clements, P. y Northrop, L. (2001) definen una línea de producto de software como un conjunto de sistemas de software, que comparten un conjunto común y gestionado de aspectos que satisfacen las necesidades específicas de un segmento de mercado o misión y que son desarrollados a partir de un conjunto común de activos fundamentales de software de una manera prescrita.

Diversos autores como Clements, P., y Northrop, L., (2002), Pohl, K., Böckle, G., y Linden, F. (2005) y Piattini, J. y Garzás, J. (2007) coinciden en que una LPS está constituida por dos procesos esenciales: la ingeniería de dominio y la ingeniería

de aplicación, la primera es responsable de generar los elementos comunes (reutilizables), y la segunda, no menos importante, da forma concreta a los productos (software) que los clientes demandan, ambos procesos revisten gran importancia y requieren de elementos de calidad de proceso y de producto.

Las LPS formalmente planteadas, son relativamente nuevas, surgieron en la década de 1990 de aproximaciones realizadas de ingeniería de dominio, pero estas no alcanzaron las metas propuestas. Posteriormente proyectos como PuLSE (Bayer et al., 1999), KobrA (Atkinson et al., 2000), CoPAM (America et al., 2000), Feature-Oriented Reuse Method (FORM), Engineering Software Architectures, Processes and Platforms (ESAPS[5]), Concepts to Application in system-Family Engineering (CAFE[6]) y Fact-based Maturity through Institutionalisation, Lessons-learned and Involved Exploration of System-family engineering (FAMILIES[7]) representan aportes muy significativos en la evolución del enfoque de LPS, en estos trabajos se observan similitudes y diferencias entre ellos, pero es notable un mayor énfasis en las etapas de la ingeniería de dominio y en aspectos organizacionales, siendo menos estudiado, explícito, o ausente el proceso de la ingeniería de la aplicación.

Es posible generalizar que la ingeniería de dominio ha sido más estudiada y difundida, en principio por gestarse como modelo independiente y luego como parte fundamental de LPS y de otros enfoques. Contrariamente, de la ingeniería de la aplicación se encuentran menos estudios que referencien las actividades y artefactos de sus disciplinas, hay diversidad de opiniones y puntos de vista encontrados.

Sin embargo, de acuerdo a Bermejo, J., (2007) las LPS apuntan a beneficios tales como disminución en el esfuerzo de desarrollo y mantenimiento, mejoras en calidad y tiempo de entrega del producto. Respecto a la calidad, Piattini, J. y Garzás, J. (2007) afirman que resulta fundamental evaluar la calidad durante toda la fabricación del software, tanto desde el punto de vista de la calidad del producto como del proceso software, para asegurar al usuario que la aplicación entregada

[5] http://www.sse.uni-due.de/en/projects/esaps
[6] http://www.sse.uni-due.de/en/projects/d-cafe
[7] http://www.esi.es/Families/

cumple con lo solicitado. Por tanto, es preciso prestar atención al proceso y los productos de la línea, ya que pequeños errores pueden generar grandes problemas a futuro en el sistema. Esto se evidenciará la insatisfacción de los usuarios y posteriormente en el menoscabo de la productividad organizacional.

Según Camarero, M. (2014), algunas metodologías para LPS existentes en el mercado son "el modelo TWIN, el modelo WATCH, el modelo ESPLEP o el modelo SEI", en estas también se observa menos especificidad en las disciplinas de la ingeniería de la aplicación, es decir, escaso detalle en las actividades y artefactos de cada una, situación que dificulta la gestión del proyecto. De igual forma, las normas de calidad que se aplican al proceso y producto software son menos explicitas, esto impide disponer un marco que referencie sus bondades, asunto de suma importancia en nuestros días, así se afirma en el Portal ISO 25000[8] "La calidad del producto, junto con la calidad del proceso, es uno de los aspectos más importantes actualmente en el desarrollo de software".

Al mismo tiempo Rivero, L. (2011) propone que también se puede usar el proceso InDoCaS de Canelón, R. (2010) como método para LPS, lo anterior por distinguirse de otras propuestas, al ser minucioso en las actividades y artefactos que constituyen sus disciplinas y por contemplar el enfoque de calidad de la norma ISO/IEC 25010.

En cuanto a la producción de software a nivel nacional Zabala, S. (2013) y Montilva, J., Montilva W., y Barrios J. (2011) citan una investigación de Rivero, M., Montilva, J., Barrios, J., Murúa, M. y Granados, G. (2009) donde concluyen que la industria venezolana del software "está en pleno proceso de desarrollo y que, como tal, adolece, en la mayoría de los casos, de la madurez necesaria para producir software con los altos niveles de calidad", lo anterior es reflejado en "la poca formalización y estandarización de modelos de procesos y métodos de desarrollo que esta industria emplea". Igualmente afirman que los "modelos y métodos no están bien establecidos en la mayoría de las empresas, así como no está bien establecido el uso

[8] http://iso25000.com/

de técnicas y herramientas que apoyen todo el proceso de desarrollo". Es decir, la investigación en el área de modelos y procesos de desarrollo es un tema relevante en nuestro país.

En la UPTP "Juan de Jesús Montilla", se utiliza software diseñado en la institución y por terceros, en ambos casos los usuarios manifiestan problemas como retraso en la entrega de nuevas aplicaciones, notable concurrencia de defectos del software, los sistemas no ejecutan todos los procesos demandados, existencia de sistemas que requieren migración a nuevas plataformas. En vista de los planteamientos anteriores, en la unidad de sistemas se detectó la ausencia de un método de desarrollo de software, la escasez de normas de calidad en el proceso y al producto, la incorrecta reutilización de software para nuevas necesidades, personal de desarrollo escaso y el volumen de trabajo abundante, estos dos últimos aspectos se escapan del alcance de este estudio, pero se considera que los aspectos metodológicos pueden ser abordados.

De todo lo anterior, se plantea un proceso de software para la ingeniería de la aplicación en líneas de producto de software (denominado en lo sucesivo InALPro), enmarcado dentro de un enfoque de calidad; para formular el proceso InALPro se aplica el enfoque de extensión de la ingeniería de método situacional para identificar las actividades, artefactos y actores de las disciplinas de análisis y diseño de la ingeniería de la aplicación en LPS. El enfoque de extensión consiste en tomar un método base y complementarlo con partes de otros métodos donde haga falta.

Para la aplicación de InALPro dentro del ámbito de las LPS, es necesario contar con un técnica que genere los artefactos de la ingeniería de dominio, para lo cual se escogió el proceso Ingeniería del Dominio basado en Calidad de Software (InDoCaS), estos dos procesos se aplicaran al dominio eLearning para ejemplificar la implementación de una línea de producto de software en la unidad de sistemas de la UPTP; el planteamiento de una plataforma elearning es uno de los proyectos pendientes en dicha unidad, y con lo cual se estudiará el proceso de LPS.

Para el desarrollo de la presente investigación es preciso satisfacer las siguientes interrogantes:

¿Cómo se caracterizan las disciplinas del proceso de ingeniería de la aplicación en líneas de producto de software?

¿Cuáles son las actividades, artefactos y actores de las disciplinas del proceso de ingeniería de la aplicación en líneas de producto de software?

¿Qué características de calidad se deben contemplar en las disciplinas del proceso de ingeniería de la aplicación en líneas de producto de software?

¿Cómo representar las actividades y artefactos de las disciplinas de la ingeniería de la aplicación en líneas de producto de software (InALPro)?

¿Cómo es la aplicación del proceso InALPro a un dominio de aplicación?

Las interrogantes antes planteadas serán respondidas a través de del logro de los objetivos específicos enunciados en el presente estudio.

Objetivos

Objetivo General

Proponer un proceso de software para la ingeniería de la aplicación en líneas de producto con características de calidad.

Objetivos Específicos

1. Caracterizar las disciplinas del proceso de ingeniería de la aplicación en líneas de producto de software.
2. Especializar las disciplinas del proceso de ingeniería de la aplicación en líneas de producto de software.
3. Incorporar características de calidad en las disciplinas del proceso de ingeniería de la aplicación en líneas de producto de software.
4. Expresar las actividades y artefactos de las disciplinas de la ingeniería de la aplicación en líneas de producto de software (InALPro) mediante SPEM.
5. Aplicar el proceso InALPro en el dominio eLearning para ejemplificar su uso.

Justificación e Importancia

El objetivo principal de implementar un sistema de software es satisfacer exigencias de personas u organizaciones, es decir, aportar solución a algún problema, de acuerdo a Hamar, V. (2003) "Esta solución no es única, es por esto que los métodos deben seguir evolucionando y adaptándose a las nuevas necesidades". Por tanto, la existencia de nuevos modelos de procesos de software genera soluciones novedosas a las cada vez más complejas situaciones.

Aportar soluciones de software a problemas reales tiene dificultades relacionadas al método de desarrollo, al respecto Parnas, D. y Clements, P. (1986) y Nandhakumar, J. y Avison, J. (1999) argumentaban que los métodos tradicionales para sistemas de información son vistos principalmente como una ficción necesaria para presentar una imagen de control y además, son metodologías muy mecánicas para ser usadas en detalle.

En un estudio más reciente, The Standish Group en su Chaos Manifesto 2013 muestra cifras de los proyectos ágiles, donde apenas el 46% de los proyectos son exitosos, el 48% fueron deficientes y el 6% fracasaron; de los proyectos en cascada, sólo 49% son exitosos, el 43% fueron deficientes y el 8% fracasaron. Estas estadísticas son preocupantes ya que se refieren a métodos de desarrollo bastante conocidos y usados. Al respecto Pressman, R. (2010), señala que el código defectuoso es el responsable del 45% del tiempo que están fuera los sistemas basados en computadoras, con costos a las empresas estadounidenses alrededor de 100 mil millones de dólares según The Standish Group (2010).

En un artículo Gacitúa, R. (2003, p. 24) analiza estadísticas del software y concluye que "el desarrollo de software no sólo no ha logrado estándares de calidad aceptables" sino "que el desarrollo de software es una actividad caótica", en consecuencia, es pertinente mejorarlo. En particular, los métodos de LPS revisados carecen de normas claras de calidad.

Otro problema en el desarrollo de aplicaciones, es lo expuesto por Espinoza, G. (2012) "requisitos crecientes y cambiantes, lo cual puede incrementar o modificar

funcionalidades ya implementadas, que por no quedar claros al principio, generalmente aumentan costos y agendas planificadas, obteniéndose finalmente un producto con serias deficiencias técnicas". Un elemento relevante asociado, es el hecho de la inconformidad del usuario o cliente insatisfecho, lo cual se puede traducir en costos ocultos no apreciables o no cuantificables fácilmente.

También se encuentran pocos procesos basados en reutilización, a pesar de que Sametinger, J. (1997) afirma que aproximadamente el 75% de las funciones de una aplicación son reutilizables; es decir, la tendencia de la reutilización es aprovechar al máximo todos los elementos generados en el proceso ingenieril; según Montilva, J. (2006), estos elementos pueden ser: un componente de software, una especificación de requisitos, un modelo de negocios, un plan, una arquitectura de dominio, una especificación de prueba, entre otros; en consecuencia, se considera un enfoque interesante dentro del cual es pertinente continuar estudios.

Dentro del enfoque de reutilización se encuentran las líneas de producto de software (LPS), las cuales se han originado de la evolución de otros enfoques basados en reutilización, ver Figura 2; según Clements, P., y Northrop, L., (2002) las líneas de producto de software permiten la mejora en el tiempo de entrega y calidad del producto, como también, la disminución en la tasa de errores y en los costos de producción, mediante el aprovechamiento de los aspectos comunes y la gestión de los aspectos variables.

Sin embargo, la LPS presenta poca especificidad dentro de su fase de ingeniería de la aplicación, es decir, qué artefactos, disciplinas y/o actividades contempla su ejecución. De acuerdo a Sánchez, P., García-Saiz, D. y Zorrilla, M. (2013), la importancia de contar con procesos que guíen la fase de la ingeniería de la aplicación, radica en que la ingeniería de la aplicación debe ser instanciada tantas veces como productos concretos se necesiten, es decir el proceso se puede ejecutar varias veces durante la vida útil de la LPS.

Por todo lo anteriormente expuesto, se considera conveniente estudiar el proceso de LPS, en especial, su fase de ingeniería de la aplicación, para posteriormente

plantear un proceso de software fundamentado en la reutilización y con características de calidad.

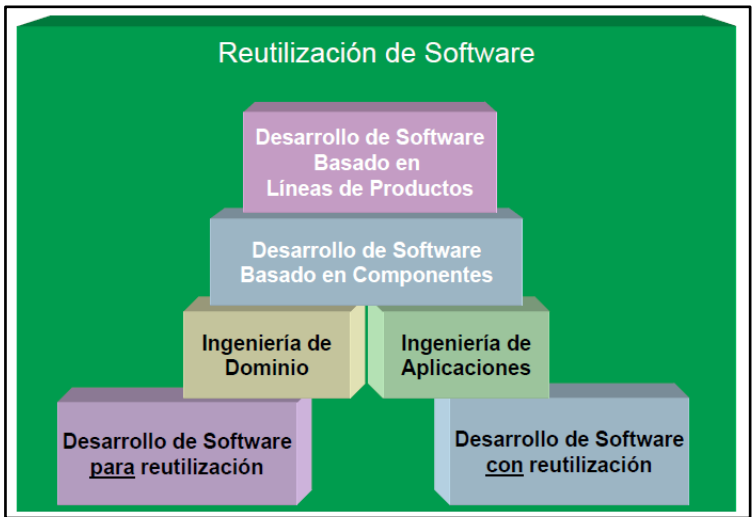


Figura 2. Evolución de la Reutilización de Software.
Fuente: Montilva, J., (2006).

La contribución primordial del presente estudio es abordar la disciplina de la ingeniería de la aplicación, básicamente focalizado en los aspectos que presentan debilidades como la calidad del proceso y del producto; es decir, la propuesta del proceso InALPro, el cual enmarca en un modelo de calidad las actividades y artefactos de esta fase de la LPS, aspecto no encontrado en procesos similares. Con la propuesta de InALPro se complementa InDoCaS planteando una extensión de este y así proponer un proceso para LPS con enfoque de calidad, lo que constituye una solución metodológica en el ámbito del desarrollo de software académico y empresarial. Vale mencionar que la utilización de metodologías para el desarrollo de software es positivo para las organizaciones, según INTECO (2009):

- Ayuda en la comprensión del problema.
- Facilita la planificación, control y seguimiento de un proyecto.
- Facilita el mantenimiento del producto final.
- Permite la reutilización de partes del producto.
- Mejora la relación costo/beneficio y optimiza el uso de recursos disponibles.
- Facilita la evaluación de resultados y cumplimiento de los objetivos.

- Favorece la calidad en el producto final.

De igual manera, se espera que el enfoque investigativo de la ciencia del diseño, así como también la implementación de la ingeniería de método situacional, sea un aporte al conocimiento académico para la realización de estudios similares en el ámbito de la ingeniería del software. Contribuyendo así a los proyectos de investigación en software en los institutos de educación universitaria en Venezuela y en el resto del mundo.

Alcance

Esta investigación comprende la revisión de métodos reconocidos de LPS, TWIN, WATCH, ESPLEP, InDoCaS y el modelo del SEI; luego la aplicación de la ingeniería de método situacional, para elaborar la propuesta de extensión de InDoCaS y generar un proceso de software para la ingeniería de la aplicación con enfoque de calidad la norma ISO/IEC 25010, este proceso sólo contempla las disciplinas de análisis y diseño, dejando para un estudio futuro la disciplina de la implementación de la aplicación. El proceso obtenido es denominado Ingeniería de la Aplicación para las Líneas de Producto Software, InALPro, (Figura 3), el cual se aplica conjuntamente con InDoCaS al dominio eLearning para ejemplificar el proceso de implementación de una LPS con enfoque de calidad en la UPTP "Juan de Jesús Montilla".

Limitaciones

Los métodos encontrados de LPS no reflejan especificidad en sus disciplinas, las actividades indicadas son muy generales, dejando interrogantes al seguir el modelo, lo que puede ocasionar pérdida de tiempo, resultados incompletos o inesperados y errores en el producto, en resumen, una experiencia insatisfactoria con resultados impredecibles. Por otro lado, estos modelos no indican explícitamente normas de calidad en el proceso y producto software, lo que dificulta la estimación de la calidad en las disciplinas de la LPS.

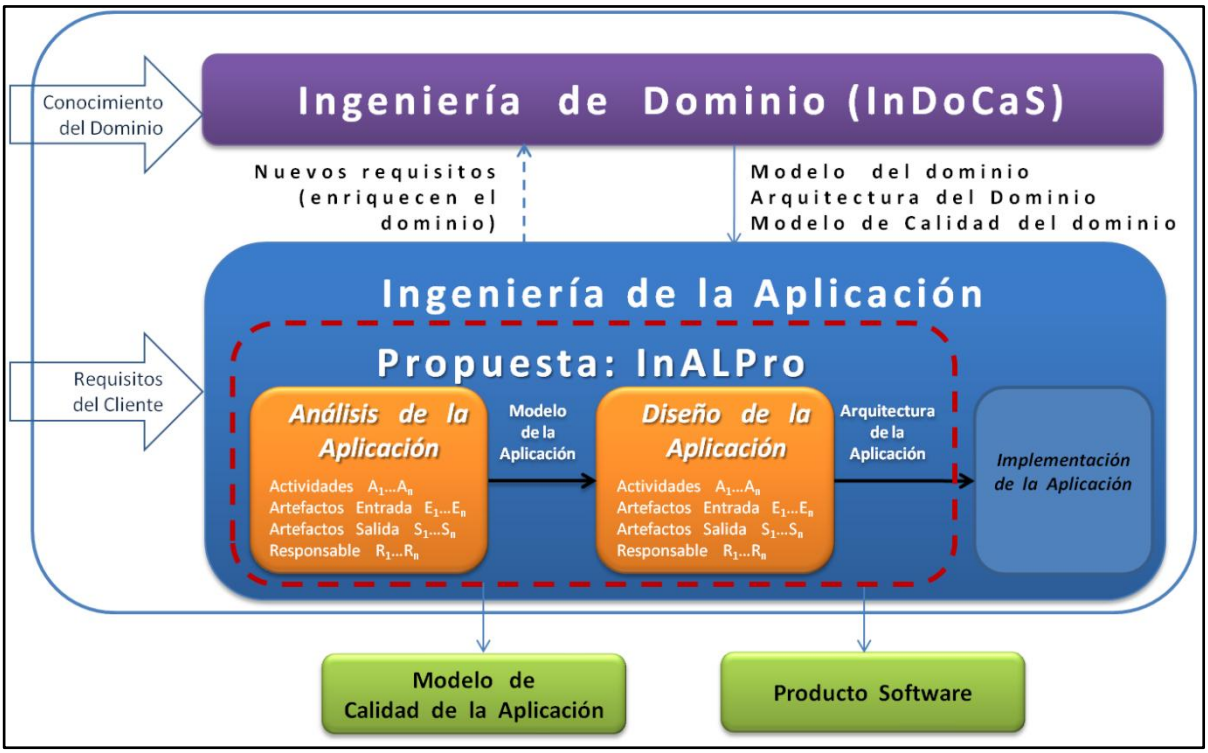


Figura 3. Alcance de la investigación: InALPro, proceso propuesto en el contexto de las líneas de producto de software.
Fuente: Autora de la investigación (2015).

CAPITULO II

MARCO TEÓRICO

Antecedentes

Desde la década de los 60's los desarrolladores han tenido la inquietud de reutilizar el software fabricado, pero solo era posible, a lo sumo, reutilizar sub-rutinas, ya en los 70's hubo un avance, se reutilizaban módulos, pero se debía modificar el código para adaptar los aspectos no compatibles en el nuevo sistema, en los 80's la programación orientada a objetos permitió reutilizar el código a través de la herencia, y finalmente en los 90's, se da inicio a los enfoques de reutilización a nivel de componentes, elementos arquitecturales y LPS.

En el enfoque de LPS según Piattini, M. y Garzás, J. (2007) "La reutilización ya no es oportunista, sino planificada", esto significa un reto para los ingenieros de software, quienes deben desarrollar todos los activos de software de manera que en un futuro no muy lejano puedan ser reusados.

Para desarrollar esta visión planificada de la reutilización es necesario contar con un modelo o proceso de desarrollo, de acuerdo a lo planteado por Canelón, R. (2010), "Un modelo de proceso para el desarrollo de software debe ser utilizado para capturar las mejores prácticas que permitan delinear el problema dado, el mismo debe estar bien definido de modo de poder ser observado, medido y mejorado".

En este orden de ideas, se llevó a cabo una revisión en el ámbito de las propuestas de procesos de desarrollo de software en líneas de producto y calidad del software, ingeniería de métodos y eLearning como antecedentes del presente estudio. En la Figura 4, un mapa mental esquematiza los antecedentes de investigación.

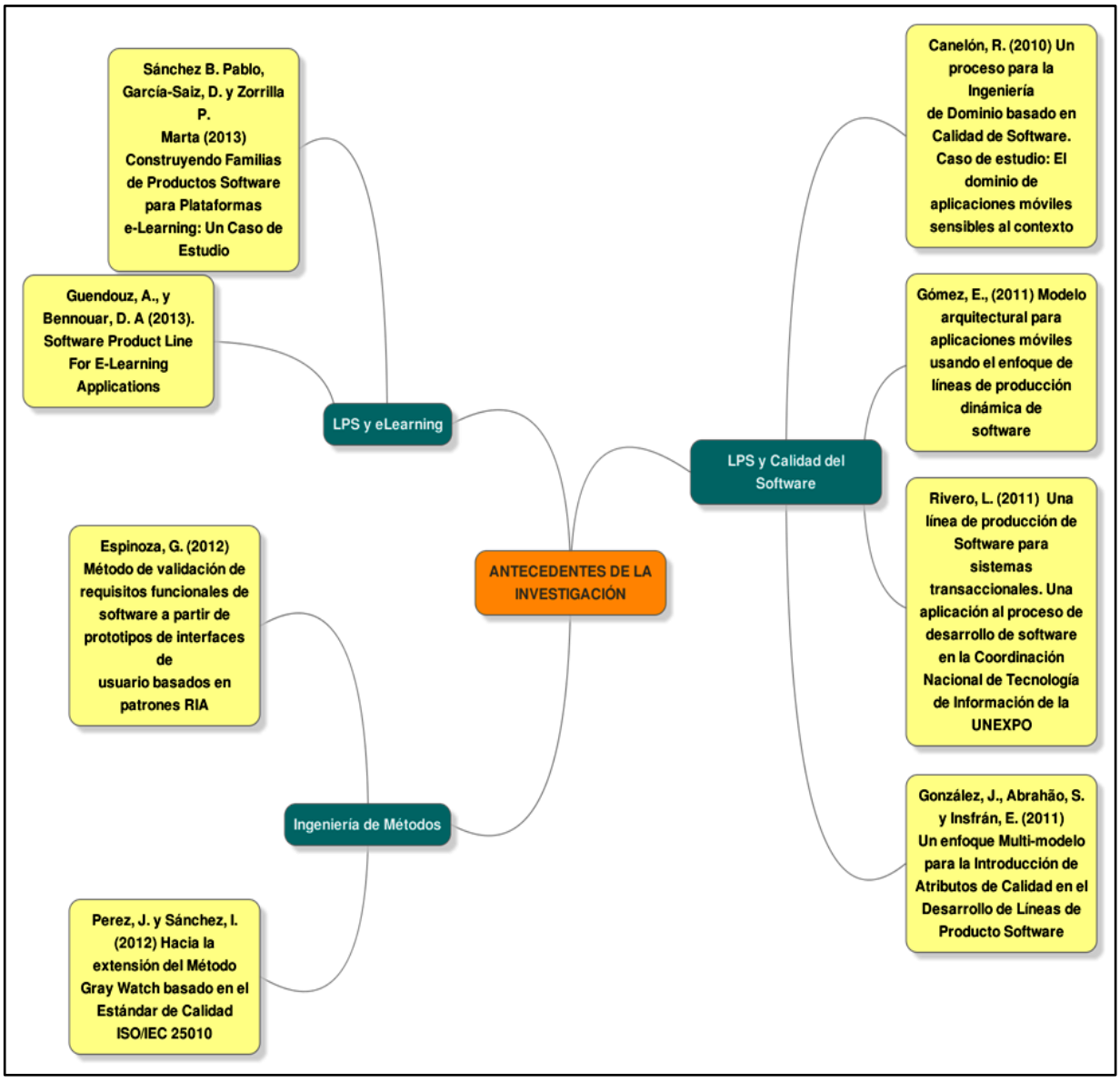


Figura 4. Mapa de antecedentes de investigación.
Fuente: Autora de la investigación (2015).

A continuación, resumen de antecedentes de los trabajos anteriormente mencionados.

- Canelón, R. (2010), presenta en su tesis doctoral "Un proceso para la Ingeniería de Dominio basado en Calidad de Software. Caso de estudio: El dominio de aplicaciones móviles sensibles al contexto" (InDoCaS), este proceso es un

método general de diseño arquitectónico con un enfoque de calidad del software basado en el estándar ISO/IEC 25010 (calidad del producto y del proceso software) y contempla específicamente dos de las tres disciplinas de la ingeniería de dominio: el análisis y el diseño. Con la aplicación de InDoCaS, se obtiene una arquitectura base para una familia de productos, es decir, este proceso puede ser utilizado en el desarrollo de LPS. Este antecedente constituye el fundamento de la presente investigación, porque InDoCaS fue seleccionado como proceso particular de la ingeniería de dominio por ser un proceso bien definido y con un enfoque de calidad asociado. Se debe resaltar que la ingeniería de dominio es el proceso inicial de todo modelo de desarrollo de LPS. Por lo tanto, para implementar las disciplinas de la ingeniería de la aplicación, es necesario, en primera instancia, cumplir las disciplinas del proceso InDoCaS. El proceso InDoCaS también fue presentado mediante un caso de estudio, específicamente, para aplicaciones de aprendizaje electrónico móvil sensibles al contexto.

- Espinoza, G. (2012), en su trabajo de grado de maestría "Método de validación de requisitos funcionales de software a partir de prototipos de interfaces de usuario basados en patrones RIA", diseña un método que permite al ingeniero de software validar requisitos funcionales de software a partir de interfaces de usuario, bajo un enfoque de calidad. Para el diseño de este método, implementa la ingeniería de método situacional en su enfoque basado en ensamblaje. Este trabajo se toma como antecedente por el uso de las características de calidad ISO/IEC 25010 y la aplicación de la ingeniería de método situacional, aunque, en el presente estudio se usa el enfoque de extensión.
- Gómez, E., (2012), en su trabajo de grado de maestría "Modelo arquitectural para aplicaciones móviles usando el enfoque de líneas de producción dinámica de software", diseña un modelo arquitectural fundamentado en el enfoque de líneas de producción dinámica de software para dispositivos móviles sensibles al contexto incorporando características de calidad de la norma ISO/IEC 25010; para el diseño de este modelo arquitectural se vale de la especialización de InDoCaS, este estudio coincide con la presente investigación en la aplicación de

las características de calidad ISO/IEC 25010, el uso de InDoCaS y de la especialización de procesos, es por tanto de referencia obligatoria, aunque en el proceso de ingeniería de aplicación no se dan mayores detalles de las disciplinas que lo componen y de los artefactos producidos.

- González, J., Abrahão, S. y Insfrán, E. (2011) en el artículo "Un enfoque Multi-modelo para la Introducción de Atributos de Calidad en el Desarrollo de Líneas de Producto Software", proponen agregar atributos de calidad a las LPS como una vista más del sistema, es decir, bajo la perspectiva de las líneas de producto, los sistemas deben tener: la vista de variabilidad, la vista de funcionalidad y la vista de la calidad. Estas características de calidad y su relación con las demás vistas del sistema, son tomadas en cuenta en la fase de ingeniería de la aplicación para la configuración del producto de acuerdo a sus requisitos. Los atributos de calidad en líneas de producto y su relación con la fase de ingeniería de la aplicación referencia el tema principal del presente estudio, permite disponer de una guía en la forma como modelan los aspectos de calidad a la línea y su influencia en el producto final.
- Guendouz, A., y Bennouar, D. A (2013), a través de su artículo titulado "A Software Product Line for eLearning", plantean la construcción de una línea de producto de software para eLearning como una alternativa eficiente y económica de desarrollo. Aquí se aplican la ingeniería de dominio clásica de las líneas de producto, es decir, análisis, diseño y realización del dominio, posteriormente, en la fase de ingeniería de la aplicación, no se aplican las disciplinas de análisis, diseño y construcción del producto, simplemente hacen una derivación de la arquitectura del dominio para obtener la arquitectura del producto, con la respectiva selección de los artefactos arquitecturales específicos. A pesar de lo anterior, son muy útiles las actividades explicadas y artefactos identificados como guía del presente estudio.
- Pérez, J. y Sánchez, I. (2012), en su artículo titulado "Hacia la extensión del Método Gray Watch basado en el Estándar de Calidad ISO/IEC 25010", proponen una extensión del método Gray Watch, específicamente en los procesos análisis y

diseño, acoplando los productos obtenidos al proceso de implementación. Dicha propuesta consiste en utilizar el estándar de calidad del producto ISO/IEC 25010, que establece criterios para la especificación de requisitos de calidad de productos de software, sus métricas y su evaluación, e incluye un modelo de calidad compuesto por características y subcaracterísticas. Este articulo referencia el uso de la ingeniería de métodos, utiliza el método Gray Watch y referencia la implementación de características de calidad ISO/IEC 25010, aspectos estudiados en la presente investigación.

- Rivero, L. (2011), en su trabajo de grado de maestría titulado "Una línea de producción de Software para sistemas transaccionales. Una aplicación al proceso de desarrollo de software en la Coordinación Nacional de Tecnología de Información de la UNEXPO", en ese estudio, el autor, diseña una línea de producción de software para sistemas transaccionales, con este fin expande el proceso InDoCaS, agregándole las actividades de la disciplina de implementación del dominio con características de calidad como en InDoCaS, para esto, instanció el método Watch-Component. Oportunamente, Rivero usa InDoCaS (como base) para implementar una línea de producto de software. Este procedimiento es similar al planteado en este estudio, debido a que se toma un método base y se extiende su alcance, se usa el modelo de calidad de InDoCaS y finalmente se implementa para crear una LPS. La deficiencia observada es que para la implementación de la LPS no se aplica ninguna disciplina de la ingeniería de la aplicación, es decir, no se especializa la arquitectura, por lo cual no se considera completo el proceso de LPS.
- Sánchez et al. (2013), en su artículo titulado "Construyendo Familias de Productos Software para Plataformas eLearning: Un Caso de Estudio", muestran cómo construir una línea de producto software para una familia de aplicaciones en el dominio eLearning, este proceso se ejemplifica mediante el caso de estudio E-learning Web Miner (ElWM). Para lograr el objetivo planteado, los autores, realizaron un análisis de variabilidad a la familia de productos y obtener el modelo de características del dominio estudiado. A continuación, diseñan una

arquitectura de referencia para el dominio, y luego automatizan el proceso de instanciación de la arquitectura para obtener la infraestructura específica de la línea, esta automatización la plantean mediante el uso de plantillas de generación de código que guían el proceso. Posteriormente, pasan a la fase de la ingeniería de la aplicación, donde realizan actividades de configuración y derivación automática del producto, que permite obtener los artefactos de software concretos para la aplicación. Este articulo orienta la presente investigación por servir de guía en el desarrollo de la línea de producto software, aunque no es suficientemente explicito en la fase de ingeniería de la aplicación.

Los antecedentes antes descritos se resumen en la Tabla 1.

Antecedente	Objetivo	Aporte	Limitación
Canelón, R. (2010) Un proceso para la Ingeniería de Dominio basado en Calidad de Software. Caso de estudio: El dominio de aplicaciones móviles sensibles al contexto	Proponer un método general de diseño arquitectónico de ingeniería de dominio con un enfoque de calidad	Proceso para ingeniería de dominio para familia de productos enfocado en calidad del proceso y del producto software (ISO/IEC 25010)	No contempla la disciplina de implementación del dominio, ni la fase de ingeniería de la aplicación.
Espinoza, G. (2012) Método de validación de requisitos funcionales de software a partir de prototipos de interfaces de usuario basados en patrones RIA	Diseñar un método de validación de requisitos funcionales de software a partir de interfaces de usuario basados en patrones RIA bajo un enfoque de calidad	Usa el modelo de calidad ISO/IEC 25010 y evidencia la aplicación de la ingeniería de método situacional.	Este método sólo referencia la etapa inicial del proceso de desarrollo de software
Gómez, E., (2011) Modelo arquitectural para aplicaciones móviles usando el enfoque de líneas de producción dinámica de software	Proponer un modelo arquitectural para aplicaciones móviles usando el enfoque de líneas de producción dinámica de software	Incorpora un punto de variación y agrega la actividad "Generar modelos dinámicos" al proceso InDoCaS.	Están ausentes las disciplinas implementadas en el proceso de ingeniería de la aplicación.
González, J., Abrahão, S. y Insfrán, E. (2011) Un enfoque Multi-modelo para la Introducción de Atributos de Calidad en el Desarrollo de Líneas de Producto Software	Proponer la integración de características de calidad en líneas de producto, permitiendo establecer relaciones entre las diferentes vistas del sistema (variabilidad, funcionalidad y calidad)	Modelo de calidad para líneas de producto, en especial la influencia de este modelo sobre la fase de ingeniería de la aplicación.	No se evidencian actividades del proceso de desarrollo de la línea de producto, ni de sus fases.

Guendouz, A., y Bennouar, D. A (2013) Software Product Line For E-Learning Applications	Muestran cómo construir una línea de producto de software para aplicaciones eLearning	Ejemplifica detalladamente la fase de ingeniería de dominio de la línea de producto software. Y menos especifica, pero útil la fase de ingeniería de la aplicación.	En la disciplina de la ingeniería de la aplicación se explican las tareas de forma general y no se ejemplifican los resultados parciales.
Pérez, J. y Sánchez, I. (2012) Hacia la extensión del Método Gray Watch basado en el Estándar de Calidad ISO/IEC 25010	Proponer una extensión del método Gray Watch, utilizando como base InDoCaS y el estándar ISO/IEC 25010	Extiende un método usando la ingeniería de métodos y usa como soporte InDoCaS, además de aplicar el estándar de calidad del proceso y del producto software	No es detallado en el proceso de extensión del método Gray Watch, y solo incorpora características de calidad al producto software
Rivero, L. (2011) Una línea de producción de Software para sistemas transaccionales. Una aplicación al proceso de desarrollo de software en la Coordinación Nacional de Tecnología de Información de la UNEXPO	Diseñar una línea de producción de software para sistemas transaccionales usando InDoCaS	Expansión del proceso InDoCaS e implementación de una línea de producción de software para sistemas transaccionales. Bajo el enfoque de calidad propuesto en InDoCaS	No referencia las disciplinas de la fase de ingeniería de la aplicación.
Sánchez et al. (2013) Construyendo Familias de Productos Software para Plataformas e-Learning: Un Caso de Estudio	Adaptación de una plataforma eLearning mediante la implementación de una línea de producto software que aplica refactorización al código.	Evidencia el proceso de ingeniería de la aplicación en arquitecturas eLearning.	Las actividades de la ingeniería de la aplicación: configuración y derivación del producto, no son especificadas.

Tabla 1. Resumen de antecedentes de investigación.
Fuente: Autora de la investigación (2014).

Los trabajos anteriormente descritos, fueron desarrollados en los contextos del proceso de desarrollo de líneas de producto y calidad del software, ejes fundamentales de la presente investigación. La variedad de estos estudios refleja la vigencia e importancia de los procesos de desarrollo para LPS, pero es notorio que los métodos encontrados no son muy detallados en las disciplinas e implementación de características de calidad en el proceso de ingeniería de la aplicación. Por lo anterior, se propone un proceso para la ingeniería de la aplicación en líneas de producto de software con características de calidad (InALPro).

Bases Teóricas

Para la realización del presente estudio se hizo la revisión bibliográfica de las temáticas relacionadas a la reutilización, LPS, calidad del software, eLearning, ciencias del diseño, ingeniería de método situacional y SPEM, en la Figura 5 se muestra gráficamente el sustento del proceso para la ingeniería de la aplicación en líneas de producto.

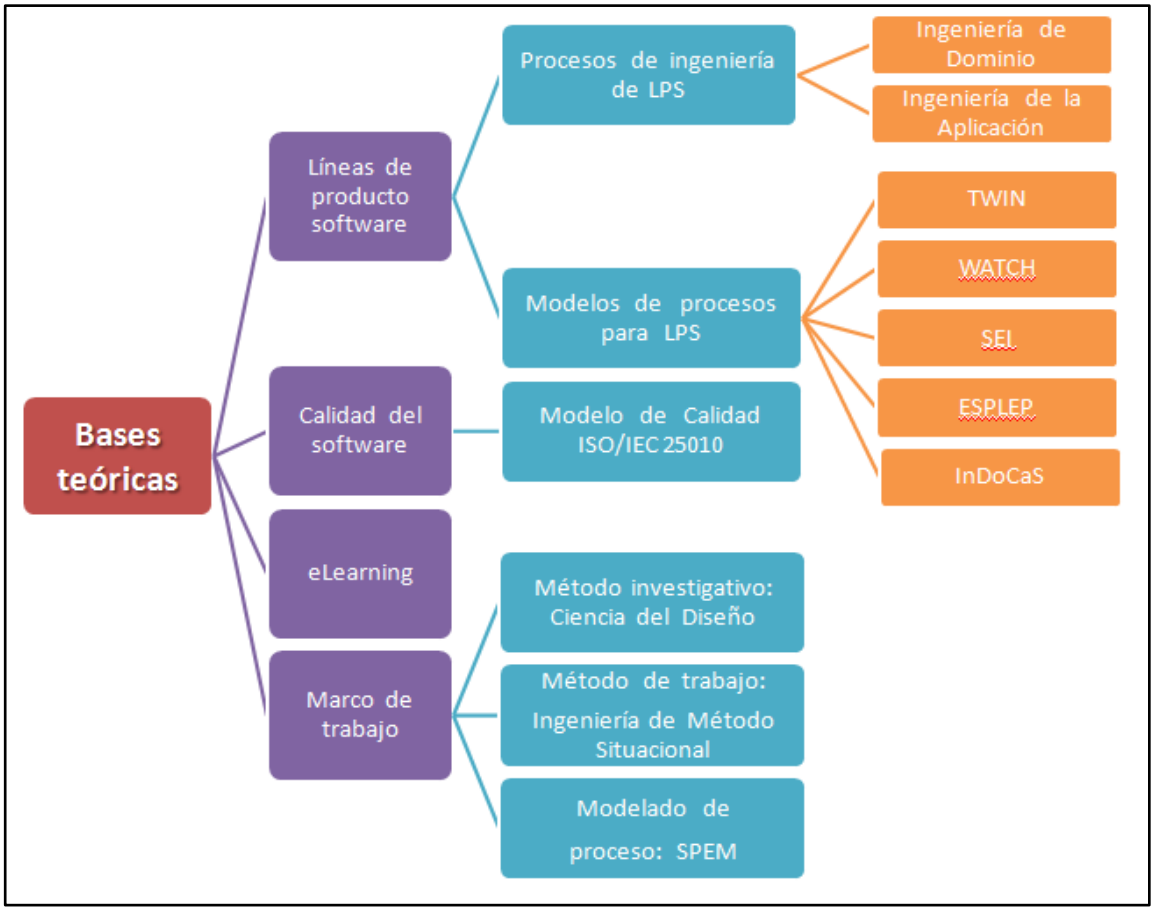


Figura 5. Esquema de las bases teóricas de la investigación.
Fuente: Autora de la investigación (2015).

Enfoques de desarrollo en la ingeniería de software

Leyvan, E., Prieto, J., Sampalo, M., y Garzón, M. (2006) proponen que "Una metodología de desarrollo es una recopilación de técnicas y procedimientos estructurados en fases para la producción de productos software de manera eficaz y englobando todo el ciclo de vida del mismo". En otras palabras, la metodología o

enfoque de desarrollo, indica qué hacer, cómo, cuándo y quién debe hacerlo, y también, determina las etapas y controles a aplicar.

En este mismo orden de ideas, Sommerville, I. (2005) "Un modelo de proceso del software es una descripción simplificada de un proceso del software que presenta una visión de ese proceso", y también define "Un método de ingeniería del software es un enfoque estructurado para el desarrollo de software cuyo propósito es facilitar la producción de software de alta calidad y de una forma costeable".

Vistas las anteriores definiciones, se puede deducir, que la importancia de adoptar un modelo de desarrollo de software es disponer de una guía organizada que permita gestionar el proceso, definir la estructura y propiedades del software a producir. En el marco de esta investigación se pretende proponer un proceso de software para la ingeniería de la aplicación con enfoque de calidad.

Reutilización

Reutilizar es usar varias veces lo que se construyó una vez, como lo define Sametinger, J. (1997) la reutilización de software es el proceso de crear aplicaciones de software a partir de software existente, en lugar de desarrollarlo desde cero. Esta idea de construir una vez, para usar muchas veces es una meta que la ingeniería del software plantea por los beneficios ofrecidos, así lo esboza Sommerville, I., (2005), la tendencia hacia el desarrollo basado en reutilización viene dada como respuesta a las demandas de una menor producción de software y de menores costos de mantenimiento, de una entrega más rápida y del incremento en la calidad.

Las LPS son uno de los modelos más recientes para abordar el enfoque de reutilización, el cual se detalla a continuación.

Líneas de producto de software

Hanssen, G., (2010) Define una línea de productos como un conjunto de productos de software relacionados que tienen algunas partes comunes, por ejemplo componentes de software, diseño arquitectónico, estructuras de datos, pero que aún presentan características distintas (p. 15).

Según Clements, P., y Northrop, L. (2001, p. 5), las líneas de producto de software son un conjunto de sistemas de software, que comparten un grupo 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.

Bass, L., Clements, P., Cohen, S., Northrop, L. y Withey, J., (1997) plantean que una línea de producto es definida como un grupo de productos que comparten un conjunto de características comunes gestionadas, que satisfacen las necesidades específicas de un sector concreto del mercado.

De lo anterior, es posible definir la línea de productos como el conjunto de activos relacionados, con características comunes y no comunes que satisfacen las necesidades de una porción del mercado y que han sido desarrollados de forma planificada.

Beneficios de las líneas de producto de software

De acuerdo a Piattini, M. y Garzás, J. (2007), "Las LPS pueden incrementar significativamente la productividad de los ingenieros de software, entendida como una reducción en el esfuerzo y el coste necesario para desarrollar, poner en marcha y mantener un conjunto de productos software similares". Es decir, este beneficio se percibe directamente en aumento de la productividad y disminución en el costo de desarrollo y mantenimiento de varios productos. Esto se debe a un enfoque planificado de la reutilización, donde se aprovechan los aspectos comunes y se gestionan los variables. Al compararse con un enfoque de desarrollo convencional puede parecer más costoso el de LPS, pero a mediano plazo se observan las ventajas. Se estima que a partir de tres productos se comienza a percibir el ahorro en costos. Ver Figura 6.

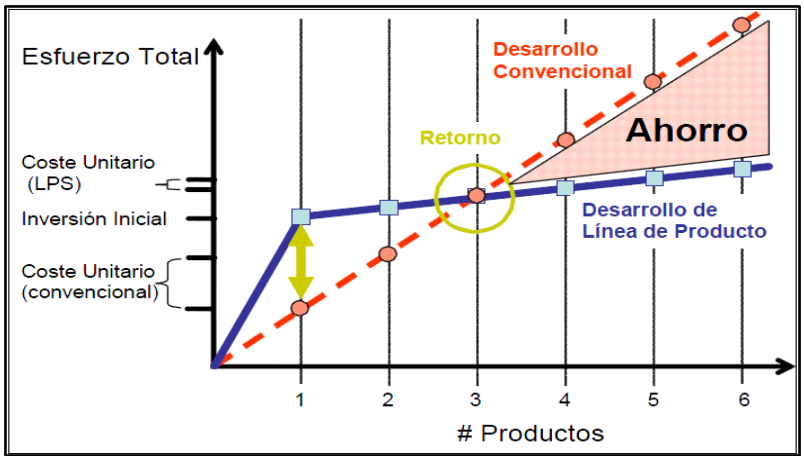


Figura 6. Desarrollo convencional Vs. Línea de producto.
Fuente: Piattini, M. y Garzás, J. (2007)

Los beneficios relacionados a la calidad, según Piattini, M. y Garzás, J. (2007), son el aumento del grado de precisión con que el producto se ajusta a las necesidades de cada cliente y la disminución en la tasa de defectos de los productos de la línea, estos beneficios se derivan directamente de la reutilización de los elementos comunes. La continuada utilización de estos elementos a lo largo del tiempo hace que finalmente estén muy depurados/probados. Además, los beneficios de encontrar y eliminar un defecto en un *activo principal* no se limitan al producto donde se detecta el error, sino que se disemina entre todos los productos de la LPS.

Procesos de la ingeniería de líneas de producto de software

Como ya se mencionó, Piattini, M. y Garzás, J. (2007), afirman que un aspecto central compartido por las distintas metodologías de desarrollo de Línea de Producto Software es la división de los procesos de ingeniería en dos equipos de trabajo, el primer equipo se encarga de la Ingeniería de Dominio y el segundo equipo se encarga de la Ingeniería de Producto (aplicación).

De igual forma, Ávila, O., Estévez, A., Sánchez, E. y Roda, J., (2007), plantean la existencia de una "dicotomía entre ingeniería de dominio e ingeniería de aplicación" en el paradigma de desarrollo de LPS.

La mayoría de los autores, coinciden que las LPS están constituidas al menos por estos dos procesos, otros autores, agregan un proceso de gestión organizacional, que

va a depender de equipo que implemente dicha línea de producto. En la Figura 7, la estructura de línea de producto planteada por Pohl et al. (2005), este planteamiento coincide de forma general con los autores antes citados.

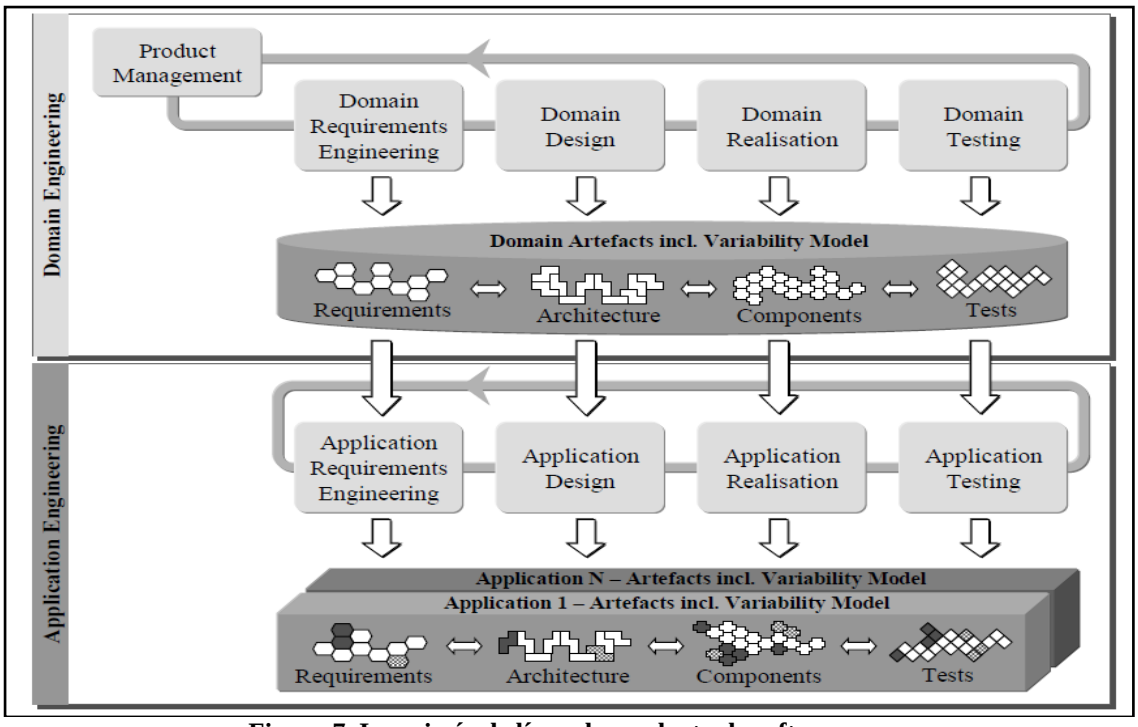


Figura 7. Ingeniería de línea de producto de software.
Fuente: Pohl, K., Böckle, G., y Linden, F. (2005)

Ingeniería del Dominio

Para Pohl et al. (2005), La ingeniería de dominio es el proceso de ingeniería de la línea de productos de software en el que se definen y construyen el carácter común y la variabilidad de la línea de productos.

Según Piattini, M. y Garzás, J. (2007), el equipo de ingeniería de dominio es responsable de "estudiar el dominio, definir su alcance (requisitos) dentro del mercado objetivo de la LPS, definir las features, implementar los core assets reutilizables y su mecanismo de variabilidad, y establecer cómo es el plan de producción".

Según Pohl et al. (2005), las metas claves del proceso de ingeniería de dominio son:

- Definir la concordancia y la variabilidad de la línea de producto de software.
- Definir el conjunto de aplicaciones previstas para la línea de producto de software, es decir, definir el alcance de la línea de productos.
- Definir y construir artefactos reusables con la variabilidad deseada.

Los subprocesos de la ingeniería de dominio según Günther, S. (2008), Montagud, S. (2009) y Piattini, M. y Garzás, J. (2007) son:

Análisis

Es la actividad de identificar y sistematizar la información necesaria para caracterizar el dominio de la LPS, este conocimiento será reutilizable para todos los productos de la línea.

Diseño

Consiste en identificar los componentes comunes y su relación entre ellos, en esta fase se organiza la arquitectura de referencia de la línea de producto.

Implementación

En esta etapa, se desarrollan los activos comunes, para cada activo se decide si es conveniente construirlo desde el comienzo o reutilizar implementaciones existentes.

Ingeniería de la Aplicación

Para Pohl et al. (2005), la ingeniería de aplicación es el proceso de ingeniería de la línea de productos de software en el que las aplicaciones de la línea se construyen mediante la reutilización de artefactos del dominio y la explotación de la variabilidad de la LPS.

Según, Piattini, M. y Garzás, J. (2007), el equipo de ingeniería de aplicación tiene responsabilidades que "incluyen desarrollar los productos para clientes concretos, a partir de los recursos basados no en los requisitos del dominio, sino en requisitos concretos de clientes", pero usando los recursos creados por el equipo de ingeniería de dominio.

Según Pohl et al. (2005), las metas claves del proceso de ingeniería de aplicación son:

- Lograr el más alto grado de reutilización posible en los activos del dominio y desarrollar una aplicación de línea de producto.
- Explotar la convergencia y la variabilidad de la línea de producto de software durante el desarrollo de una aplicación de línea de productos.
- Documentar los artefactos de la aplicación, es decir, requerimientos de aplicación, la arquitectura y los relacionan con los objetos de dominio.
- Enlazar la variabilidad de acuerdo a las necesidades de aplicación de los requisitos sobre la arquitectura, los componentes y los casos de prueba.
- Estimar el impacto entre las diferencias entre los requisitos del dominio y de la aplicación sobre la arquitectura, los componentes y las pruebas.

Los subprocesos o disciplinas de la ingeniería de aplicación según Günther, S. (2008), Montagud, S. (2009) y Piattini, M. y Garzás, J. (2007) son: análisis, diseño e implementación de la aplicación.

Análisis de la aplicación

En esta fase, se obtienen los requisitos derivándolos de los requisitos de la ingeniería del dominio, los requisitos del cliente, así como de las características de la aplicación contenidas en el modelo variabilidad y tiene como salidas los requisitos de la aplicación. Se deben detectar los artefactos de la ingeniería de requisitos del dominio que no satisfacen los requisitos de los interesados en la ingeniería de la aplicación, estos requisitos son evaluados en relación al esfuerzo de adaptación necesaria y la adecuada documentación en la ingeniería de requisitos del dominio.

Diseño de la aplicación

Pohl et al. (2005), acota que este subproceso usa como referencia la arquitectura generada en la ingeniería del dominio incorporando nuevas adaptaciones especificas en la aplicación. Tiene como objetivo generar la arquitectura de la aplicación. Recibe

como entrada la arquitectura de referencia y los requisitos capturados de la aplicación y genera como salida la arquitectura de la aplicación.

Implementación de la aplicación

Pohl et al. (2005), plantea que en esta disciplina se lleva a cabo la implementación de la aplicación, cuyo objetivo es la selección y configuración de los componentes de software reutilizables, así como la realización de los activos específicos de la aplicación; ambas actividades son agrupadas para crear la aplicación. La entrada del proceso consiste en la arquitectura de la aplicación y en los artefactos reutilizables. La salida genera una aplicación que se ejecuta en conjunto con los artefactos del diseño específico.

Modelos de procesos para líneas de producto de software

De acuerdo a Sommerville, I., (2005), "un proceso de software se define como un marco de trabajo para las tareas que se requieren en la construcción de software de alta calidad" (P. 23), en este orden de ideas, es importante revisar los principales modelos para líneas de producto software.

Según Montilva, J. (2006), algunos modelos de procesos representativos para las líneas de producto de software son TWIN, WATCH, Software Engineering Institute (SEI), ESPLEP. A continuación una breve revisión de cada uno.

Modelo TWIN

Según Canelón, R. (2010), es un modelo de desarrollo basado en la reutilización de componentes. Este modelo integra los modelos de procesos para:

- Desarrollo de activos reutilizables, ingeniería de dominio.
- Desarrollo de aplicaciones basadas en reutilización. Ingeniería de aplicación.

Este modelo se basa en los artefactos y componentes reutilizables generados por la ingeniería del dominio, para luego ser desarrollados en la ingeniería de aplicación, es decir, es un clásico modelo de línea de producto de software. Ver Figura 8.

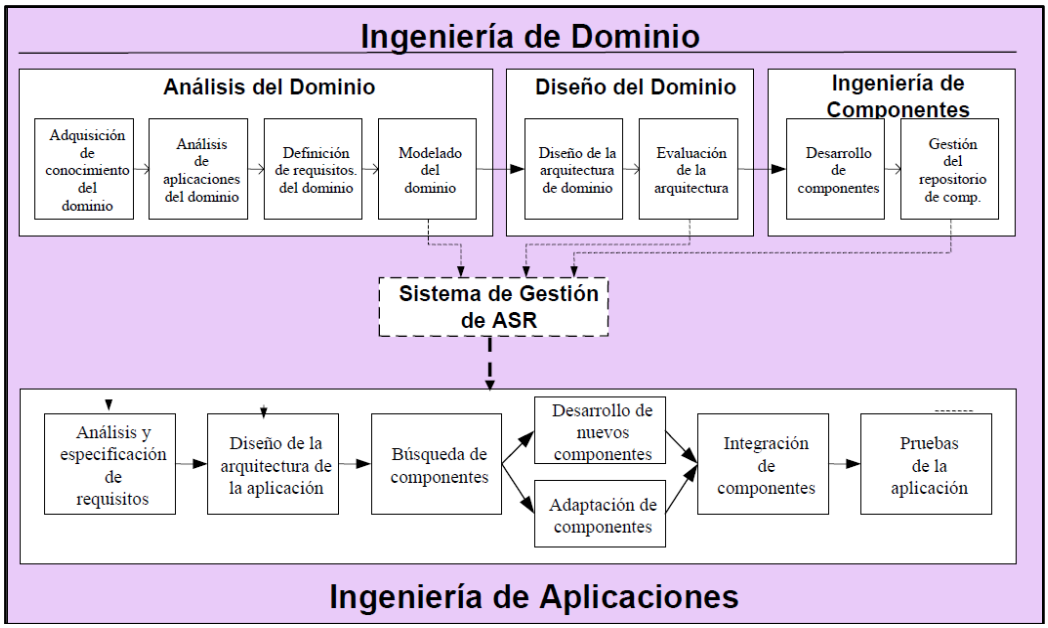


Figura 8. Modelo Twin.
Fuente: Montilva, J. (2006).

Modelo WATCH

Según Montilva J., Barrios J. y Rivero M., (2008), El método WATCH surgió de la necesidad de un modelo de proceso para el desarrollo de proyectos de software pequeños y medianos, desarrollado por pequeñas empresas, este fue el resultado de la integración de modelos de desarrollo de software tales como: espiral, V y orientado a objetos descrito por Bruegge y Dutoit, (2000). El estándar (IEEE 1074, 1995) para el desarrollo de procesos de ciclo de vida permitió el marco metodológico necesario para desarrollar al modelo WATCH. El método WATCH ha sido refinado por varios autores, entre ellos Hamar, V. (2003) y Pérez, J. y Sánchez, I. (2012), es un marco metodológico que describe los procesos técnicos, gerenciales y de soporte que deben emplear los equipos de trabajo que tendrán a su cargo el desarrollo de aplicaciones de software empresarial. El método WATCH consta de dos componentes metodológicos:

WATCH Componet, Hamar, V. y Montilva, J. (2003) Es un método desarrollado expresamente para producir Componentes de Software Reutilizable (CSR) adaptable al proceso de ingeniería de dominio, ver Figura 9.

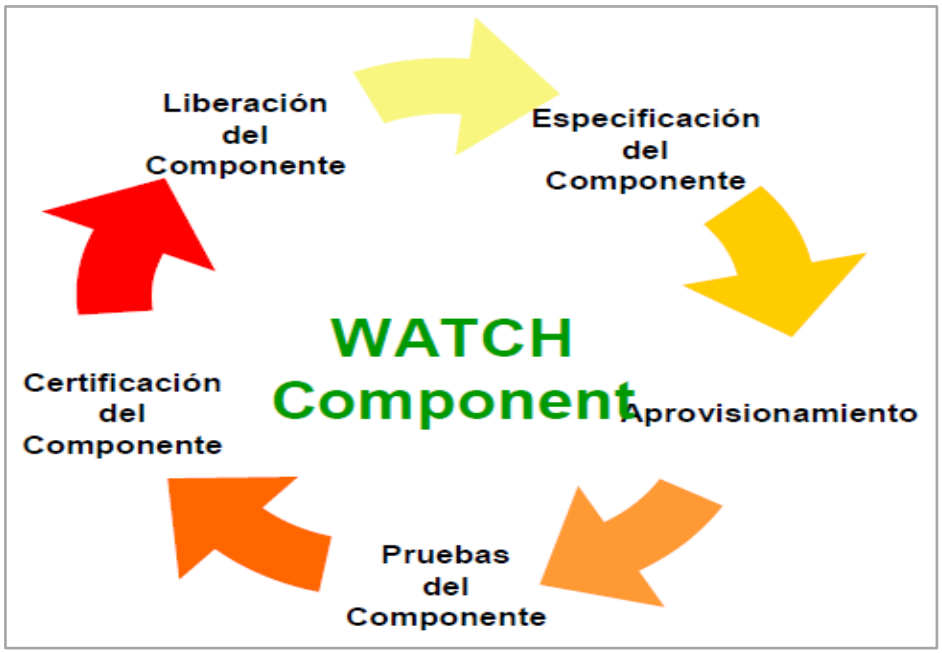


Figura 9. Metódo WATCH Component.
Fuente: Montilva, J. (2006)

WATCH Application: Es un modelo de procesos para el desarrollo de aplicaciones empresariales diseñado por Montilva, J. y Barrios, (2004) adaptable al proceso de ingeniería de la aplicación, ver Figura 10.

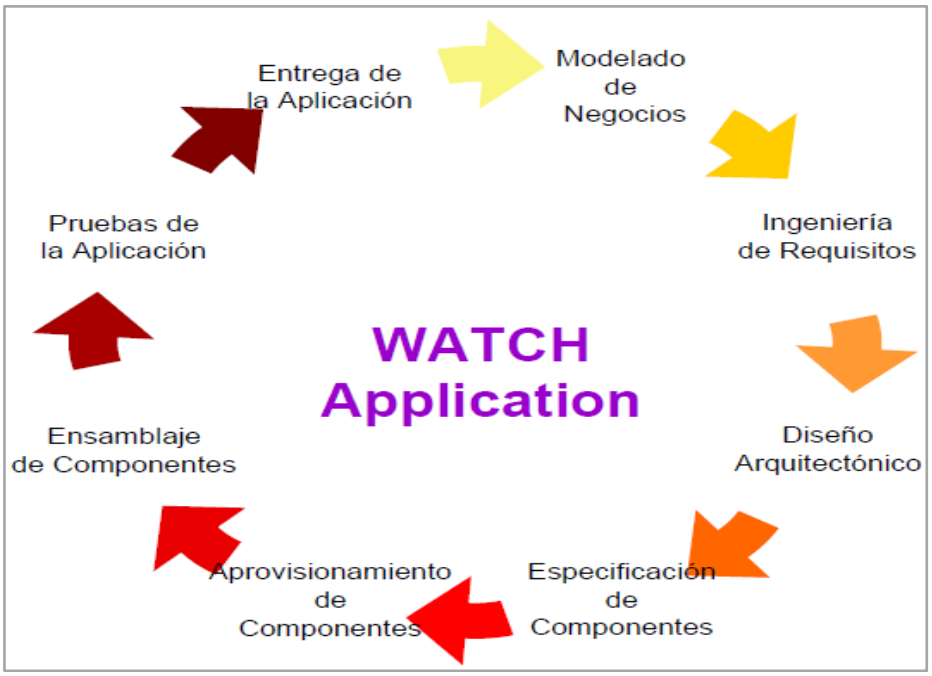


Figura 10. Método WATCH Application.
Fuente: Montilva, J. (2006).

Modelo del SEI

Es un Framework para el desarrollo de líneas de productos de software. Actualmente dispone de varias versiones ya que ha ido evolucionando a lo largo del tiempo. Esta información ha sido obtenida de los estudios de las organizaciones que han construido líneas de productos, y de destacados profesionales de las líneas de productos de software que han aportado sus experiencias. El objetivo de este modelo es proporcionar los recursos, coordinar y supervisar el desarrollo de activos y productos. Ver Figura 11.

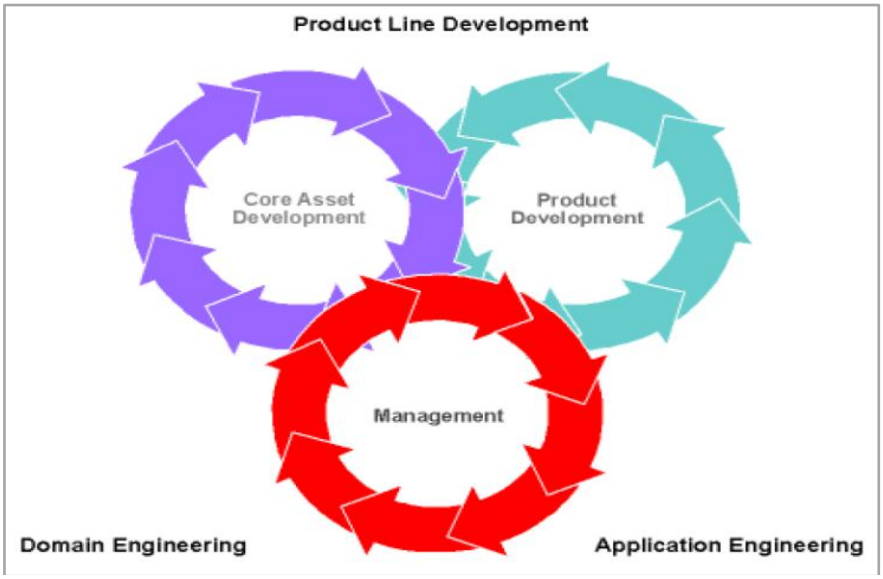


Figura 11. Modelo SEI.
Fuente: Clements, P., y Northrop, L., (2002)

Modelo ESPLEP

Según Gomaa, H., (2011), el Evolutionary Software Product Line Engineering Process (ESPLEP), es un modelo de proceso de software que elimina la distinción tradicional entre el desarrollo y mantenimiento de software. En cambio, los sistemas evolucionan a través de varias iteraciones. Por lo tanto, los sistemas desarrollados con este enfoque necesitan ser capaces de adaptarse a los cambios en las necesidades durante cada iteración. Además, debido a los nuevos sistemas de software son a menudo extensiones de las ya existentes, el modelo ESPLEP toma un producto de software punto de vista de línea, lo que permite el desarrollo de las familias de

productos de software. Al considerar el desarrollo de una familia de productos que constituyen una línea de productos, es necesario reemplazar las actividades de software basados en el modelo de desarrollo para los sistemas individuales de modelado de requisitos, modelado, análisis y modelado de diseño de las actividades de desarrollo que abarcan la gama de productos, ver Figura 12 y Figura 13.

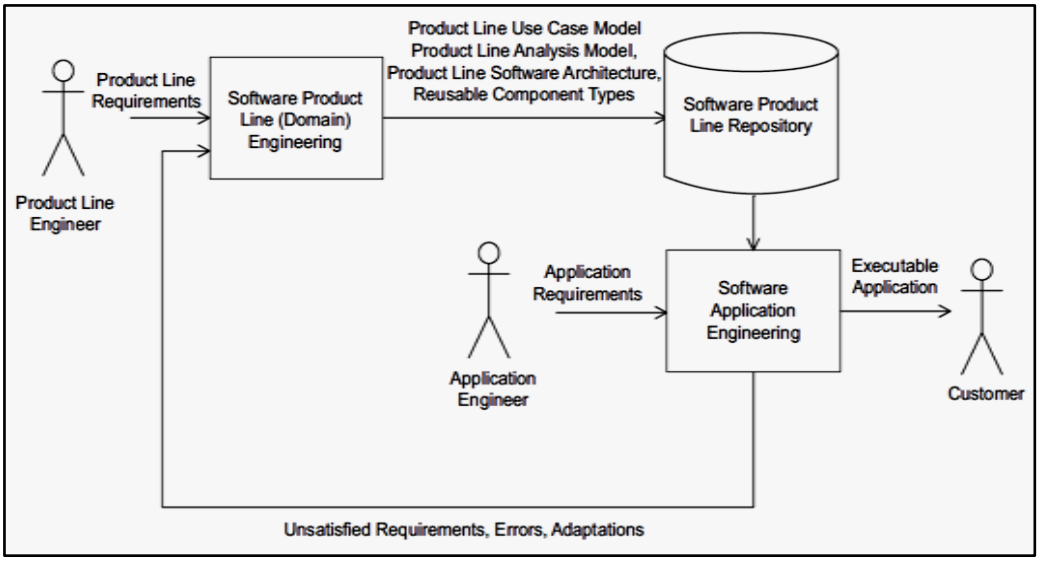


Figura 12. Modelo ESPLEP.
Fuente: Gomaa, H. (2011)

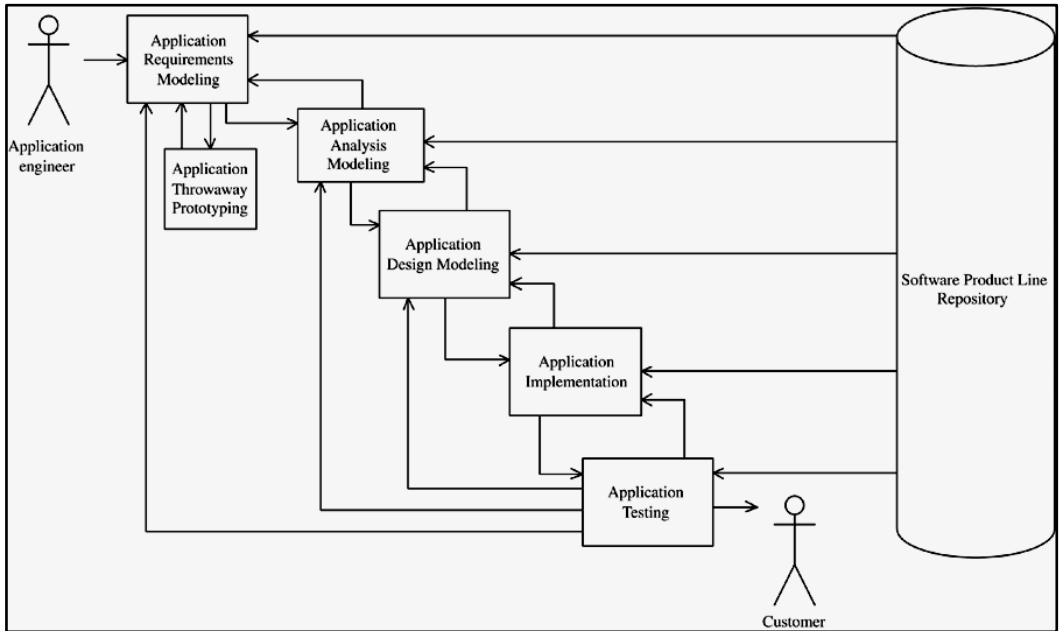


Figura 13. Fases de la Ingeniería de Línea de Productos del método ESPLEP.
Fuente: Gomaa, H. (2005).

Proceso para la Ingeniería del Dominio basado en Calidad de Software (InDoCaS)

InDoCaS es el proceso para la Ingeniería del Dominio basado en Calidad de Software desarrollado por Canelón, R. (2010), ver Figura 14, su objetivo es obtener una arquitectura base para una familia de productos, aplicando las actividades correspondientes a las disciplinas de análisis y diseño del dominio. Aunque InDoCaS fue desarrollado para la ingeniería de dominio puede ser utilizado en el enfoque de desarrollo de LPS, el cual puede ser instanciado para un dominio especifico y los activos de software producidos pueden ser reutilizados en la ingeniería de la aplicación para generar un producto específico.

La estructura del proceso está basada en lo siguiente:

- RECLAMO (Requirement Clasification Model): Modelo de clasificación de requisitos.
- ISO/IEC 25010: Estándar Internacional que describe un modelo bipartito para la calidad del producto de software.
- Un proceso para el análisis del dominio para construir el modelo de calidad.
- FODA (Feature-Oriented Domain Analisys): análisis del dominio orientado a rasgos, desarrollado por el Software Engineering Institute (SEI) para el modelo de variabilidad.
- Un proceso general para el diseño arquitectónico del dominio.
- ADD (Attribute-Driven Design Method): Método de diseño dirigido por atributos, usado para la formulación de escenarios de calidad.
- ATAM (Architecture Tradeoff Analysis Method): Método para elegir una arquitectura para un sistema, utilizado en InDoCaS para la evaluación arquitectural.

Según Canelón, R. (2010), las dos fases de InDoCaS son análisis y diseño del dominio. El análisis (ver Figura 15) permite la caracterización del dominio y definición del modelo de calidad asociado. Esta disciplina contempla las actividades de identificación de requisitos, obtención del modelo de similitudes y variabilidad,

identificación de propiedades de calidad asociadas al conjunto minimal de requisitos funcionales y no funcionales, obtención del modelo de calidad asociado al dominio, creación de escenarios de calidad del dominio e identificación de los estilos arquitecturales para el dominio. En la fase de diseño del dominio, se especializan las actividades para la síntesis (ver Figura 16) y evaluación (ver Figura 17) arquitectural del dominio, proponiendo y adaptando un conjunto de técnicas especificas para la construcción de dichas actividades y artefactos.

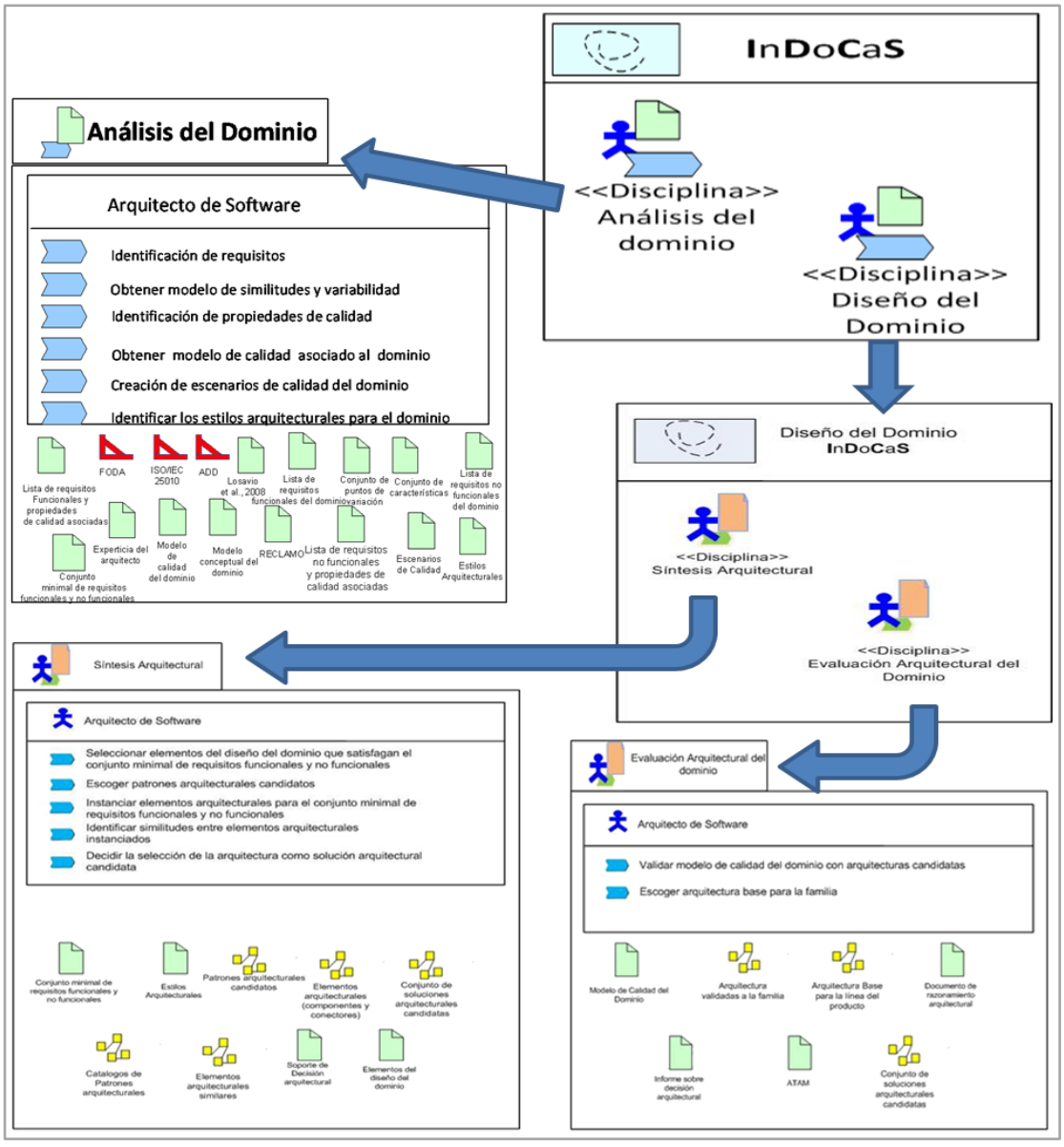


Figura 14. Proceso InDoCaS.
Fuente: Canelón, R. (2010)

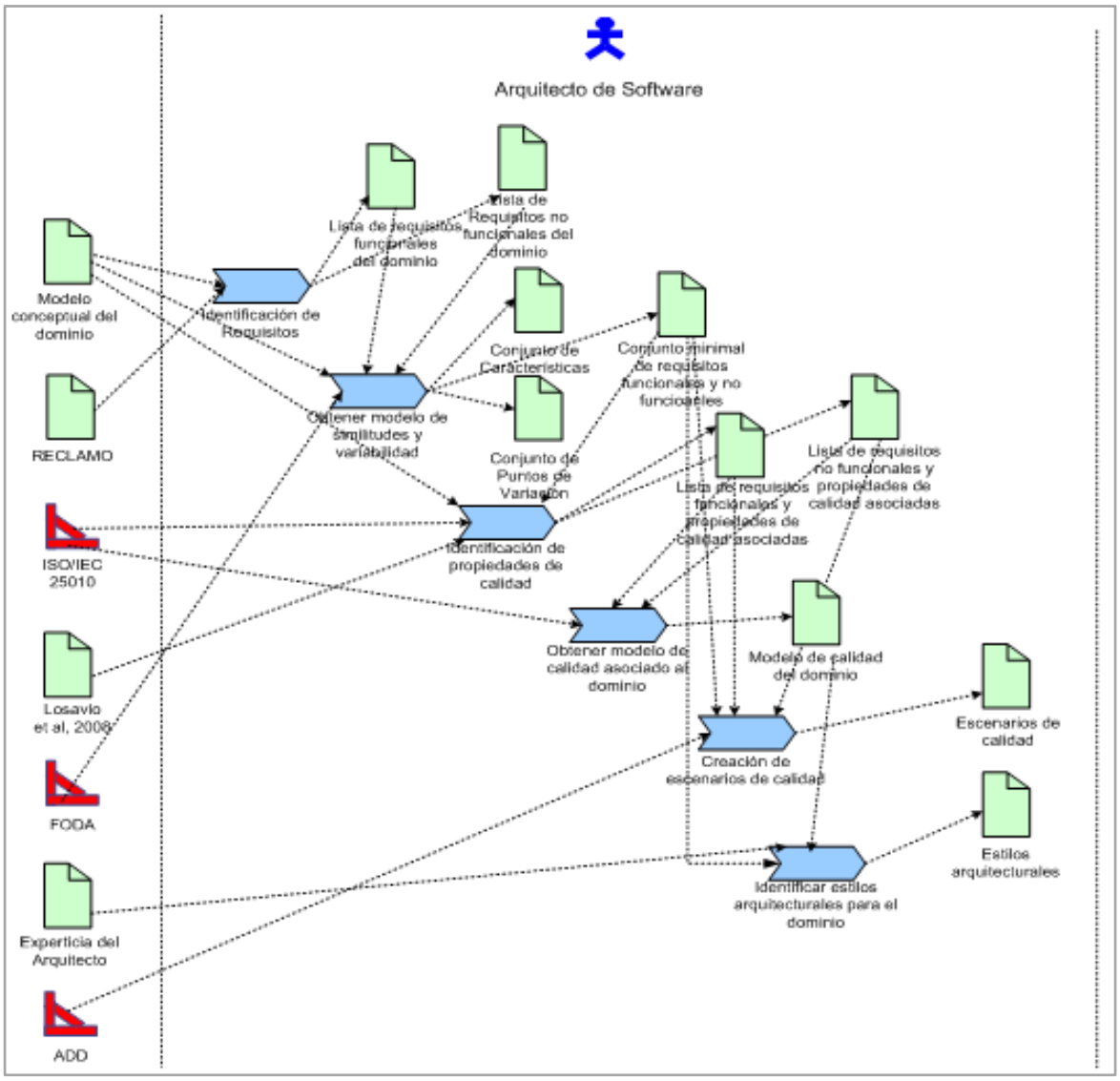


Figura 15. Actividades de la disciplina Análisis de Dominio de InDoCaS.
Fuente: Canelón, R. (2010)

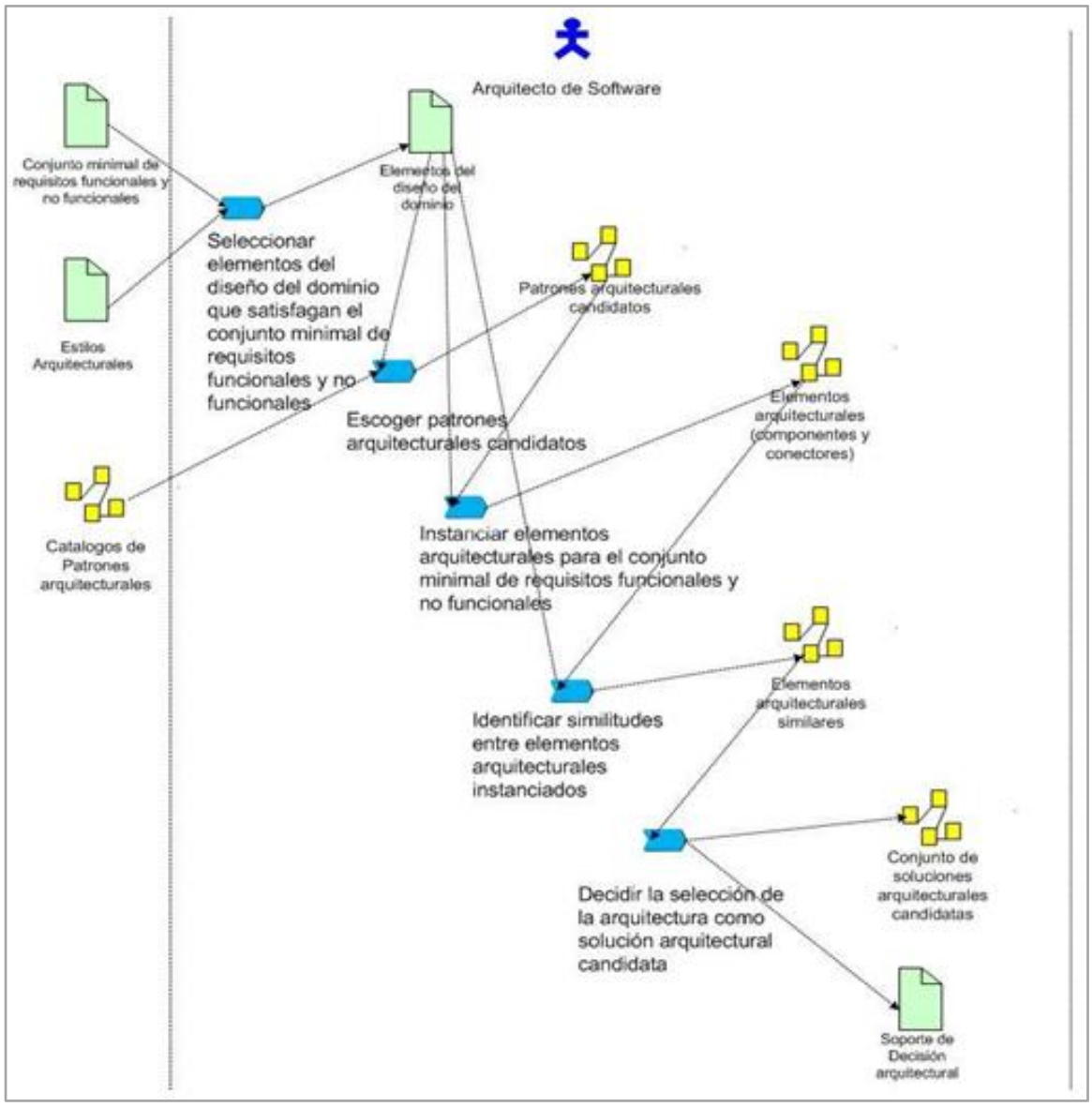


Figura 16. Actividades de la disciplina Síntesis Arquitectural de InDoCaS
Fuente: Canelón, R. (2010)

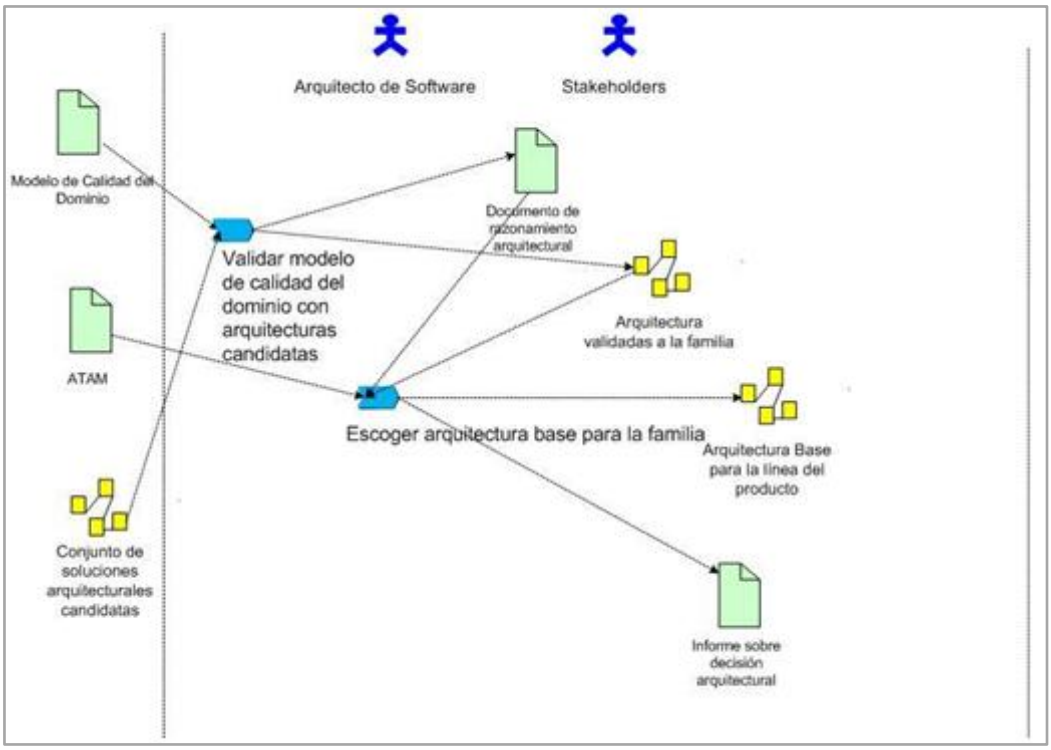


Figura 17. Actividades de la disciplina Evaluación Arquitectural de InDoCaS. Fuente: Canelón, R. (2010)

Calidad del software: Estándar ISO/IEC 25010

La calidad del software según Pressman, R., (2010) "es el cumplimiento de los requisitos de funcionalidad y desempeño explícitamente establecidos, de los estándares de desarrollo explícitamente documentados y de las características implícitas que se esperan de todo software desarrollado profesionalmente".

La Organización Internacional para la Estandarización (ISO) publicó el estándar ISO-IEC 9126-1, en el año 2001. De acuerdo a las debilidades encontradas en este estándar, se hizo una revisión y que generó el ISO/IEC 25010 publicado en octubre del 2007. El mismo, es parte de la serie de estándares SQuaRE[9] y fue preparado por el Comité Técnico Conjunto ISO/IEC JTC 1, Subcomité SC 7 de Ingeniería de Software y Sistemas.

[9] System and Software Quality Requirements and Evaluation.

El Estándar Internacional ISO/IEC 25010 describe un modelo bipartito para la calidad del producto y del proceso software, ver Figura 18.

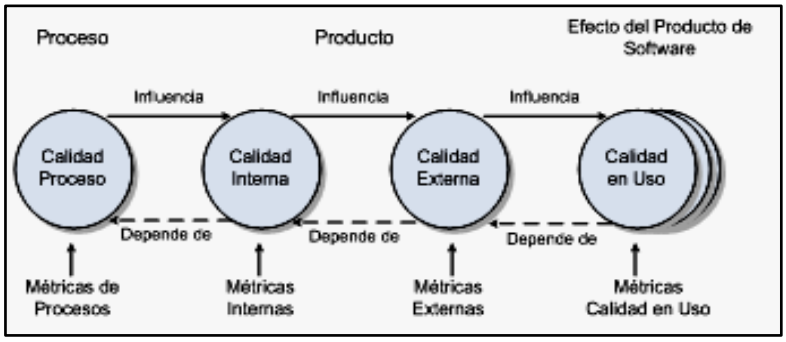


Figura 18. Enfoques de Calidad.
Fuente: ISO/IEC 25010 (2009).

Calidad interna y externa: la Calidad Interna, proporciona una visión de la "caja blanca" del software y trata las características del producto de software que están disponibles durante el desarrollo. Está relacionada con las características estáticas del software y tiene un impacto en la calidad externa del software, que tiene a su vez un impacto en la calidad funcional. Así mismo, la Calidad externa, proporciona una visión de la "caja negra" del software y trata las características relacionadas con la ejecución del software.

Calidad en el uso: Es una medida de la calidad del sistema en su ambiente operacional para usuarios específicos que realizan tareas específicas. La calidad funcional del software es la capacidad de permitir la misma en su ambiente operacional para ejecutar tareas específicas que realizan los usuarios.

La primera parte del modelo especifica ocho características para la calidad interna y externa, ver Figura 19, que se subdividen más a fondo en sub-características, que se manifiestan externamente cuando el software se utiliza como parte de un sistema informático, y es resultado de las cualidades internas del software. Este estándar internacional no elabora el modelo para la calidad interna y externa debajo del nivel de sub-características.

Figura 19. Características y subcaracterísticas de ISO/IEC 25010.
Fuente: ISO/IEC 25010: 2011.

La segunda parte del modelo especifica cinco características de la calidad en uso, es el efecto combinado para el usuario de las ocho características de la calidad del producto de software. Las características definidas son aplicables a todo tipo de software, ver Figura 20.

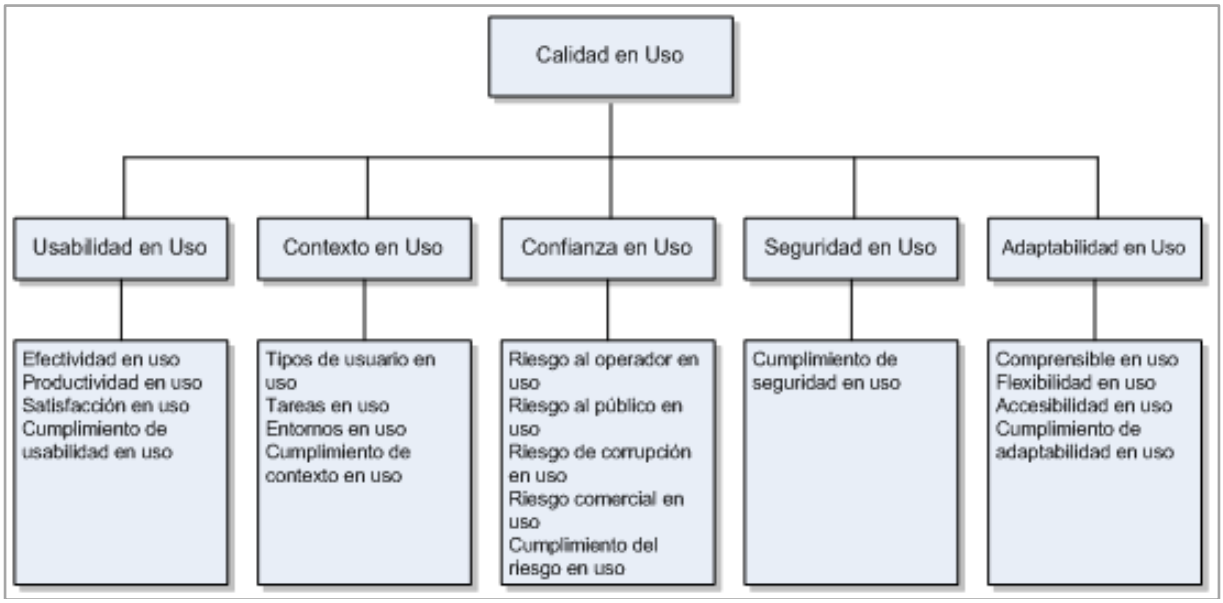


Figura 20. Características y subcaracterísticas en calidad de uso.
Fuente: ISO/IEC 25010, 2009.

Un modelo de la calidad se puede utilizar para apoyar la especificación y la evaluación del software de diversas perspectivas por aquellas personas asociadas al proceso de adquisición, requisitos, desarrollo, uso, evaluación, soporte,

mantenimiento, garantía de calidad y auditoria del software. Puede, por ejemplo, ser utilizado por los desarrolladores, compradores, personal evaluador de la garantía de calidad y evaluadores independientes, particularmente aquellos responsables de especificar y de evaluar la calidad del producto de software.

Aprendizaje electrónico o eLearning

La dinámica de la vida actual, limita el tiempo que las personas tienen para realizar ciertas actividades, por lo que internet ha brindado solución a muchas de ellas, hoy día son comunes la comunicación interpersonal (por correo electrónico, chats, video), compras por internet, transacciones bancarias online, los juegos y muchas otras, de esta manera el aprendizaje también ha tenido su espacio, ya no es obligatorio estar en un aula de clases para recibir un curso de alguna asignatura, ahora mediante el uso de internet se practica el aprendizaje electrónico o eLearning.

Pero estas aplicaciones no solo son utilizadas en instituciones educativas, Guendouz, A., y Bennouar, D. A (2013), afirman que pueden ser utilizadas por corporaciones para proporcionar capacitación para sus empleados. Adicionalmente, en una encuesta aplicada por Learning & Performance Brasil[10] arrojó como resultado que el 76% de las organizaciones considera de gran importancia integrar los sistemas de gestión de aprendizaje con otros software de administración, mientras que el 24% lo consideró importante. En otras palabras, el aprendizaje electrónico sigue siendo un tema de basta importancia en la actualidad.

De acuerdo a Soler, J., Prados, F., Poch, J. y Boada, I. (2012) las principales funciones de un sistema de gestión de aprendizaje son, entre otras: gestionar usuarios, recursos, materiales y actividades de formación, administrar el acceso, controlar y hacer seguimiento del proceso de aprendizaje, realizar evaluaciones, generar informes, gestionar servicios de comunicación como foros de discusión y videoconferencias y otras. Estas plataformas ofrecen el aprendizaje de una cierta materia a través de las técnicas habituales como pueden ser: el acceso y/o descarga de

[10] http://www.elearningbrasil.com.br/

materiales interactivos, la evaluación a través de cuestionarios con respuestas cerradas (elección múltiple, tipo test, relleno de blancos), la interacción con uno o más tutores y con el resto de alumnos.

Líneas de producto de software para el dominio eLearning

Las plataformas de eLearning son similares entre sí, soportan conceptos como actividad, entrega o calificación, sin embargo, también presentan una amplia gama de diferencias entre ellas, por ejemplo, sus esquemas de bases de datos (repositorios), aunque se refieren a los mismos conceptos, son diferentes. Lo anterior evidencia la aplicabilidad de LPS, donde se aprovechan los artefactos comunes y se maneja la variabilidad de los no comunes.

En este contexto, Sánchez et al. (2013), plantean que alrededor de estas se ha generado un nuevo mercado de aplicaciones auxiliares que amplían sus funcionalidades, por ejemplo Moodle[11] cuenta actualmente con 786 módulos complementarios y plugins en su web, por tanto, estas plataformas auxiliares deben personalizarse de acuerdo a las necesidades del cliente, y en concreto adaptarse a las diferentes plataformas de aprendizaje, a las diferentes estrategias de aprendizaje, temáticas y métodos, como también a los diferentes entornos.

Guendouz, A., y Bennouar, D. A (2013), plantean que las líneas de producto de software pueden proporcionar una solución a problemas de variabilidad que presentan los sistemas eLearning, es decir, responder a versiones para distintos niveles educativos: escuelas o universidades, o ambientes diferentes como corporaciones, en culturas diversas en muchos países, con una misma plataforma de línea de productos.

SPEM

El Metamodelo para la Ingeniería de Procesos de Software SPEM (Software Process Engineering Metamodel), es un estándar definido por el Object Management Group (OMG) para el modelado de procesos de desarrollo de sistemas de software y

[11] https://moodle.org/?lang=es

el establecimiento del lenguaje de definición de dichos procesos. SPEM representa los procesos en base a tres elementos fundamentales: los roles, los artefactos (productos) y las tareas. Los roles representan a los responsables de la acción, los artefactos son las entradas o salidas y las tareas la actividad ejecutada. Autores como Ruiz-Rube, I., Dodero, J., y Ruiz, M. (2014) consideran que "El lenguaje más utilizado por los autores para el diseño de los modelos de proceso es SPEM". La versión actual de SPEM es la 2.0., sin embargo, en el presente trabajo se usa la empleada por InDoCaS, por tratarse una extensión del mismo.

La Ciencia del Diseño

Hevner, A., March, S., Park, J. y Ram, S. (2004) afirman que dos paradigmas caracterizan gran parte de la investigación en la disciplina de los Sistemas de Información: la ciencia del comportamiento y la ciencia del diseño. La ciencia del diseño se ha vuelto notoria desde la década de los 70's cuando Herbert A. Simon en su libro "Las ciencias de lo artificial" planteó que la ciencia del diseño es posible y tiene los componentes necesarios para identificarse como una ciencia de lo artificial y merece un lugar al lado de las ciencias naturales.

Hevner, A., y Chatterjee, S. (2010, p. 5) definen la investigación en ciencias del diseño como un paradigma investigativo que permite resolver problemas humanos mediante la creación de artefactos innovadores que contribuyen al cuerpo del conocimiento del problema. Por su parte González, W. (2007, p. 3) afirma que la Ciencia del Diseño "designa conocimientos específicos que son elaborados para resolver, de manera articulada, problemas concretos que surgen en el entorno humano", además "identifica un nuevo campo del saber" y "abarca también, un conjunto de prácticas científicas propias".

Según Simon, H. (1996) el paradigma la ciencia del diseño tiene sus raíces en la ingeniería y las ciencias de lo artificial, es decir, se inclina a la resolución práctica de problemas. Algo similar asevera González, W. (2007) "Las Ciencias de Diseño ocupan un puesto particularmente próximo a la Tecnología". Específicamente en los sistemas de información Hevner et al. (2004) aseguran que la Ciencia del Diseño

aborda la investigación a través de la construcción y la evaluación de artefactos diseñados para satisfacer las necesidades de negocio.

De lo anterior, se prevé es un enfoque investigativo ideal para la ingeniería, especialmente para el espacio del software. En efecto, Winter, R. (2008) afirma que la investigación en ciencias del diseño tiene una larga tradición en Europa, y específicamente el paradigma de investigación en ciencias de diseño orientada a los sistemas de información es dominante en países de habla alemana. Por su parte, Gacenga, F., Cater-Steel, A., Toleman, M. y Tan, W-G. (2012) refieren estudios de diferentes países, donde mencionan el aspecto resaltante en cada trabajo, esto permite evidenciar lo nutrida de la investigación en la ciencia del diseño, ver Tabla 2.

Author(s)	Design & Development Focus
Cole, Purao, Rossi & Sein (2005)	Build (model, instantiate)
Hevner, March, Park & Ram (2004)	Iterative search process, artefact
March & Smith (1995)	Build
Nunamaker, Chen & Purdin (1990-1)	Understand the studied domain, application of relevant scientific and technical knowledge, creation of alternatives, and synthesis and evaluation of proposed alternative solutions
Takeda, Veerkamp, Tomiyama & Yoshikawa (1990)	Suggestion, development
Vaishnavi & Keuchler (2008; Vaishnavi y Kuechler 2009);	Suggestion, development
Walls, Widmeyer & El Sawy (1992)	Design method, meta design
Offermann, Levina, Schonherr & Bub (2009)	Design artefact, literature research
Peffers, Tuunane, Rothenberger, & Chatterjee (2008)	Design and development
Sein, Henfridsson, Purao, Rossi & Lindgren (2011)	Building, intervention and evaluation
McLaren, Head, Yuan & Chan (McLaren et al. 2011)	Design and evaluation

Tabla 2. Design step in the IS design science literature.
Fuente: Gacenga, F., Cater-Steel, A., Toleman, M. y Tan, W-G. (2012).

Modelo general de investigación en ciencias del diseño

Vaishnavi, V., y Kuechler, W. (2004) y luego en Vaishnavi, V., Kuechler, W. y Kuechler, W. Jr. (2007, p. 12) plantean un modelo general de un ciclo de investigación en ciencias de diseño, ver Figura 21. En este modelo, los diseños comienzan con el *Awareness of Problem* (conocimiento del problema), esta conciencia o conocimiento del problema puede ser también la necesidad de mejora de

una situación, la resolución de un problema y el incremento en el rendimiento de un proceso. A continuación la *Suggestion* (Propuesta), es la sugerencia de solución basada en el conocimiento o base teórica disponible del problema. Luego el *Development* (Desarrollo), es un intento en la implementación de la solución sugerida (Propuesta). Finalizado el Desarrollo, se procede a la *Evaluation* (Evaluación), es el éxito parcial o total de la implementación de la Propuesta, este éxito, se verificará de acuerdo al cumplimiento de las funcionalidades implícitas o explicitas de la Propuesta. La Propuesta, el Desarrollo y la Evaluación son frecuentemente actividades iterativas en el curso de la investigación. La Conclusión indica la finalización del proyecto y su respectivo aporte de conocimiento del problema.

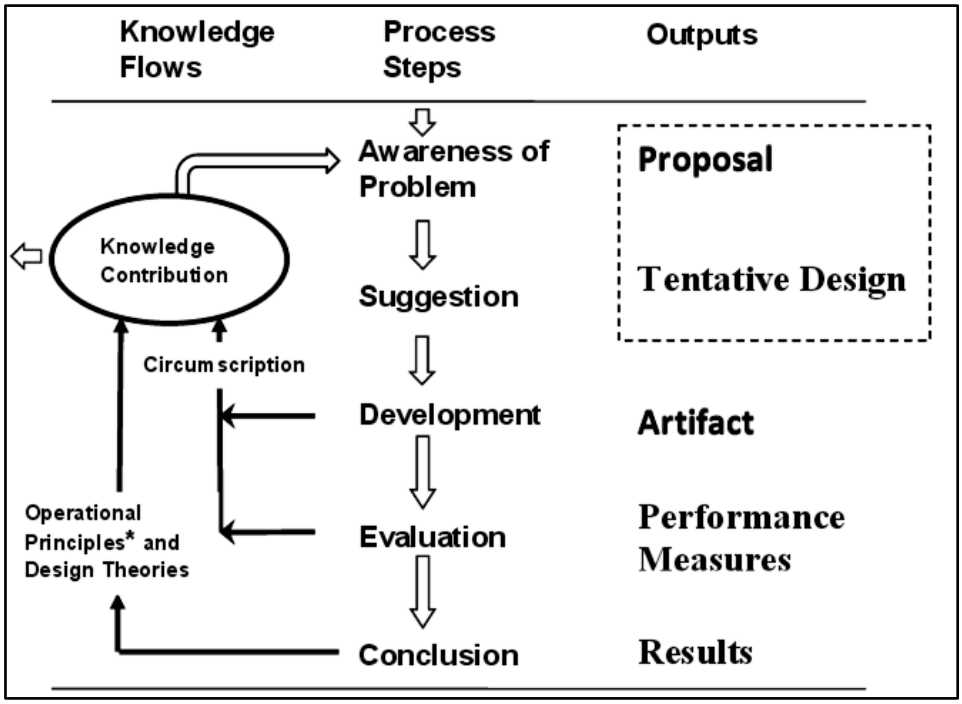


Figura 21. Modelo general de un ciclo de investigación en ciencia del diseño. Fuente: adaptación de Vaishnavi, V. y Kuechler, W. (2004).

Ingeniería de Método Situacional

De forma general los métodos son pasos sucesivos que conducen a una meta. Específicamente, Barros, O. (1995, p. 56) lo define como "un conjunto de tareas lógicamente relacionadas que existen para conseguir un resultado bien definido

dentro de un negocio; por lo tanto, toman una entrada y le agregan valor para producir una salida". Este concepto es importante para las diferentes ramas de la ingeniería, donde es común la creación o transformación de artefactos con el fin de mejorar los resultados del proceso.

La ingeniería de métodos es originada en la industria y adoptada por la ingeniería del software, en este campo Pérez, J. y Sánchez, I. (2012, p. 6) la definen "como la disciplina que apunta sus esfuerzos en aportar soluciones eficaces al diseño, el mejoramiento y la evolución de los métodos de desarrollo de los Sistemas de Información". Similarmente Brinkkemper, S. (1996, p. 276) afirma es la disciplina de la ingeniería para diseñar, construir y adaptar métodos, técnicas y herramientas para el desarrollo de sistemas de información.

Un enfoque particular de la ingeniería de métodos, es la Ingeniería de Método Situacional, según Harmsen, A. (1997) es la sub-área de la Ingeniería de Métodos dirigida hacia la construcción controlada, formal y asistida por computadora de métodos para situaciones específicas. Por su parte, Henderson-Sellers, B. y Ralyté, J. (2010, p. 429) conceptualizan a la ingeniería de método situacional como una ingeniería de métodos aplicada a una situación particular, es decir, a proyectos con características específicas. De manera semejante, Ralyté, J., Deneckère, R. y Rolland, C. (2003) opinan que promueve la construcción de un método mediante el ensamblaje de fragmentos método reutilizables almacenados en un método base.

La ingeniería de método situacional ofrece ventajas respecto a otros enfoques, por ejemplo Harmsen, A. (1997) menciona la *flexibilidad controlada*, lo que se refiere a la facilidad de construcción de métodos uniformes que pueden ser configurados y manipulados formalmente para satisfacer las exigencias organizacionales. La flexibilidad controlada también aboga por una adaptable asignación de recursos, basada en directrices de experiencias anteriores (si las hubiere).

En un artículo de Harmsen, F., Brinkkemper, S. y Oei, H. (1994) también se mencionan beneficios de la ingeniería de método situacional, particularmente para metodologías de desarrollo de sistemas de información. Como lo mencionado

anteriormente, la *flexibilidad* es un aspecto notorio, debido a que el método resultante se acopla satisfactoriamente a la situación del proyecto. Otro aspecto, es la *acumulación de experiencia* que se obtiene en la práctica de la adaptación de los componentes básicos del método, que pueden ser usados posteriormente. Adicionalmente, la *integración* de herramientas de apoyo y bloques de métodos almacenados en un repositorio común, que sustenta el desarrollo del método situacional. Y por último, la *calidad* es garantizada en el método situacional a través de la flexibilidad controlada, que endosa los requisitos de calidad del método origen, además, la estandarización de los bloques de métodos construidos facilita la medición de características de calidad.

La ingeniería de método situacional procura adaptar métodos existentes o construir nuevos, acordes a la situación actual del proyecto de desarrollo. En consecuencia, Ralyté, J., (2002) considera que el primer paso del proceso de ingeniería de método situacional es el análisis de la situación actual del proyecto de desarrollo y la identificación de los requisitos del método que soporte tal situación. El segundo paso consiste en la construcción del método, el cual definirá las estrategias a implementar, que podria ser:

- Definir un nuevo método para satisfacer una serie de requisitos situacionales;
- Agregar formas de trabajo alternativos en un método;
- Extender un método para una nueva funcionalidad;
- Para seleccionar sólo funcionalidades relevantes de un método existente.

Por su parte Harmsen, A. (1997), representa un proceso genérico de la ingeniería de método situacional (Figura 22). Este comienza con la caracterización de la situación (entorno de desarrollo del proyecto), a continuación se seleccionan (del método base) fragmentos de métodos convenientes a las características del proyecto, luego estos fragmentos son ensamblados, lo que genera un método situacional, finalmente, el desempeño del proyecto se evidencia con la experiencia de uso. Todo este proceso es dinámico y se va perfeccionando con la experiencia.

No obstante, Henderson-Sellers B., y Ralyte J. (2010, p. 447) mencionan tres enfoques de aplicación de la ingeniería de método situacional: basado en ensamblaje, en extensión o en paradigma (ver Tabla 3).

De los enfoques anteriores, se utiliza el basado en Extensión (Figura 23) para diseñar el proceso para la ingeniería de la aplicación en líneas de producto de software con enfoque de calidad (InALPro). Gehlert, A. y Pfeiffer, D. (2007) afirman que el enfoque basado en la extensión, se centra en un método existente y proporciona nuevas adiciones a la misma. El objetivo de este enfoque es mejorar un método con las nuevas características que son útiles para satisfacer las necesidades del proyecto.

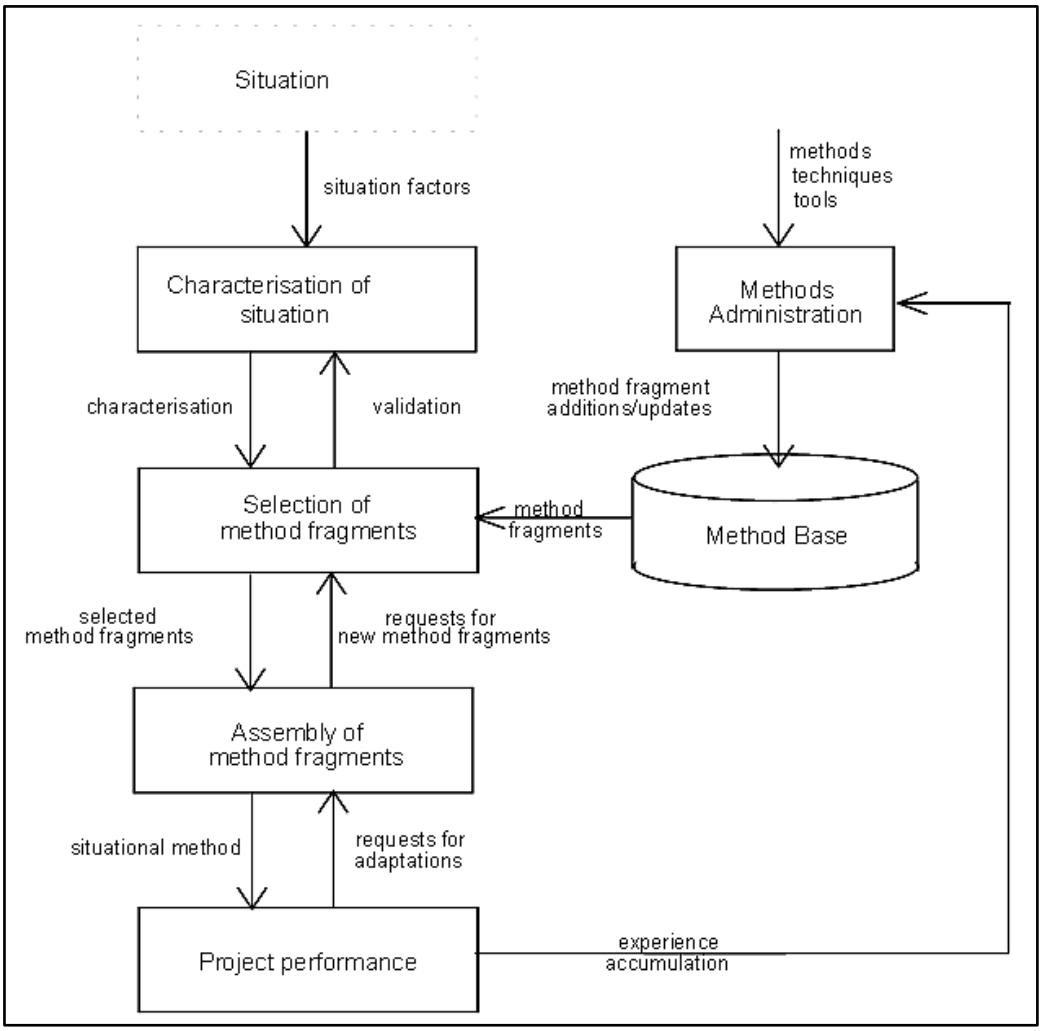


Figura 22. Proceso genérico de la Ingeniería de Método Situacional.
Fuente: Harmsen, A. (1997, p. 31)

Enfoque	Descripción
Basado en Ensamblaje	Un método nuevo se construye a partir de fragmentos de métodos ya existentes. Los fragmentos pueden ser categorizados en fragmentos de producto y fragmento de procesos. Este enfoque sigue la estrategia de reutilización.
Basado en la Extensión	El método requerido es creado a partir de un método ya existente el cual se extiende haciendo uso de patrones y rellenando las faltas del método original.
Basado en Paradigma	Este enfoque se fundamenta en el concepto que un nuevo método se obtiene a partir de una abstracción de un método existente. Se le llama también enfoque basado en la evolución.

Tabla 3. Enfoques para la creación de Métodos en la Ingeniería de Métodos Situacional. Fuente: Espinoza, G. (2012), adaptación de Henderson-Sellers B., y Ralyte J. (2010)

El enfoque de extensión de la ingeniería de método situacional, al igual que los otros enfoques, contempla como etapa inicial la caracterización de la situación donde se desarrolla el proyecto, posteriormente según Henderson-Sellers B., y Ralyte J. (2010, p. 461) se deben realizar las siguientes actividades:

1. Seleccionar un método base preexistente.
2. Integrar trozos de otros métodos para llenar las lagunas en el método base.
3. Desarrollar nuevos trozos de métodos si no es posible integrarlos de métodos existentes.

Este enfoque también es mencionado en los trabajos de: Ralyté, J., Deneckère, R. y Rolland, C. (2003), Cervera, M., Albert, M., Torres, V. y Pelechano, V. (2011, mayo) y Henderson-Sellers, B., Ralyté, J., Ågerfalk, P. y Rossi, M. (2014).

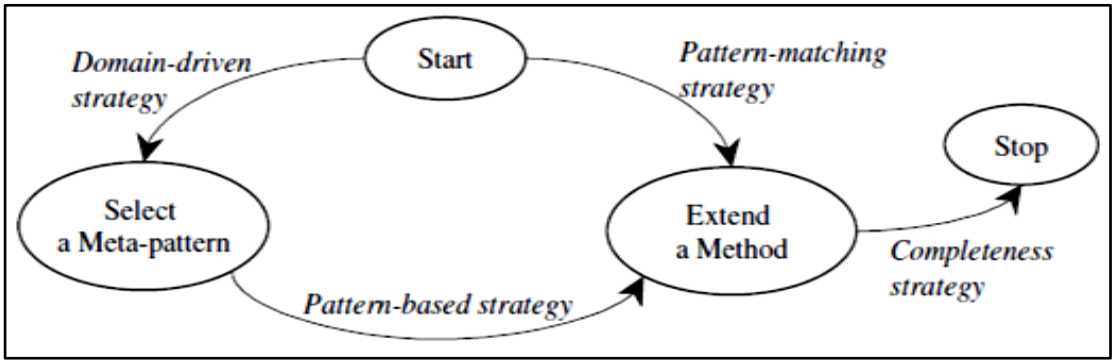


Figura 23. Enfoque basado en Extensión de la ingeniería de método situacional. Fuente: Ralyté, J., Deneckère, R. y Rolland, C. (2003)

De los aspectos revisados en las bases teóricas se considera que las LPS permiten diseñar aplicaciones eficientes, de forma económica y minimizando errores, tiempo de entrega y de mantenimiento. No obstante, los procesos de ingeniería más conocidos, no contemplan modelos bien estructurados y definidos de la disciplina de ingeniería de la aplicación, aunque sí reconocen su enorme importancia dentro del contexto de las líneas de producto.

En igual forma, la calidad es un aspecto importante en todas las etapas del desarrollo de software, contrariamente los modelos de procesos revisados no asumen explícitamente este asunto, observándose únicamente en el trabajo de Canelón, R. (2010), Rivero, L. (2011) y Gómez, E. (2012). También se observó que el dominio eLearning es sencillo, accesible y parametrizable para la implementación de una línea de producto de software.

Ahora bien, la ciencia del diseño es un paradigma de investigación novedoso, que facilita la creación de elementos innovadores y se ajusta a las características propias de la investigación en el contexto de la ingeniería del software. Y de igual manera, la ingeniería de método situacional, donde se plantean formas de ampliar el conocimiento a partir del conocimiento existente, en particular, crear nuevos métodos de métodos existentes de forma coherente.

CAPÍTULO III

MARCO METODOLÓGICO

Naturaleza de la Investigación

Al pasar del tiempo, la ciencia estipuló el desarrollo del conocimiento formal y originó los paradigmas investigativos tradicionales del presente, sin embargo, el avance tecnológico demanda nuevos procedimientos o evolución de los existentes que satisfagan nuevas necesidades, en este sentido, Cross, N., Naughton, J. y Walker, D. (1981, octubre) consideraban al método científico como un patrón de comportamiento resolutivo en descubrir la naturaleza de lo que existe, y plantearon un innovador método basado en el diseño, con la finalidad de crear cosas de valor que aún no existen.

Hevner, A., y Chatterjee, S. (2010) refieren que las actividades asociadas al diseño tienen una larga historia en muchas disciplinas como arquitectura, ingeniería, educación y otras, en la computación y tecnologías de la información no se había adoptado de lleno, hasta que en la década de 1990 comunidades científicas en Workshop on Information Technology and Systems (WITS) y Electronic Group Decision Support Systems (GDSS) de la University of Arizona, reconocieron su importancia en la investigación para la mejora de la efectividad y utilidad de artefactos de tecnología de la información.

El presente estudio está concebido dentro del enfoque de ciencia del diseño, que Simon, H. (1996) define como un paradigma de investigación pragmático que aboga por la creación de artefactos innovadores para la resolución de problemas del mundo real (p. 111). Por su parte Vaishnavi, V. y Kuechler, W. (2004) afirman que la investigación en ciencias de diseño consiste en la creación de nuevos conocimientos a través del diseño de artefactos novedosos o innovadores (cosas o procesos) y del análisis de su uso y/o desempeño.

Los impulsadores del paradigma de la ciencia del diseño plantean que al necesitarse un artefacto que no existe para satisfacer una necesidad humana, los

enfoques tradicionales no son adecuados, ya que ellos están orientados al estudio de cosas o fenómenos que existen, por esta razón se considera ideal este paradigma para el presente estudio, es decir, para el diseño de un proceso para la ingeniería de la aplicación en LPS.

Este estudio se enmarca en el estilo proyecto especial que es definido por UPEL (2006) como "Trabajos que lleven a creaciones tangibles, susceptibles de ser utilizadas como soluciones a problemas demostrados". Igualmente UPEL (Ob. Cit.) asevera que "Se incluyen en esta categoría los trabajos de elaboración de libros de texto y de materiales de apoyo educativo, el desarrollo de software, prototipos y de productos tecnológicos en general". En concordancia su objetivo principal es proponer un proceso que aporte conocimiento al área de las LPS, en particular al espacio metodológico de la ingeniería de la aplicación.

Diseño de la Investigación

Respecto al diseño de la investigación, Balestrini, M. (2002) expone "es la estrategia o plan que guía la investigación", y según Espinoza, G. (2012) "debido a las diferencias que radican en la naturaleza de las ingenierías, incluida la Ingeniería del Software, respecto de otras disciplinas empíricas y formales, dificultan la aplicación directa de los métodos de investigación tradicionales", por lo tanto, han surgido nuevas propuestas metodológicas como el modelo de proceso general para la investigación en la Ciencia del Diseño propuesto por Vaishnavi, V., y Kuechler, W. (2004) y Vaishnavi et al. (2007, p. 12), el cual es una adaptación del modelo propuesto por Takeda et al. (1990), ver Figura 24.

Este modelo es referenciado en múltiples estudios de investigación, como por ejemplo en Bukvova, H. (2009), Gacenga, F. et al. (2012), Hill, G. (2009), Offermann, P. et al. (2009) y Pfeiffer, D. (2008), además es interesante por el enfoque orientado a los sistemas de información, en consecuencia, fue seleccionado dicho procedimiento como guía para realizar la presente investigación.

Procedimiento de la Investigación

Fase 1. Conocimiento del problema (Awareness of Problem): es la fase inicial del proceso de investigativo, consiste en el descubrimiento de un problema o necesidad. En el Capítulo I, Planteamiento del Problema, se explica el objeto o problema del presente estudio.

Fase 2. Propuesta (Suggestion): es la fase siguiente al conocimiento del problema, constituye la elaboración de un modelo, diseño tentativo o prototipo de solución al problema planteado, esta solución puede ser un planteamiento completamente nuevo o una nueva funcionalidad para una propuesta existente. Para el presente estudio, la propuesta de solución está comprendida por el alcance de los objetivos de la investigación, donde el objetivo principal es proponer un proceso para la ingeniería de la aplicación en líneas de producto de software con características de calidad.

Fase 3. Desarrollo (Development): en esta fase se construye o desarrolla la propuesta, las técnicas para su implementación variaran dependiendo del artefacto que se desea crear, el artefacto puede ser un algoritmo, un método, un sistema…, la novedad del proceso está en la confección de la propuesta, la ejecución de la misma puede ser común, dependiendo del estado de la práctica del artefacto. Para efectos de la obtención de un proceso para la ingeniería de la aplicación en líneas de producto de software con características de calidad se utiliza como guía la Ingeniería de Método Situacional.

De acuerdo a Harmsen et al. (1994), la Ingeniería de Método Situacional es un tipo de ingeniería de método, conducente a la construcción de métodos para el diseño de sistemas de información en una situación específica (p. 6), por lo que se considera adecuado para el presente estudio. Dentro de la Ingeniería de Método Situacional se utiliza en particular el Enfoque Basado en Extensión, que según Henderson-Sellers, B. y Ralyté, J. (2010), es el que permite construir un nuevo método completando los vacios de un método existente con fragmentos de otros métodos y omitiendo los fragmentos irrelevantes.

Las actividades a realizar según el enfoque basado en extensión de la ingeniería de método situacional y usado en este trabajo para la construcción del proceso para la ingeniería de la aplicación en LPS son las siguientes:

1. **Caracterizar la situación:** en esta actividad se analiza la situación actual del proyecto de desarrollo y se identifican los requisitos del método que le dará soporte. En este sentido, se consideran los aspectos técnicos, económicos, humanos y metas del proyecto en sí.
2. **Seleccionar el método base:** a continuación, se determinan las ventajas y limitaciones de los métodos candidatos, lo que permite seleccionar el método base que será extendido para el proceso propuesto, y que abarca las disciplinas de análisis y diseño de la ingeniería de la aplicación para líneas de producto de software con enfoque de calidad, es preciso acotar, que el modelo de calidad a aplicar es el propuesto por Canelón, R. (2010) y usado por Rivero, L. (2011), Gómez, E. (2011) y Duarte, D. (2012). Adicionalmente se realizan las actividades mencionadas a continuación:
    a. Determinar las actividades faltantes del método base de las disciplinas de análisis y diseño de la ingeniería de la aplicación en líneas de producto de software (flujo de trabajo del método propuesto).
    b. Identificar los artefactos involucrados en las disciplinas de análisis y diseño de la aplicación en LPS.
    c. Identificar los fragmentos de métodos que permitirán extender o especializar el método base.
    d. Elaborar el modelo preliminar del proceso de ingeniería de la aplicación en líneas de producto de software con enfoque de calidad (InALPro).
3. **Integrar los trozos de otros métodos:** determinadas las actividades y artefactos de las disciplinas del proceso InALPro, se adaptan los fragmentos de método y define el flujo del proceso, las responsabilidades de sus actores y artefactos de entrada y salida, de igual manera se incorporan al proceso InALPro las actividades necesarias para ajustar el modelo de calidad del dominio según los requisitos de calidad específicos de la aplicación.

Posteriormente, se ejemplifica el proceso propuesto en el dominio eLearning, donde inicialmente se aplica el proceso InDoCaS para obtener los artefactos resultantes de la ingeniería de dominio, y que son insumo para el proceso InALPro, de esta manera se obtiene una propuesta metodológica de línea de producto software con enfoque de calidad. En la Figura 24, se presenta el gráfico del procedimiento de investigación.

Fase 4. Evaluación (Evaluation): una vez construido el artefacto se evalúa con los criterios formulados en la propuesta, se observan las desviaciones cuantitativas y cualitativas de las expectativas, es decir, se analiza el comportamiento del artefacto para juzgar si cumple con lo estipulado en la propuesta, es un proceso equivalente a la verificación de la hipótesis en el enfoque positivista. Las posibles desviaciones se corrigen y generan conocimiento que retroalimenta la conciencia del problema.

El proceso para la ingeniería de la aplicación en líneas de producto de software con características de calidad, planteado en esta investigación se ejemplifica en un dominio de aplicación para reflexionar sobre el alcance de los objetivos propuestos. Esta reflexión se toma como la fase de evaluación del método, verificando si se obtienen los resultados esperados.

Fase 5: Conclusión: esta fase puede ser el fin de un ciclo investigativo, el cual genera conocimiento a la conciencia del problema (permite corregir) o el fin de un esfuerzo de investigación, que no significa que el artefacto diseñado sea perfecto, pero si lo suficientemente bueno como solución al problema planteado. Para efectos de esta investigación, se consideraran como conclusiones las realizadas en el capítulo cinco de este trabajo.

La ciencia del diseño aplicada en esta investigación generó un conocimiento que enriquece la conciencia del problema, así que, para trabajos futuros es recomendable sean utilizados los artefactos aquí producidos (modelo de calidad de la aplicación, casos de uso de la aplicación y arquitectura para la aplicación) para la definición de la disciplina de implementación de la aplicación.

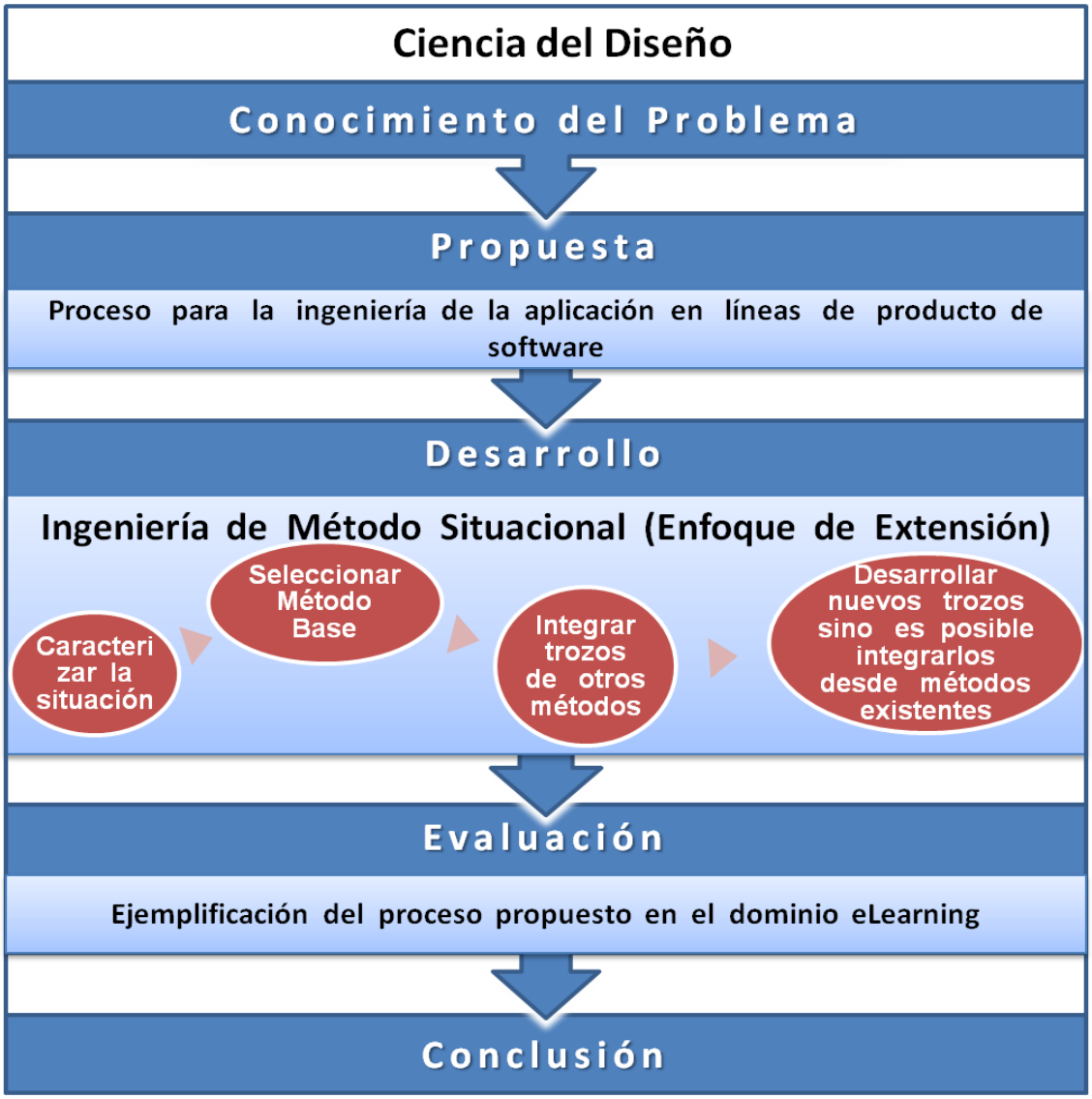


Figura 24. Procedimiento de la investigación.
Fuente: Autora de la investigación (2015)

CAPÍTULO IV

LA PROPUESTA

Descripción de la Propuesta

Se realizó un estudio en el contexto del paradigma investigativo ciencia del diseño, donde se conoció el problema de los inadecuados y escasamente explicados métodos para la ingeniería de la aplicación en LPS, por lo que se propone un *Proceso para la ingeniería de la aplicación en líneas de producto de software (InALPro)*, para lo anterior se utiliza el enfoque de extensión de la ingeniería de método situacional, donde es seleccionado el proceso InDoCaS como método base y se integran fragmentos del método Gray Watch dando como resultado el proceso InALPro, este proceso se distingue de otros por contemplar explícitamente características de calidad en su ejecución. Finalmente, se ejemplifica el proceso InALPro en el dominio eLearning de la UPTP "Juan de Jesús Montilla". En la Figura 25, se muestra gráficamente la propuesta de investigación.

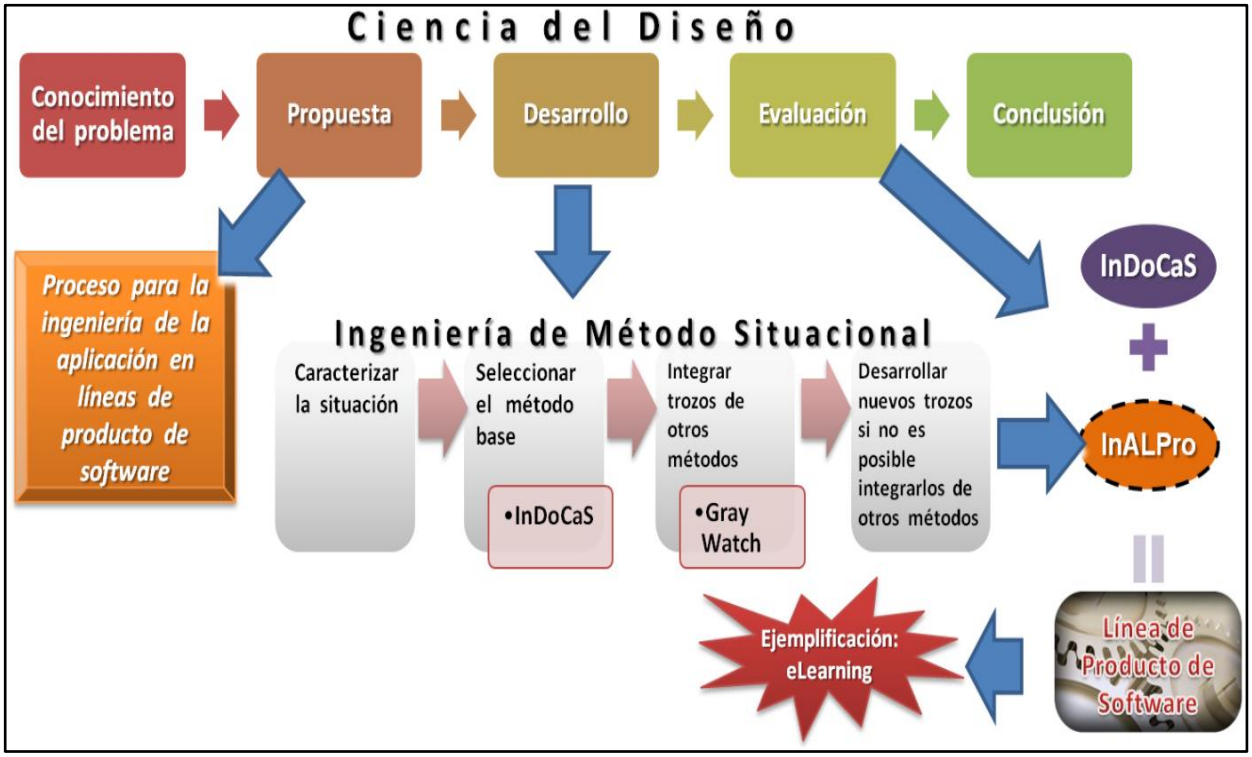


Figura 25. Propuesta de investigación.
Fuente: Autora de la investigación (2015).

Para aplicar la ingeniería de método situacional es pertinente realizar unos pasos previos, como lo son los indicados en la Figura 26.

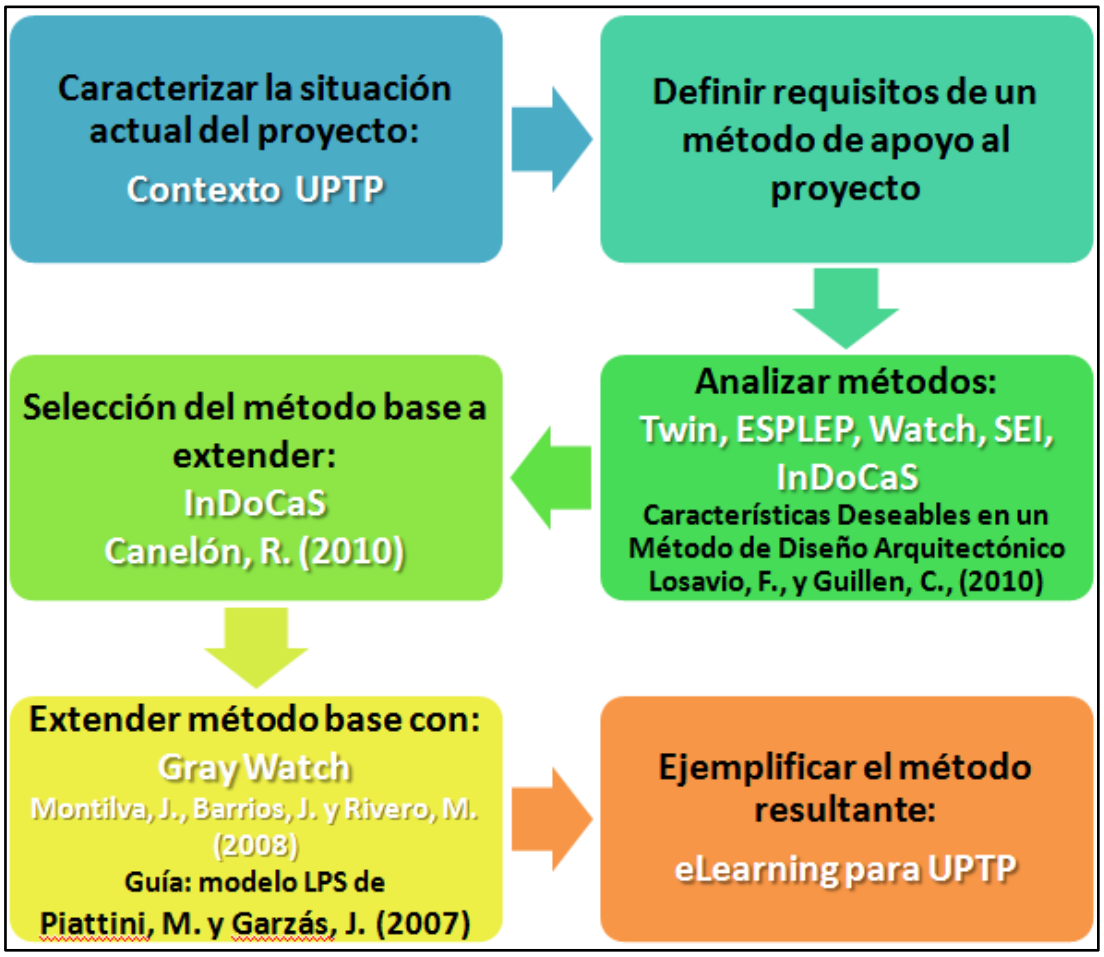


Figura 26. Adaptación de las actividades de la ingeniería de método situacional.
Fuente: Autora de la investigación (2015).

Caracterización de la Situación del Proyecto.

Es el primer paso de la ingeniería de método situacional, consiste en definir el contexto donde se implementa el método. El contexto es la Universidad Politécnica Territorial del estado Portuguesa "Juan de Jesús Montilla", UPTP, este fue seleccionado porque dentro del plan estratégico institucional se encuentra planificada una plataforma eLearning, la misma daría apoyo a las actividades académicas. En este sentido, Sánchez et al. (2013) consideran que las plataformas eLearning genéricas no necesariamente satisfacen a todas las instituciones, debido a las

necesidades variables de los clientes, donde por lo general, quedan requisitos vacios y otros disponibles que no se usan.

La UPTP “Juan de Jesús Montilla”, es una institución pública de educación superior, donde se imparten programas nacionales de formación (PNF) en Administración, Informática, Electricidad, Mecánica, Mantenimiento y Agroalimentación. Estos PNF están homologados a nivel nacional entre las universidades politécnicas territoriales y los institutos y colegios universitarios públicos (28 instituciones en total).

En la UPTP se utiliza software diseñado en la institución y por terceros, en ambos casos los usuarios manifiestan problemas como retraso en la entrega de nuevas aplicaciones, notable concurrencia de defectos del software, los sistemas no ejecutan todos los procesos que se requieren, existen sistemas que deben ser migrados a plataformas actuales.

En vista de los planteamientos anteriores, se consultó al personal de la unidad de sistemas donde se detectó que no se utiliza método de desarrollo de software, tampoco son aplicados enfoques de calidad en el proceso y al producto, la reutilización de software se realiza copiando y adaptando aplicaciones existentes a nuevas necesidades, el personal de la unidad es escaso y el volumen de trabajo es excesivo, estos dos últimos aspectos se escapan del alcance de este estudio.

De todo lo anterior, se plantea la propuesta de un método de trabajo para la unidad de sistemas, fundamentado en el modelo de líneas de producto software, para lo cual, se extiende el proceso InDoCaS, el dominio es una línea de productos software de eLearning. En la Figura 27, se muestran las características necesarias en el contexto de implementación del método.

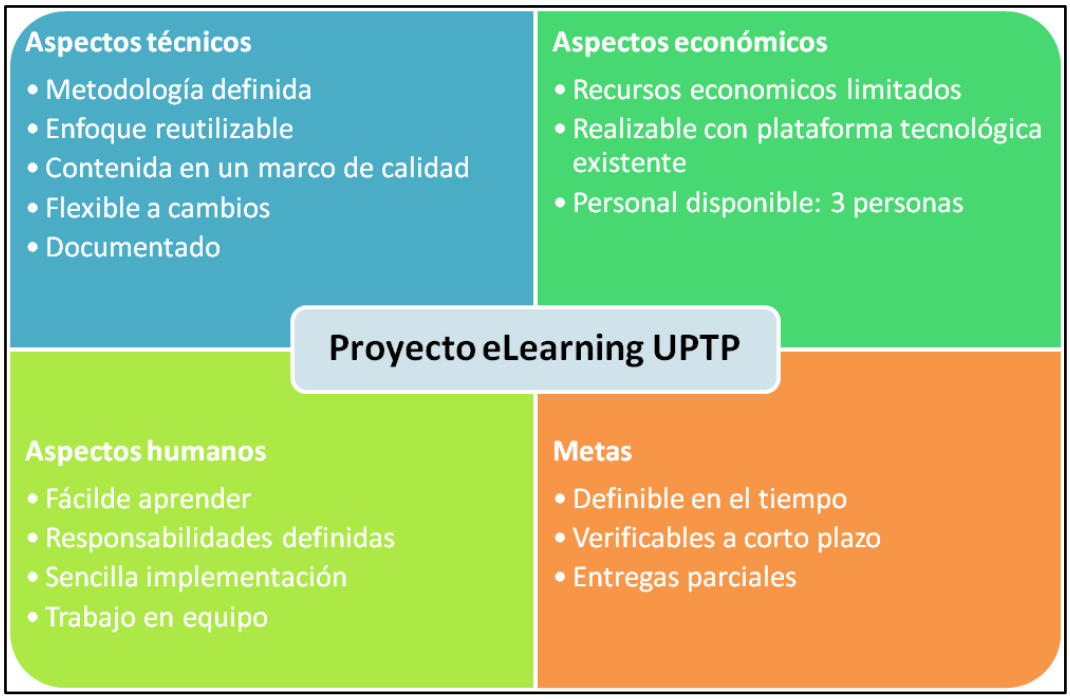


Figura 27. Caracterización de la situación del proyecto.
Fuente: Autora de la investigación (2015).

De lo anterior, se formulan los siguientes requisitos del método para el proyecto:

- Definido en el tiempo y con puntos de control.
- Contemplar una metodología con actividades comprensiblemente organizadas.
- Generar una plataforma de software reutilizable y de licencia libre.
- Enmarcado dentro de parámetros de calidad.
- Las responsabilidades de los actores del proceso deben estar definidas.

Selección del Método Base

El siguiente paso es analizar métodos conocidos de LPS, y seleccionar el método base. Para esta selección, es analizado el perfil arquitectural de los métodos conocidos y luego son comparados con el patrón del proceso estándar de una línea de producto software.

Análisis del perfil arquitectural de los métodos estudiados

Para esto se utilizó las Características Deseables en un Método de Diseño Arquitectónico propuesto por Losavio, F., y Guillen, C., (2010), estas características se consideran presente (1) y ausente (0). Ver Tabla 4.

Característica/Método	Gray Watch	Twin	ESPLEP	Modelo del SEI	InDoCaS
Poseer un modelo conceptual de la arquitectura.	**1**	**1**	**1**	**1**	**1**
Tratar explícitamente los requisitos no funcionales.	**1**	**0**	**0**	**1**	**1**
Proporcionar mecanismos de derivación arquitectónica.	**1**	**1**	**1**	**1**	**1**
Aplicar mecanismos de descripción arquitectónica con el nivel de abstracción adecuado.	**1**	**1**	**1**	**1**	**1**
Tener como objetivo el logro de cualquier combinación de características de calidad en general.	**0**	**0**	**0**	**0**	**1**
Proporcionar actividades de negociación entre los especialistas involucrados (Stakeholders).	**1**	**0**	**1**	**1**	**1**
Proporcionar actividades de transformación arquitectónica para satisfacer el modelo de calidad del sistema.	**1**	**1**	**0**	**1**	**1**
Estar apoyado por herramientas de software que auxilien al arquitecto en la recolección de datos.	**0**	**0**	**0**	**1**	**0**
Estar validado.	**1**	**1**	**1**	**1**	**1**
Ser fácil de usar.	**1**	**1**	**0**	**0**	**1**
Total características presentes	**8**	**6**	**5**	**8**	**9**

Tabla 4. Adaptación de Características Deseables en un Método de Diseño Arquitectónico. Fuente: Losavio, F., y Guillen, C., (2010).

De la tabla anterior se puede afirmar que los métodos con un mejor perfil arquitectónico son InDoCaS, Gray Watch y SEI, debido a que presentan mayor cantidad de características deseables. No obstante, el proceso InDoCaS cuenta

arquitecturalmente con la mayor cantidad de características deseables para el proyecto planteado, es adaptable a la situación del proyecto, único en manejar aspectos de calidad explícitos y sus artefactos de software pueden ser reutilizados; es por tanto, elegido como método base para ser extendido

Integración de trozos de otros métodos

A continuación, se toma un patrón que guie la integración de los trozos de otros métodos que permitirán extender el método base, este debe indicar actividades, artefactos de entrada y salida. El patrón utilizado es el modelo general de línea de producto de software planteado por Piattini, M. y Garzás, J. (2007), ver Figura 28Figura 28, este modelo permite visualizar las disciplinas de la ingeniería de la aplicación. Dicho modelo se usa para comparar las disciplinas de cada uno de los métodos analizados. En la Tabla 5, se resumen las disciplinas y artefactos constitutivos de cada método.

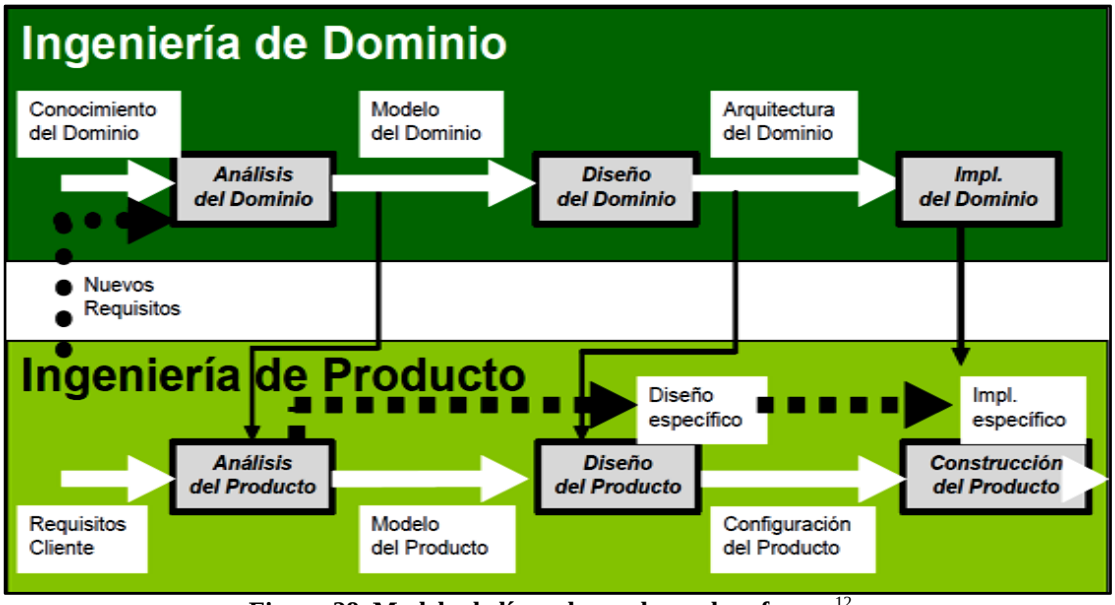


Figura 28. Modelo de línea de producto de software[12].
Fuente: Piattini, M. y Garzás, J. (2007).

[12] Los autores denominan producto a la aplicación, para efectos de esta investigación, ingeniería del producto es equivalente a ingeniería de la aplicación.

Método/ Proceso	Disciplinas del proceso de análisis y diseño de la aplicación	Artefactos involucrados	Observaciones
Gray Watch	*-Análisis-* Modelado de Negocios. Ingeniería de Requisitos. *-Diseño-* Diseño Arquitectónico. Especificación de Componentes	*Entrada:* Conocimiento del negocio. *Salida:* Modelo del negocio. Documento de requisitos. Documento de diseño.	No indica explícitamente que son las disciplinas de análisis y diseño, sin embargo, Pérez, J. y Sánchez, I. (2012) así lo reconocen.
Twin	Análisis y especificación de requisitos. Diseño de la arquitectura de la aplicación.	*Entrada*: Especificación de Requisitos. Modelo de Análisis. Diseño genérico. Componentes. *Salida*: Diseño de la arquitectura.	Los artefactos de entrada se usan a lo largo de todo el proceso, y no se indican los subproductos intermedios.
ESPLEP	*-Análisis-* Modelado de requisitos. Modelado de análisis.	*Entrada*: Requisitos de la aplicación. Modelo arquitectural de la línea. *Salida*: Modelo de requisitos de la aplicación. Modelo de análisis de la aplicación.	
	-Diseño- Modelado de diseño	*Entrada*: Modelo de requisitos de la aplicación. Modelo de análisis de la aplicación. *Salida*: Modelo arquitectural del producto.	
Modelo del SEI	Análisis de componentes específicos que serán fabricados, comprados o reutilizados.	*Entrada*: Especificación de componentes. Información sobre la capacidad de fabricación y/o adaptación de componentes y plan de producción. Disponibilidad del componente en el mercado. Estrategia de adquisición. *Salida*: Componente. Sistema integrado. Pruebas del software.	No contempla disciplinas específicas para la ingeniería de la aplicación, es un solo proceso que se repite para generar los artefactos específicos para el producto.
InDoCaS	Se realiza de la derivación de la arquitectura de referencia del dominio	*Entrada*: arquitectura de referencia de la línea de productos.	No contempla disciplinas específicas para la ingeniería de la aplicación.

Tabla 5. Comparativo de las disciplinas y artefactos de la ingeniería de la aplicación en líneas de producto de software.
Fuente: Autora de la investigación (2015).

Comparadas las disciplinas y artefactos anteriores, se eligen los trozos (más convenientes) de método para extender el método base (InDoCaS), en el Análisis de la Aplicación es seleccionado el fragmento de Ingeniería de Requisitos de Gray Watch y se omite el Modelado de Negocio ya contemplado por InDoCaS. Para el diseño de la Aplicación se optó por el Diseño Arquitectónico también de Gray Watch, y se omite la Especificación de Componentes por estar fuera del alcance de este estudio.

La tercera fase de la ingeniería de método situacional, es decir, el Desarrollo de nuevos trozos de método, no se ejecutó por considerarse innecesaria para este proyecto, esto motivado a que el proceso Gray Watch proporcionó los trozos de método adecuados para la extensión.

La Figura 29 muestra gráficamente la propuesta con la notación SPEM, proceso para la ingeniería de la aplicación con enfoque de calidad para líneas de producto de software, denominada InALPro.

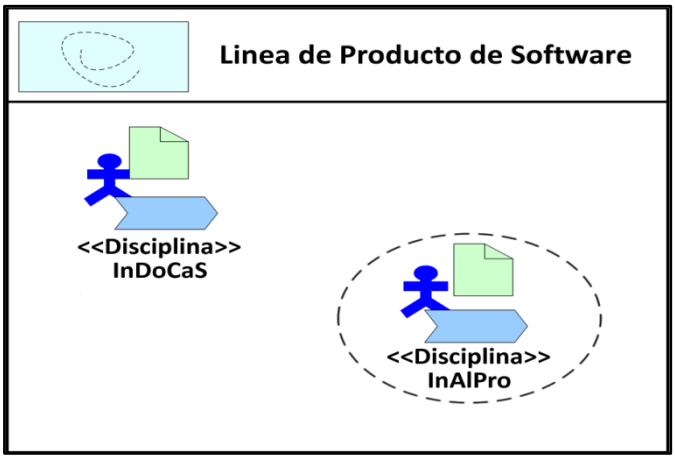


Figura 29. Proceso InALPro en el contexto de línea de producto de software.
Fuente: Autora de la investigación (2015).

El proceso InALPro consta de dos disciplinas instanciadas del Método Gray Watch, la Ingeniería de Requisitos y el Diseño Arquitectónico, los cuales toman artefactos producidos por InDoCaS para especializarlos a un producto específico. A partir de aquí, los procesos instanciados se denominarán Ingeniería de Requisitos de la Aplicación y Diseño Arquitectónico de la Aplicación, ver Figura 30.

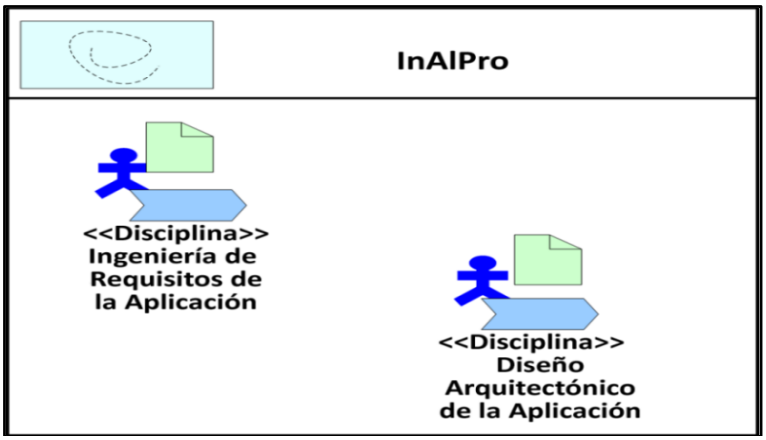


Figura 30. Proceso propuesto InALPro.
Fuente: Autora de la investigación (2015).

Siguiendo la notación de InDoCaS, en la Tabla 6 se presentan las actividades a realizar en el proceso InALPro.

InALPro					
ENTRADA	Información del negocio, objetivos y limitaciones de la aplicación. Artefactos de InDoCaS				
DISCIPLINA	**ACTIVIDADES**			**ARTEFACTOS**	
	Nro.	**Tablas**	**Nombre de la actividad**	**Nro.**	**Tablas**
Ingeniería de Requisitos de la Aplicación	A_14	7	Análisis de Requisitos de la Aplicación	21,22,23,24, 25,26,27,28	8,9,10,11,12, 13,14,15
	A_15	16	Validación de Requisitos de la Aplicación	29,30	17,18
Diseño Arquitectónico de la Aplicación	A_16	19	Revisión de la arquitectura de referencia	31	20
	A_17	21	Ajustar la arquitectura de referencia	32,33	22,23
	A_18	24	Evaluar la arquitectura propuesta para la aplicación	34,35	25,26

Tabla 6. Actividades propuestas para el proceso InALPro.
Fuente: Autora de la investigación (2015).

Definición de actividades y artefactos de InALPro

El enfoque de extensión de la Ingeniería de Método Situacional permitió extender el proceso InDoCaS, dando origen a InALPro, para esta extensión se usaron actividades de dos disciplinas del método Gray Watch de Montilva, J., Barrios, J. y Rivero, M. (2008). Estas disciplinas son Ingeniería de Requisitos y Diseño Arquitectónico.

Disciplina Ingeniería de Requisitos de la Aplicación

La Ingeniería de Requisitos de la Aplicación, identifica los requisitos funcionales y no-funcionales específicos del producto software, los valida con la Lista de Requisitos (funcionales y no-funcionales) con sus propiedades de calidad del dominio elaborado por InDoCaS. Si son encontrados requisitos que no fueron contemplados en el dominio, se envía esta información a InDoCaS para enriquecer el dominio (artefacto opcional útil para otra instanciación del dominio).

En la Figura 31, el diagrama de actividades propuesto de esta disciplina.

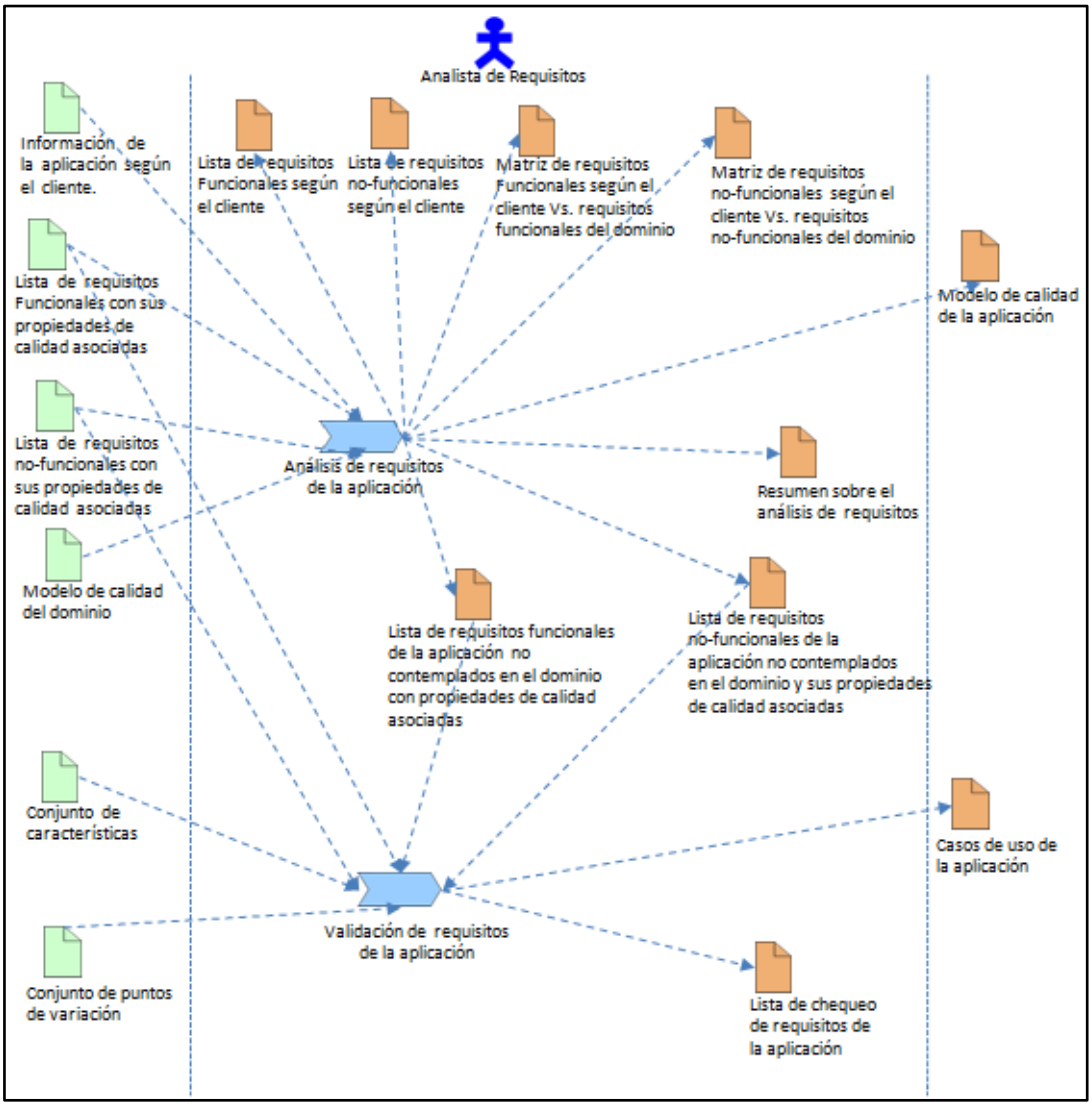


Figura 31. Disciplina Ingeniería de Requisitos de la Aplicación.
Fuente: Autora de la investigación (2015).

A continuación las actividades de la Disciplina Ingeniería de Requisitos de la Aplicación:

Actividad A_14: Análisis de Requisitos de la Aplicación.

Esta actividad define los requisitos funcionales y no-funcionales de la aplicación (requisitos según el cliente o interesado), por lo que se generan la Lista de requisitos funcionales según el cliente (artefacto 21) y la Lista de requisitos no-funcionales según el cliente (artefacto 22), estas dos listas son adaptación del artefacto 1 de InDoCaS.

Con las listas anteriores se construye una Matriz de requisitos Vs. requisitos con la Lista de requisitos funcionales según el cliente (artefacto 21) y los requisitos de la Lista de requisitos funcionales con sus propiedades de calidad asociadas de InDoCaS (artefacto 6), la matriz anterior corresponde al artefacto 23 de InALPro. Similarmente se construye otra Matriz de requisitos Vs. requisitos con la Lista de requisitos no-funcionales según el cliente (artefacto 22) y los requisitos de la Lista de requisitos no-funcionales con sus propiedades de calidad asociadas de InDoCaS (artefacto 7), esta matriz es el artefacto 24 de InALPro.

Con las dos matrices anteriores, se chequea que los requisitos funcionales y no-funcionales según el cliente estén contenidos dentro de los requisitos del dominio. Si todos los requisitos según el cliente están contemplados en los requisitos del dominio, la Lista de requisitos funcionales con sus propiedades de calidad asociada (artefacto 6 de InDoCaS) se utiliza como Lista de requisitos funcionales de la aplicación con sus propiedades de calidad asociadas y la Lista de requisitos no-funcionales con sus propiedades de calidad asociada (artefacto 7 de InDoCaS) se utiliza como Lista de requisitos no-funcionales de la aplicación con sus propiedades de calidad asociadas, y se continua con la actividad Validación de Requisitos (A_15).

En caso contrario, los requisitos funcionales no contemplados en el dominio se registran en la Lista de requisitos funcionales de la aplicación no contemplados en el dominio con sus propiedades de calidad asociada (artefacto 25). Y los requisitos no-funcionales no contemplados en el dominio se registran en la en Lista de requisitos no-funcionales de la aplicación no contemplados en el dominio con sus propiedades

de calidad asociada (artefacto 26). Los artefactos 25 y 26 son una adaptación los artefactos Nro. 6 y 7 de InDoCaS.

Las dos listas anteriores serán suministradas a la actividad Identificación de requisitos de InDoCaS (A_01), para incorporarlas al dominio y generar nuevas listas de requisitos funcionales/no-funcionales con características de calidad asociadas. Los cambios en los requisitos funcionales/no-funcionales de la aplicación, se reflejaran en el Modelo de calidad del dominio (artefacto 8 de InDoCaS), originando el Modelo de calidad de la aplicación (artefacto 27).

Los ajustes y decisiones tomadas respecto a la actividad A_14: Análisis de Requisitos de la Aplicación se refleja en el Resumen sobre el Análisis de Requisitos de la Aplicación (artefacto 28).

Esta actividad se resume en la Tabla 7.

InALPro	
ACTIVIDAD	**DESCRIPCIÓN**
Nombre de la Actividad:	Análisis de Requisitos de la Aplicación.
Responsable:	Analista de Requisitos.
Objetivo:	Verificar los requisitos de la aplicación.
Nro. Identificación:	A_14
Tipo de documento generado:	Tabla, Documento
Técnica(s) utilizada(s):	Adaptación de los subprocesos de descubrimiento y análisis de requisitos del Método Gray Watch.
Artefacto(s) de entrada:	Información de la aplicación según el cliente. Lista de requisitos funcionales con sus propiedades de calidad asociada (artefacto 6 de InDoCaS). Lista de requisitos no-funcionales con sus propiedades de calidad asociada (artefacto 7 de InDoCaS). Modelo de calidad del dominio (artefacto 8 de InDoCaS)
Artefacto(s) de salida:	-Lista de requisitos funcionales según el cliente (artefacto 21) -Lista de requisitos no-funcionales según el cliente (artefacto 22) -Matriz de requisitos funcionales según el cliente Vs. requisitos funcionales del dominio (artefacto 23). -Matriz de requisitos no-funcionales según el cliente Vs. requisitos no-funcionales del dominio (artefacto 24). -Lista de requisitos funcionales de la aplicación no contemplados en el dominio con sus propiedades de calidad asociadas (artefacto 25). -Lista de requisitos no-funcionales de la aplicación no contemplados en el dominio con sus propiedades de calidad asociadas (artefacto 26). -Modelo de calidad de la aplicación (artefacto 27) -Resumen sobre el Análisis de Requisitos de la Aplicación (artefacto 28)

Tabla 7. Actividad A_14: Análisis de Requisitos de la Aplicación.
Fuente: Autora de la investigación (2015).

La salida de esta actividad está constituida por la Lista de requisitos funcionales según el cliente (artefacto 21), Lista de requisitos no-funcionales según el cliente (artefacto 22), Matriz de requisitos funcionales según el cliente Vs. requisitos funcionales del dominio (artefacto 23), Matriz de requisitos no-funcionales según el cliente Vs. requisitos no-funcionales del dominio (artefacto 24), Lista de requisitos funcionales de la aplicación no contemplados en el dominio con sus propiedades de calidad asociadas (artefacto 25), Lista de requisitos no-funcionales de la aplicación no contemplados en el dominio con sus propiedades de calidad asociadas (artefacto 26), el Modelo de calidad de la aplicación (artefacto 27) y el Resumen sobre el Análisis de Requisitos de la Aplicación (artefacto 28), donde ya están contemplados los requisitos específicos de la aplicación. Ver Tabla 8, Tabla 9, Tabla 10, Tabla 11, Tabla 12, Tabla 13, Tabla 14 y Tabla 15.

InALPro	
ARTEFACTO	**DESCRIPCIÓN**
Nombre	Lista de requisitos funcionales según el cliente.
Constructor	Analista de requisitos
Nro. Identificación ACTIVIDAD	A_14
Nro. Identificación ARTEFACTO	21

Formato Asociado:

Nro. Identificación	Descripción del requisito funcional

Tabla 8. Artefacto 21: Lista de requisitos funcionales según el cliente.
Fuente: Autora de la investigación (2015).

InALPro	
ARTEFACTO	**DESCRIPCIÓN**
Nombre	Lista de requisitos no-funcionales según el cliente.
Constructor	Analista de requisitos
Nro. Identificación ACTIVIDAD	A_14
Nro. Identificación ARTEFACTO	22

Formato Asociado:

Nro. Identificación	Reglas del negocio asociadas a la aplicación	Descripción del requisito funcional

Tabla 9. Artefacto 22: Lista de requisitos no-funcionales según el cliente.
Fuente: Autora de la investigación (2015).

InALPro	
ARTEFACTO	**DESCRIPCIÓN**
Nombre	Matriz de requisitos funcionales según el cliente Vs. requisitos funcionales del dominio.
Constructor	Analista de requisitos
Nro. Identificación ACTIVIDAD	A_14
Nro. Identificación ARTEFACTO	23

Formato Asociado:

Requisitos funcionales según el cliente. / **Requisitos funcionales del dominio.**		**Nro. Identificación**			
		Descripción del requisito funcional			
Nro. Identificación	**Descripción del requisito funcional**				

Observaciones:

Tabla 10. Artefacto 23: Matriz de requisitos funcionales según el cliente Vs. requisitos funcionales del dominio.
Fuente: Autora de la investigación (2015).

InALPro	
ARTEFACTO	**DESCRIPCIÓN**
Nombre	Matriz de requisitos no-funcionales según el cliente Vs. requisitos no-funcionales del dominio.
Constructor	Analista de requisitos
Nro. Identificación ACTIVIDAD	A_14
Nro. Identificación ARTEFACTO	24

Formato Asociado:

Requisitos no-funcionales del dominio. / **Requisitos no-funcionales según el cliente.**		**Nro. Identifi-cación**			
		Descripción del requisito funcional			
Nro. Identificación	**Descripción del requisito funcional**				

Observaciones:

Tabla 11. Artefacto 24: Matriz de requisitos no-funcionales según el cliente Vs. requisitos no-funcionales del dominio.
Fuente: Autora de la investigación (2015).

InALPro	
ARTEFACTO	**DESCRIPCIÓN**
Nombre	Lista de requisitos funcionales de la aplicación no contemplados en el dominio con sus propiedades de calidad asociadas.
Constructor	Analista de requisitos
Nro. Identificación ACTIVIDAD	A_14
Nro. Identificación ARTEFACTO	25

Formato Asociado:

Nro. Identificación	Descripción del requisito funcional	Características de calidad ISO/IEC 25010	Atributos (Características)

Tabla 12. Artefacto 25: Lista de requisitos funcionales de la aplicación no contemplados en el dominio con sus propiedades de calidad asociadas.
Fuente: Autora de la investigación (2015).

InALPro	
ARTEFACTO	**DESCRIPCIÓN**
Nombre	Lista de requisitos no-funcionales de la aplicación no contemplados en el dominio con sus propiedades de calidad asociadas.
Constructor	Analista de requisitos
Nro. Identificación ACTIVIDAD	A_14
Nro. Identificación ARTEFACTO	26

Formato Asociado:

Nro. Identificación	Reglas del negocio asociadas a la aplicación.	Requisitos no-funcionales derivados de las reglas del negocio.	Propiedades de Calidad asociadas a los requisitos no-funcionales ISO/IEC 25010	Atributos (Características)
	Políticas			
	Procesamiento			
	Implementación			

Tabla 13. Artefacto 26: Lista de requisitos no-funcionales de la aplicación con sus propiedades de calidad asociadas.
Fuente: Autora de la investigación (2015).

InALPro	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	Modelo de calidad de la aplicación
Constructor:	Analista de Requisitos
Nro. Identificación ACTIVIDAD:	A_14
Nro. Identificación ARTEFACTO:	27
Formato Asociado: Modelo de calidad de la aplicación.	

Tabla 14. Artefacto 27: Modelo de calidad de la aplicación.
Fuente: Autora de la investigación (2015).

InALPro	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	Resumen sobre el Análisis de Requisitos de la Aplicación
Constructor:	Arquitecto de software
Nro. Identificación ACTIVIDAD:	A_14
Nro. Identificación ARTEFACTO:	28
Texto resumen sobre decisiones sobre el análisis de requisitos funcionales y no-funcionales de la aplicación.	

Tabla 15. Artefacto 28: Resumen sobre el Análisis de Requisitos de la Aplicación.
Fuente: Autora de la investigación (2015).

Actividad A_15: Validación de Requisitos de la Aplicación.

Es la actividad de evaluación y revisión de los requisitos para asegurar que estos definen la aplicación correctamente. Para esta validación, se examinan la Lista de requisitos funcionales con sus propiedades de calidad asociada (artefacto 6 de InDoCaS) y la Lista de requisitos no-funcionales con sus propiedades de calidad asociada (artefacto 7 de InDoCaS) ya actualizadas en la actividad Análisis de Requisitos de la Aplicación (A_14).

Seguidamente se construyen Casos de Uso de la aplicación (artefacto 29), para estos se toman en cuenta el Conjunto de Características (artefacto 3 de InDoCaS), el Conjunto de puntos de variación (artefacto 4 de InDoCaS), Listas de requisitos funcionales/no-funcionales con sus propiedades de calidad asociadas (artefacto 6 y 7 de InDoCaS).

De igual forma, se crea una Lista de chequeo de requisitos de la aplicación (artefacto 30) basada en Listas de requisitos funcionales/no-funcionales con sus propiedades de calidad asociadas (artefacto 6 y 7 de InDoCaS).

Con los Casos de Uso de la aplicación (artefacto 29) se analiza conjuntamente con el cliente/usuario el alcance de las funcionalidades del sistema y se indica en la Lista de chequeo de requisitos de la aplicación (artefacto 30) los requisitos verificados. En caso de encontrarse alguna funcionalidad no contemplada, se repetirá la actividad A_14: Análisis de Requisitos de la Aplicación.

En la Tabla 16 se resume esta actividad.

InALPro	
ACTIVIDAD	**DESCRIPCIÓN**
Nombre de la Actividad:	Validación de Requisitos de la aplicación.
Responsable:	Analista de Requisitos.
Objetivo:	Validar los requisitos y funcionalidades de la aplicación.
Nro. Identificación:	A_15
Tipo de documento generado:	Tabla, Documento
Técnica(s) utilizada(s):	Adaptación del subproceso de validación requisitos del Método Gray Watch.
Artefacto(s) de entrada:	Conjunto de características (artefacto 3 de InDoCaS). Conjunto de puntos de variación (artefacto 4 de InDoCaS) Lista de requisitos funcionales con sus propiedades de calidad asociada (artefacto 6 de InDoCaS) Lista de requisitos no-funcionales con sus propiedades de calidad asociada (artefacto 7 de InDoCaS)
Artefacto(s) de salida:	Casos de uso de la aplicación (artefacto 29). Lista de chequeo de requisitos de la aplicación (artefacto 30).

Tabla 16. Actividad A_15: Validación de Requisitos de la aplicación.
Fuente: Autora de la investigación (2015).

En esta actividad se generan dos artefactos nuevos, los Casos de uso de la aplicación (Tabla 17) y la Lista de chequeo de requisitos de la aplicación (Tabla 18). El resto de los artefactos simplemente se ajustan o validan si están acordes a los requisitos especificados.

InALPro	
ARTEFACTO	**DESCRIPCIÓN**
Nombre	Diagrama de Casos de uso de la aplicación
Constructor	Analista de requisitos
Nro. Identificación ACTIVIDAD	A_15
Nro. Identificación ARTEFACTO	29
Formato Asociado: Diagrama UML de Casos de Uso. **Observaciones:**	

Tabla 17. Artefacto 29: Diagrama de Casos de uso de la aplicación.
Fuente: Autora de la investigación (2015).

InALPro	
ARTEFACTO	**DESCRIPCIÓN**
Nombre	Lista de chequeo de requisitos de la aplicación.
Constructor	Analista de requisitos
Nro. Identificación ACTIVIDAD	A_15
Nro. Identificación ARTEFACTO	30

Formato Asociado:

Nro. Identificación	Descripción del requisito	Conforme (Si ó No)	Observaciones

Observaciones:

Tabla 18. Artefacto 30: Lista de Chequeo de requisitos de la aplicación.
Fuente: Autora de la investigación (2015).

Disciplina Diseño Arquitectónico de la Aplicación

Esta disciplina íntegra requisitos arquitecturales no contemplados en el dominio y válida la arquitectura del dominio represente una solución adecuada para la aplicación de acuerdo a su especificidad, en caso de diferencia de componentes, se agrega el faltante y la información se envía a InDoCaS (como artefacto opcional) para ampliar el conocimiento del dominio. En la Figura 32, el diagrama de actividades propuesto de la disciplina Diseño Arquitectónico de la Aplicación.

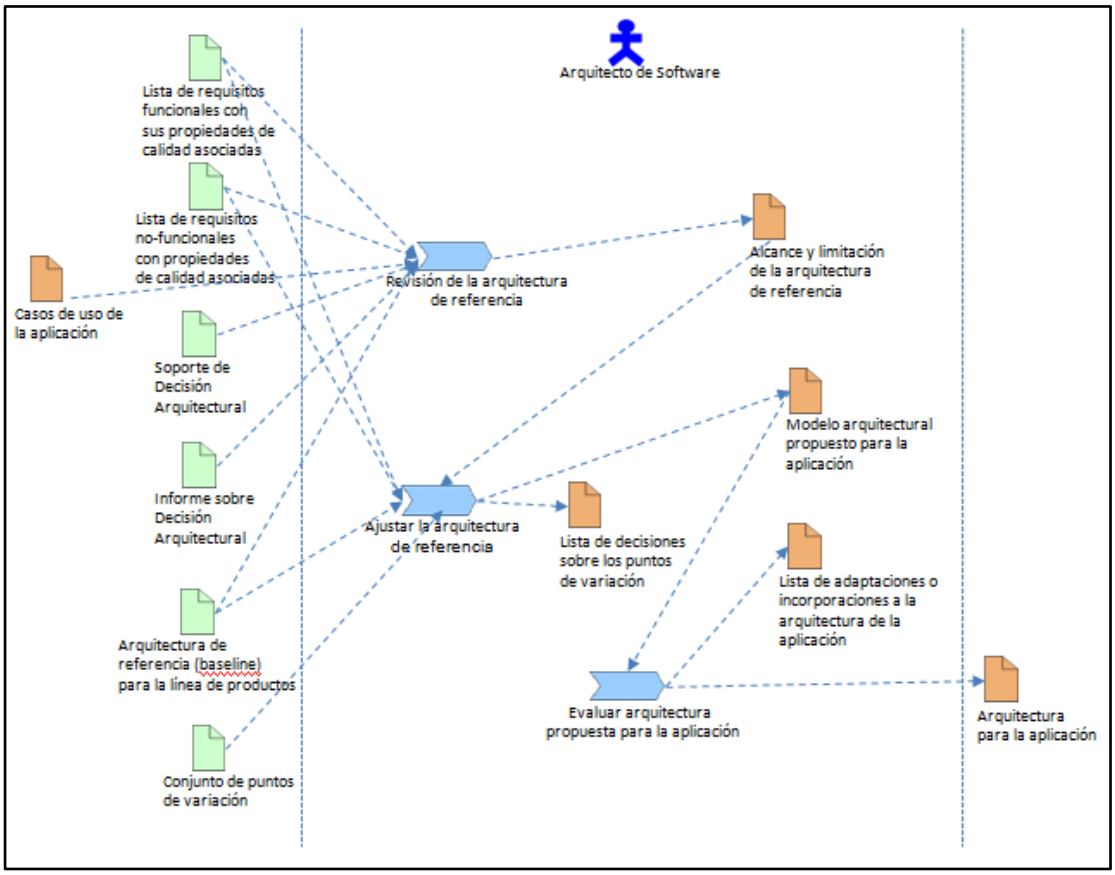


Figura 32. Disciplina Diseño Arquitectónico de la Aplicación.
Fuente: Autora de la investigación (2015).

A continuación las actividades de la Disciplina Diseño Arquitectónico de la Aplicación:

Actividad A_16: Revisión de la arquitectura de referencia.

En esta actividad se realiza una revisión de la Arquitectura de referencia (baseline) para la línea de producto (artefacto 19 de InDoCaS), como esta arquitectura es genérica, se busca definir su alcance y limitación para la aplicación especifica según las reglas del negocio.

La experticia del arquitecto de software influye sustancialmente en esta actividad, quien mediante su experiencia y usando la Lista de requisitos funcionales con sus propiedades de calidad asociadas (artefacto 6 de InDoCaS), la Lista de requisitos no-funcionales con sus propiedades de calidad asociadas (artefacto 7 de

InDoCaS), el Soporte de decisión arquitectural (artefacto 16 de InDoCaS), el Documento de razonamiento arquitectural (artefacto 18 de InDoCaS), la Arquitectura base para la línea de productos (artefacto 19 de InDoCaS) y el Informe sobre la decisión arquitectural (artefacto 20 de InDoCaS), identificará el alcance y las limitaciones de la arquitectura propuesta en InDoCaS, es decir, si la organización de componentes y conectores es suficiente para ser una solución aceptable para la aplicación, o por el contrario hay parte de la arquitectura que no es necesaria aún o si falta dar solución a algunos requisitos. Esta actividad se resume en la Tabla 19, y genera el formato Alcance y limitación de la arquitectura de referencia (artefacto 31), en la Tabla 20.

InALPro	
ACTIVIDAD	**DESCRIPCIÓN**
Nombre de la Actividad:	Revisión de la arquitectura de referencia
Responsable:	Arquitecto de software
Objetivo:	Revisar la arquitectura de referencia elaborada por InDoCaS para constatar cumple los requisitos del negocio.
Nro. Identificación:	A_16
Tipo de documento generado:	Tabla, Documento
Técnica(s) utilizada(s):	Adaptación del proceso Diseño Arquitectónico del Método Gray Watch.
Artefacto(s) de entrada:	Lista de requisitos funcionales con sus propiedades de calidad asociadas (artefacto 6 de InDoCaS). Lista de requisitos no-funcionales con sus propiedades de calidad asociadas (artefacto 7 de InDoCaS) Soporte de decisión arquitectural (artefacto 16 de InDoCaS) Documento de razonamiento arquitectural (artefacto 18 de InDoCaS) Arquitectura base para la línea de productos (artefacto 19 de InDoCaS) Informe sobre la decisión arquitectural (artefacto 20 de InDoCaS)
Artefacto(s) de salida:	Alcance y limitación de la arquitectura de referencia (artefacto 31).

Tabla 19. Actividad A_16: Revisión de la arquitectura de referencia.
Fuente: Autora de la investigación (2015).

InALPro	
ARTEFACTO	**DESCRIPCIÓN**
Nombre	Alcance y limitación de la arquitectura de referencia.
Constructor	Arquitecto de software.
Nro. Identificación ACTIVIDAD	A_16
Nro. Identificación ARTEFACTO	31

Formato Asociado:

Nro. Identificación	Aspecto alcanzado

Nro. Identificación	Limitación observada

Tabla 20. Artefacto 31: Alcance y limitación de la arquitectura de referencia.
Fuente: Autora de la investigación (2015).

Actividad A_17: Ajustar la arquitectura de referencia.

La actividad siguiente, personaliza un poco más las características de la aplicación. Para esto toma las Lista de requisitos funcionales con sus propiedades de calidad asociadas (artefacto 6 de InDoCaS), la Lista de requisitos no-funcionales con sus propiedades de calidad asociadas (artefacto 7 de InDoCaS), Alcance y limitación de la arquitectura de referencia (artefacto 31) y el Conjunto de puntos de variación (artefacto 5 de InDoCaS) para tomar las decisiones concretas donde la arquitectura de referencia ofrece alternativas a elegir. Estas decisiones se reflejarán en el formato Lista de decisiones sobre los puntos de variación (artefacto 32). De ser necesario, se realizarán los ajustes correspondientes a los Diagramas de Casos de Uso de la aplicación (artefacto 29).

Posteriormente, se realizan los ajustes pertinentes de la Lista de decisiones sobre los puntos de variación (artefacto 32) en la Arquitectura de referencia (baseline) para la línea de productos (artefacto 19 de InDoCaS) dando como resultado el Modelo

arquitectural propuesto para la aplicación (artefacto 33). La Tabla 21 resume esta actividad, en la Tabla 22 se presenta el artefacto 31: Lista de decisiones sobre los puntos de variación y en la Tabla 23: el Modelo arquitectural propuesto para la aplicación.

InALPro	
ACTIVIDAD	**DESCRIPCIÓN**
Nombre de la Actividad:	Seleccionar decisiones concretas en puntos de variación
Responsable:	Arquitecto de software
Objetivo:	Ajustar las decisiones concretas de los puntos de variación a la arquitectura base.
Nro. Identificación:	A_17
Tipo de documento generado:	Tabla, Documento
Técnica(s) utilizada(s):	Adaptación del proceso Diseño Arquitectónico del Método Gray Watch.
Artefacto(s) de entrada:	Conjunto de puntos de variación (artefacto 5 de InDoCaS) Lista de requisitos funcionales de la aplicación y propiedades de calidad asociadas (artefacto 6 de InDoCaS). Lista de requisitos no-funcionales de la aplicación y propiedades de calidad asociadas (artefacto 7 de InDoCaS). Arquitectura de referencia (baseline) para la línea de productos (artefacto 19 de InDoCaS) Alcance y limitación de la arquitectura de referencia (artefacto 31)
Artefacto(s) de salida:	Lista de decisiones sobre los puntos de variación (artefacto 32). Modelo arquitectural propuesto para la aplicación (artefacto 33).

Tabla 21. Actividad A_17: Seleccionar decisiones concretas en puntos de variación.
Fuente: Autora de la investigación (2015).

InALPro	
ARTEFACTO	**DESCRIPCIÓN**
Nombre	Lista de decisiones sobre los puntos de variación
Constructor	Arquitecto de software.
Nro. Identificación ACTIVIDAD	A_17
Nro. Identificación ARTEFACTO	32

Formato Asociado:

Nro. Identificación	Característica	Decisión seleccionada

Tabla 22. Artefacto 32: Lista de decisiones sobre los puntos de variación.
Fuente: Autora de la investigación (2015).

InALPro	
ARTEFACTO	**DESCRIPCIÓN**
Nombre	Arquitectura propuesta para la aplicación
Constructor	Arquitecto de software.
Nro. Identificación ACTIVIDAD	A_17
Nro. Identificación ARTEFACTO	33
Formato Asociado: Modelo arquitectural propuesto para la aplicación.	

Tabla 23. Artefacto 33: Arquitectura propuesta para la aplicación.
Fuente: Autora de la investigación (2015).

Actividad A_18: Evaluar la arquitectura propuesta para la aplicación.

A continuación, se revisa el Modelo Arquitectural propuesto para la aplicación (artefacto 33), esto se realiza a través de la selección de un método o técnica de evaluación de la arquitectura o mediante el análisis de fortalezas y debilidades de dicha arquitectura, esta evaluación puede arrojar como resultado la necesidad de incorporar o adaptar algún elemento arquitectural. Para esta revisión se pueden utilizar también los Soporte de decisión arquitectural (artefacto 16 de InDoCaS), Documento de razonamiento arquitectural (artefacto 18 de InDoCaS), Arquitectura base para la línea de productos (artefacto 19 de InDoCaS), Informe sobre la decisión arquitectural (artefacto 20 de InDoCaS). Mediante la Tabla 24 se resume esta actividad. Y las Tabla 25 y Tabla 26, representan la Lista de adaptaciones o incorporaciones a la arquitectura de la aplicación (artefacto 34) y la Arquitectura para la aplicación (artefacto 35), que son el resultado de esta revisión.

InALPro	
ACTIVIDAD	**DESCRIPCIÓN**
Nombre de la Actividad:	Evaluar arquitectura para la aplicación
Responsable:	Arquitecto de software
Objetivo:	Verificar y ajustar (si es necesario) la arquitectura propuesta para la aplicación.
Nro. Identificación:	A_18
Tipo de documento generado:	Tabla, Documento
Técnica(s) utilizada(s):	Adaptación del proceso Diseño Arquitectónico del Método Gray Watch.

Artefacto(s) de entrada:	Soporte de decisión arquitectural (artefacto 16 de InDoCaS) Documento de razonamiento arquitectural (artefacto 18 de InDoCaS) Arquitectura base para la línea de productos (artefacto 19 de InDoCaS) Informe sobre la decisión arquitectural (artefacto 20 de InDoCaS) Modelo arquitectural propuesto para la aplicación (artefacto 32)
Artefacto(s) de salida:	Lista de adaptaciones o incorporaciones a la arquitectura de la aplicación (artefacto 34). Arquitectura para la aplicación (artefacto 35).

Tabla 24. Actividad A_18: Evaluar arquitectura propuesta para la aplicación.
Fuente: Autora de la investigación (2015).

InALPro	
ARTEFACTO	**DESCRIPCIÓN**
Nombre	Lista de adaptaciones o incorporaciones a la arquitectura de la aplicación
Constructor	Arquitecto de software.
Nro. Identificación ACTIVIDAD	A_18
Nro. Identificación ARTEFACTO	34
Formato Asociado:	

Nro. Identificación	Modificación propuesta

Tabla 25. Artefacto 34: Lista de adaptaciones o incorporaciones a la arquitectura de la aplicación.
Fuente: Autora de la investigación (2015).

InALPro	
ARTEFACTO	**DESCRIPCIÓN**
Nombre	Arquitectura para la aplicación
Constructor	Arquitecto de software.
Nro. Identificación ACTIVIDAD	A_18
Nro. Identificación ARTEFACTO	35
Formato Asociado: Modelo arquitectural para la aplicación.	

Tabla 26. Artefacto 35: Arquitectura para la aplicación.
Fuente: Autora de la investigación (2015).

A continuación, se resumen las actividades propuestas para implementar una línea de producto de software integrando los procesos InDoCaS e InALPro, ver Tabla 27.

Proceso para Líneas de Producto de Software				
ENTRADA			Modelo conceptual del Dominio	
DISCIPLINA			**ACTIVIDADES**	
InDoCaS	Análisis del Dominio		**Nro.**	**Nombre de la actividad**
			A_01	Identificación de Requisitos
			A_02	Obtener modelo de similitudes y variabilidad
			A_03	Identificación de propiedades de calidad
			A_04	Obtener modelo de calidad asociado al dominio
			A_05	Creación de escenarios de calidad del dominio
			A_06	Identificar los estilos arquitecturales para el dominio
	Diseño del Dominio	Síntesis	A_07	Seleccionar los elementos del diseño del dominio que satisfagan el conjunto minimal de requisitos funcionales y no funcionales
			A_08	Escoger patrones arquitecturales candidatos
			A_09	Instanciar elementos arquitecturales para los elementos del diseño del dominio
			A_10	Identificar similitudes entre los elementos arquitecturales instanciados
			A_11	Decidir la selección de la arquitectura como solución arquitectural candidata
		Evaluación	A_12	Validar modelo de calidad del dominio con arquitectura candidatas
			A_13	Escoger arquitectura base para la familia
InALPro	**Ingeniería de Requisitos de la Aplicación**		**A_14**	**Análisis de Requisitos de la Aplicación**
			A_15	**Validación de Requisitos de la Aplicación**
	Diseño Arquitectónico de la Aplicación		**A_16**	**Revisión de la Arquitectura de Referencia**
			A_17	**Ajustar la arquitectura de Referencia**
			A_18	**Evaluar la arquitectura propuesta para la aplicación**

Tabla 27. Resumen de actividades para implementación de una línea de producto de software. Fuente: Autora de la investigación (2015).

Ejemplificación de la propuesta en el dominio eLearning para la UPTP "Juan de Jesús Montilla"

Esta ejemplificación funge como la fase de evaluación del paradigma de ciencia del diseño, como también para explicar un poco mejor la propuesta. En esta sección,

se aplica el proceso InALPro al dominio eLearning de la UPTP “Juan de Jesús Montilla”. El objetivo principal de InALPro es guiar los pasos de la ingeniería de la aplicación en LPS, pero una línea de producto de software tiene dos fases fundamentales, como ya se ha mencionado, la ingeniería de dominio y la ingeniería de la aplicación. Por lo tanto, se comienza ejecutando las disciplinas del proceso InDoCaS (ingeniería de dominio) y luego InALPro (ingeniería de la aplicación), de esta manera se ejemplifica como el uso de estos dos procesos permite proponer una línea de producto de software contemplando características de calidad en el proceso. El producto principal de la aplicación de InALPro es una arquitectura eLearning para la UPTP.

Para el Proyecto eLearning de la UPTP, se han levantado los siguientes requisitos de información:

- Administrar diferentes tipos de objetos de aprendizaje, audio, multimedia, texto, entre otros.
- Gestionar los perfiles de usuario.
- Gestionar los cursos, creación, configuración de sus características, la inscripción de los participantes.
- Gestionar la comunicación entre usuarios, por ejemplo a través foros, cuestionarios, chats, cartelera informativa y encuestas.
- Gestión de evaluaciones de los contenidos administrados.
- Proporcionar ayuda en línea a través de secciones de preguntas frecuentes, glosarios, enlaces a otros recursos.
- Registro y seguimiento de las actividades de docentes y estudiantes.
- Notificar por correo electrónico y mensajes de texto sobre eventos, asignaciones, evaluaciones y actividades en general.
- Compartir en redes sociales los logros alcanzados por los estudiantes.
- Interfaz de uso sencillo y amigable para usuarios con escasos conocimientos informáticos.

- Gestionar la comunicación de la plataforma eLearning con el sistema de control de estudios para obtener los datos de estudiantes y docentes.
- Garantizar la integridad de la información administrada por la plataforma.
- El acceso al sistema debe estar habilitado a través de la intranet institucional y de internet.

A continuación se aplica el proceso InDoCaS en la UPTP.

Aplicación de InDoCaS

Disciplina Análisis del Dominio

Actividad A_01: Identificación de requisitos

En esta actividad se identifican los requisitos funcionales y no funcionales del dominio, estos se reflejan en la Tabla 28 y Tabla 29 respectivamente.

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	Lista de requisitos funcionales del dominio
Constructor:	Analista de Requisitos
Nro. Identificación ACTIVIDAD:	A_01
Nro. Identificación ARTEFACTO:	1

Nro. Identificación	Requisito funcional
1	Administrar diferentes tipos de contenido, audio, multimedia, texto, entre otros.
2	Gestionar los de perfiles de usuario.
3	Gestionar los cursos, creación, configuración de sus características, la inscripción de los participantes.
4	Gestionar la comunicación entre usuarios, por ejemplo a través foros, cuestionarios, chats, cartelera informativa y encuestas.
5	Gestión de evaluaciones de los contenidos administrados.
6	Proporcionar ayuda en línea a través de secciones de preguntas frecuentes, glosarios, enlaces a otros recursos.
7	Registro y seguimiento de las actividades de docentes y estudiantes.
8	Notificar por correo electrónico y mensajes de texto sobre eventos, asignaciones, evaluaciones y actividades en general.
9	Compartir en redes sociales los logros alcanzados por los estudiantes.
10	Gestionar la comunicación de la plataforma eLearning con el sistema de control de estudios para obtener los datos de estudiantes y docentes.
11	Generar reportes de estudiantes por cursos, de actividades realizadas, de la interacción entre estudiantes y con los docentes, reportes estadísticos.

Tabla 28. Ejemplificación: Lista de requisitos funcionales del dominio.
Fuente: Autora de la investigación (2015).

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	Lista de requisitos no funcionales del dominio
Constructor:	Analista de Requisitos
Nro. Identificación ACTIVIDAD:	A_01
Nro. Identificación ARTEFACTO:	2

Nro. Identificación	Reglas del Negocio asociadas al dominio	Requisitos no funcionales derivados de las reglas del negocio
	Políticas	
1	Uso de protocolos de comunicación que permitan la disponibilidad de los servicios a través de la internet e intranet institucional.	-Cumplimiento de normativas y estándares institucionales para garantizar la disponibilidad del servicio. -Funcionamiento sobre plataforma de software libre.
	Procesamiento	
2	Los usuarios seleccionan que información desean compartir.	-El usuario es el único autorizado para compartir su información. -El usuario podrá compartir información entre miembros de su grupo, comunidades o redes sociales.
3	Disponibilidad de acceso a los datos en todo momento.	-Cumplimiento de normas y estándares que permitan el acceso a los datos centralizados. -Garantizar la conexión al sistema al iniciar sesión.
4	Garantizar la integridad de la información.	-Al completar una transacción el sistema garantiza que los datos se almacenaron correctamente en la base de datos.
	Implementación	
5	Interfaz de usuario sencilla y amigable	-La interacción entre el usuario y el sistema debe ser fluida, fácil de usar. -Se debe implementar ayudas en línea que respondan a las dudas del uso de la aplicación.
6	Control de acceso a los recursos.	-Cada usuario debe tener acceso solo a los servicios según su perfil. -Los docentes y estudiantes solo pueden tener acceso a cursos a los cuales pertenecen. -Las unidades de contenido van a tener atributos público, privado o personalizado.
7	Mecanismos de soporte de fallos.	-En situaciones inesperadas de fallos, el sistema debe disponer de rutinas de recuperación y respaldo de los datos.

Tabla 29. Ejemplificación: Lista de requisitos no funcionales del dominio.
Fuente: Autora de la investigación (2015).

Actividad A_02: Obtener modelo de similitudes y variabilidad

En esta actividad se define el conjunto de requisitos mínimos obligatorios en el sistema eLearning para la UPTP, para este fin se utiliza FODA[13]. Se obtienen tres artefactos, el conjunto de características (Tabla 30), conjunto de puntos de variación

[13] Feature-Oriented Domain Analisys.

(Tabla 31) y el conjunto minimal de requisitos funcionales y no funcionales (Tabla 32).

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	Conjunto de características (features)
Constructor:	Analista de Requisitos
Nro. Identificación ACTIVIDAD:	A_02
Nro. Identificación ARTEFACTO:	3

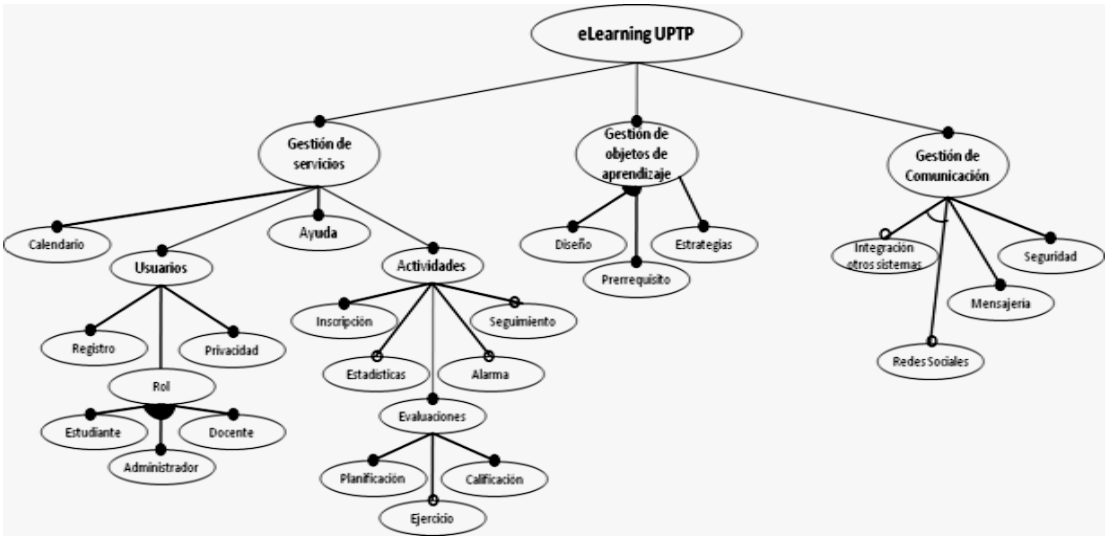


Tabla 30. Ejemplificación: Conjunto de características.
Fuente: Autora de la investigación (2015).

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	Conjunto de puntos de variación
Constructor:	Analista de Requisitos
Nro. Identificación ACTIVIDAD:	A_02
Nro. Identificación ARTEFACTO:	4

Característica	Punto de Variación
Publicación de información en las redes sociales.	Esta característica va a depender del sitio de conexión, ya que en la UPTP, no hay conexión a algunas redes sociales dentro de la institución, esto es debido a normas institucionales.
Comunicación con el sistema de control de estudios.	Por medidas de seguridad, la sincronización de la información con el sistema de control de estudios se realizará en periodos específicos, y en ese periodo no estará el sistema disponible para los usuarios en general, solo el administrador de la plataforma eLearning.

Tabla 31. Ejemplificación: Conjunto de puntos de variación.
Fuente: Autora de la investigación (2015).

A continuación, los requisitos funcionales y no funcionales mínimos para diseñar la arquitectura del dominio.

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	Conjunto minimal de requisitos funcionales y no funcionales
Constructor:	Analista de Requisitos
Nro. Identificación ACTIVIDAD:	A_02
Nro. Identificación ARTEFACTO:	5

Nro. Identificación	Descripción del requisito funcional
1	Administrar objetos de aprendizaje en diferentes presentaciones.
2	Gestionar roles de usuarios.
3	Gestionar las actividades académicas y de intercambio de información entre participantes.
4	Gestionar la comunicación entre usuarios, por ejemplo a través foros, cuestionarios, chats, cartelera informativa y encuestas.
5	Gestión de evaluaciones de los objetivos de aprendizaje.
6	Gestionar ayuda en línea mediante diferentes recursos.
7	Registro y seguimiento de las actividades de docentes y estudiantes.
8	Notificar por correo electrónico sobre eventos, asignaciones, evaluaciones y actividades en general.
9	Generar reporte de actividad en la plataforma y reportes estadísticos.

Nro. Identificación	Descripción del requisito no funcional
1	Cumplimiento de normativas y estándares institucionales para garantizar la disponibilidad del servicio.
2	Funcionamiento sobre plataforma de software libre.
3	El usuario es el único autorizado para compartir su información.
4	Cumplimiento de normas y estándares que permitan el acceso a los datos centralizados.
5	Garantizar la conexión al sistema al iniciar sesión.
6	Al completar una transacción el sistema garantiza que los datos se almacenaron correctamente en la base de datos.
7	La interacción entre el usuario y el sistema debe ser fluida, fácil de usar.
8	Se debe implementar ayudas en línea que respondan a las dudas del uso de la aplicación.
9	Cada usuario debe tener acceso solo a los servicios y cursos según su rol.
10	En situaciones inesperadas de fallos, el sistema debe disponer de rutinas de recuperación y respaldo de los datos.

Tabla 32. Ejemplificación: Conjunto minimal de requisitos funcionales y no funcionales. Fuente: Autora de la investigación (2015).

Actividad A_03: Identificación de propiedades de Calidad.

En esta actividad se identifican las propiedades de calidad asociadas a los requisitos funcionales y no funcionales de acuerdo al estándar ISO/IEC 25010, estas se muestran en la Tabla 33 y Tabla 34.

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	Lista de requisitos funcionales con sus propiedades de calidad asociada
Constructor:	Analista de Requisitos
Nro. Identificación ACTIVIDAD:	A_03
Nro. Identificación ARTEFACTO:	6

Nro. Identificación	Descripción del requisito funcional	Características de calidad ISO/IEC 25010	Atributos (Características)
1	Administrar objetos de aprendizaje en diferentes presentaciones.	**Adecuación Funcional** -Completitud **Usabilidad** -Capacidad de Aprendizaje **Fiabilidad** - Disponibilidad	-Funcionalidades disponibles al usuario medido mediante encuesta. -Porcentaje de comprensión medido por encuesta. -Atributo: tiempo de espera (segundos). Métrica: valor rango [1..20]
2	Gestionar roles de usuarios.	**Adecuación Funcional** - Completitud **Fiabilidad** - Disponibilidad **Seguridad** -Integridad -Autenticidad	-Funcionalidades disponibles al usuario medido mediante encuesta. -Atributo: tiempo de espera (segundos). Métrica: valor rango [1..20] -Protección del origen de datos y del proceso de autenticación de acceso. Métrica: porcentaje [0..1] -Presencia de mecanismo. Métrica: Booleano
3	Gestionar las actividades académicas y de intercambio de información entre participantes.	**Adecuación Funcional** - Completitud **Eficiencia** -Utilización de recursos **Fiabilidad** - Disponibilidad	Funcionalidades disponibles al usuario medido mediante encuesta. -Atributo: uso de recursos en las tareas. Métrica: porcentaje [0..1]. -Atributo: tiempo de espera (segundos). Métrica: valor rango [1..20]
4	Gestionar la comunicación entre usuarios, por ejemplo a través foros, cuestionarios, chats, cartelera informativa y encuestas.	**Eficiencia** -Utilización de recursos **Usabilidad** -Capacidad de Aprendizaje **Fiabilidad** -Disponibilidad	-Atributo: uso de recursos en las tareas. Métrica: porcentaje [0..1]. -Porcentaje de comprensión medido por encuesta. -Atributo: tiempo de espera (segundos). Métrica: valor rango [1..20]
5	Gestión de evaluaciones de los objetivos de aprendizaje.	**Adecuación Funcional** - Completitud **Usabilidad** -Protección contra errores de usuario **Fiabilidad** -Disponibilidad **Seguridad** -Integridad -Autenticidad	-Funcionalidades disponibles al usuario medido mediante encuesta. -Presencia de mecanismo de validación. -Atributo: tiempo de espera (segundos). Métrica: valor rango [1..20] -Protección del origen de datos y del proceso de autenticación de acceso. Métrica: porcentaje [0..1] -Presencia de mecanismo de autenticación. Métrica: Booleano

6	Gestionar ayuda en línea mediante diferentes recursos.	**Usabilidad** -Estética **Fiabilidad** -Disponibilidad	-Porcentaje de aceptación medido por encuesta. -Atributo: tiempo de espera (segundos). Métrica: valor rango [1..20]
7	Registro y seguimiento de las actividades de docentes y estudiantes.	**Adecuación Funcional** -Completitud **Usabilidad** -Capacidad de Aprendizaje **Fiabilidad** -Disponibilidad	Funcionalidades disponibles al usuario medido mediante encuesta. -Porcentaje de comprensión medido por encuesta. -Atributo: tiempo de espera (segundos). Métrica: valor rango [1..20]
8	Notificar por correo electrónico sobre eventos, asignaciones, evaluaciones y actividades en general.	**Fiabilidad** -Disponibilidad	-Atributo: tiempo de espera (segundos). Métrica: valor rango [1..20]
9	Generar reporte de actividad en la plataforma y reportes estadísticos.	**Eficiencia** -Utilización de recursos **Usabilidad** -Estética **Fiabilidad** -Disponibilidad	-Atributo: uso de recursos en las tareas. Métrica: porcentaje [0..1]. -Porcentaje de aceptación medido por encuesta. -Atributo: tiempo de espera (segundos). Métrica: valor rango [1..20]

Tabla 33. Ejemplificación: Lista de requisitos funcionales con sus propiedades de calidad asociada.
Fuente: Autora de la investigación (2015).

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	Lista de requisitos no funcionales con sus propiedades de calidad asociada
Constructor:	Analista de Requisitos
Nro. Identificación ACTIVIDAD:	A_03
Nro. Identificación ARTEFACTO:	7

Nro. Identificación	Reglas del negocio asociadas al dominio	Requisitos no funcionales derivados de las reglas del negocio	Características de calidad ISO/IEC 25010	Atributos (Características)
	Políticas			
1	Uso de protocolos de comunicación que permitan la disponibilidad de los servicios a través de la internet e intranet institucional.	-Cumplimiento de normativas y estándares institucionales para garantizar la disponibilidad del servicio. -Funcionamiento sobre plataforma de software libre.	**Fiabilidad** -Disponibilidad **Portabilidad** -Adaptabilidad	-Atributo: tiempo de espera (segundos). Métrica: valor rango [1..20] -Atributo: presencia de mecanismo. Métrica: valor booleano.
	Procesamiento			
2	Los usuarios seleccionan que información desean compartir.	-El usuario es el único autorizado para compartir su información.	**Seguridad** -Confidencialidad	-Atributo: presencia de mecanismo. Métrica: valor en el rango [1..5]
3	Disponibilidad de acceso a los datos en todo momento.	-Cumplimiento de normas y estándares que permitan el acceso a los datos centralizados. -Garantizar la conexión al sistema al iniciar sesión.	**Fiabilidad** -Madurez -Disponibilidad	-Atributo: presencia de mecanismo. Métrica: valor booleano. -Atributo: tiempo de espera (segundos). Métrica: valor rango [1..20]
4	Garantizar la integridad de la información.	-Al completar una transacción el sistema garantiza que los datos se almacenaron correctamente en la base de datos.	**Fiabilidad** -Madurez **Adecuación Funcional** -Completitud	-Atributo: presencia de mecanismo. Métrica: valor booleano. -Atributo: proporción de transacciones completas. Métrica: valor rango [0..1]
	Implementación			
5	Interfaz de usuario sencilla y amigable	-La interacción entre el usuario y el sistema debe ser fluida, fácil de usar. -Se debe implementar ayudas en línea que respondan a las dudas del uso de la aplicación.	**Usabilidad** -Operabilidad -Estética	-Atributo: proporción de funcionalidades que son entendidas por el usuario medido en encuesta. Métrica: valor [0..1]
6	Control de acceso a los recursos.	-Cada usuario debe tener acceso solo a los servicios y cursos según su rol.	**Seguridad** -Autenticidad	-Presencia de mecanismo de autenticación. Métrica: Booleano.
7	Mecanismos de soporte de fallos.	-En situaciones inesperadas de fallos, el sistema debe disponer de rutinas de recuperación y respaldo de los datos.	**Fiabilidad** -Tolerancia a fallos	-Presencia de mecanismo en caso de fallos. Métrica: valor booleano.

Tabla 34. Ejemplificación: Lista de requisitos no funcionales con sus propiedades de calidad asociada
Fuente: Autora de la investigación (2015).

Actividad 4: Obtener modelo de calidad asociado al dominio.

Este modelo representa las características de calidad mínimas de un producto dentro del dominio estudiado, ver Tabla 35.

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	Modelo de calidad del dominio
Constructor:	Analista de Requisitos
Nro. Identificación ACTIVIDAD:	A_04
Nro. Identificación ARTEFACTO:	8

Tabla 35. Ejemplificación: Modelo de calidad del dominio
Fuente: Autora de la investigación (2015).

Actividad 5: Creación de escenarios de calidad del dominio.

Los escenarios de calidad se usan para garantizar que se han evaluado los requisitos deseados de calidad para el dominio, estos se reflejan en la Tabla 36.

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	Escenarios de calidad
Constructor:	Arquitecto de software y analista de requisitos
Nro. Identificación ACTIVIDAD:	A_05
Nro. Identificación ARTEFACTO:	9

Escenario de Calidad	Reglas de negocio asociadas al dominio	Requisitos no funcionales derivados de las reglas del negocio	Propiedades de calidad asociadas a requisitos no funcionales (ISO/IEC 25010)
	Políticas		
1	Uso de protocolos de comunicación que permitan la disponibilidad de los servicios a través de la internet e intranet institucional.	-Cumplimiento de normativas y estándares institucionales para garantizar la disponibilidad del servicio. -Funcionamiento sobre plataforma de software libre.	**Fiabilidad** -Disponibilidad **Portabilidad** -Adaptabilidad
	Procesamiento		
2	Los usuarios seleccionan que información desean compartir.	-El usuario es el único autorizado para compartir su información.	**Seguridad** -Confidencialidad
3	Disponibilidad de acceso a los datos en todo momento.	-Cumplimiento de normas y estándares que permitan el acceso a los datos centralizados. -Garantizar la conexión al sistema al iniciar sesión.	**Fiabilidad** -Madurez -Disponibilidad
4	Garantizar la integridad de la información.	-Al completar una transacción el sistema garantiza que los datos se almacenaron correctamente en la base de datos.	**Fiabilidad** -Madurez **Adecuación Funcional** -Completitud
	Implementación		
5	Interfaz de usuario sencilla y amigable	-La interacción entre el usuario y el sistema debe ser fluida, fácil de usar. -Se debe implementar ayudas en línea que respondan a las dudas del uso de la aplicación.	**Usabilidad** -Operabilidad -Estética
6	Control de acceso a los recursos.	-Cada usuario debe tener acceso solo a los servicios y cursos según su rol.	**Seguridad** -Autenticidad
7	Mecanismos de soporte de fallos.	-En situaciones inesperadas de fallos, el sistema debe disponer de rutinas de recuperación y respaldo de los datos.	**Fiabilidad** -Tolerancia a fallos

Tabla 36. Ejemplificación: Escenarios de calidad
Fuente: Autora de la investigación (2015).

Actividad 6: Identificar los estilos arquitecturales para el dominio.

Es importante en esta actividad la experiencia del arquitecto de software en el estudio de estilos arquitecturales, que serán seleccionados atendiendo al conjunto minimal de requisitos funcionales/no-funcionales ajustados al modelo de calidad del dominio. Los estilos arquitecturales que garantizan el cumplimiento del modelo de

calidad asociado al dominio son arquitectura cliente-servidor y arquitectura modelo-vista-controlador.

El modelo cliente-servidor es una arquitectura distribuida que según Sommerville, I. (2005) modela una aplicación como un conjunto de servicios proporcionados por los "servidores" y un conjunto de "clientes" que usan estos servicios. Los "clientes" necesitan conocer que "servidores" están disponibles, pero normalmente no conocen la existencia de otros clientes. Este modelo se subdivide en cliente-servidor dos capas y tres capas, para este caso se utiliza la de tres capas, donde la aplicación está constituida por procesos separados en presentación, procesamiento de la aplicación y administración de los datos. Esta arquitectura proporciona mayor facilidad en la gestión de los recursos, el acceso y la integridad de la información, adicionalmente, permite la escalabilidad de la aplicación según nuevas necesidades.

El modelo vista controlador, es un estilo arquitectural que separa los datos de una aplicación, la interfaz de usuario y la lógica del negocio en tres componentes separados. Este modelo arquitectural potencia las ideas de reutilización y la separación de conceptos, lo que facilita el desarrollo y mantenimiento de las aplicaciones. Adicionalmente, es muy utilizado en el desarrollo de aplicaciones web. Los estilos seleccionados se indican en la Tabla 37.

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	Estilos Arquitecturales
Constructor:	Arquitecto de software y Analista de Requisitos
Nro. Identificación ACTIVIDAD:	A_06
Nro. Identificación ARTEFACTO:	10
Arquitectura cliente-servidor (Tres Capas) Modelo-Vista-Controlador(MVC)	

Tabla 37. Ejemplificación: Estilos Arquitecturales
Fuente: Autora de la investigación (2015).

Disciplina Diseño del Dominio

Actividad 7: Seleccionar elementos del diseño conceptual que satisfagan el conjunto minimal de requisitos.

En esta sección se establecen los elementos de diseño conceptual con sus prioridades, estos elementos se identifican en los requisitos. Ver Tabla 38.

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	Elementos de diseño conceptual
Constructor:	Analista de Requisitos
Nro. Identificación ACTIVIDAD:	A_07
Nro. Identificación ARTEFACTO:	11

Nro. Identificación	**Descripción del requisito funcional**	**Importancia**	**Dificultad**
1	Administrar objetos de aprendizaje en diferentes presentaciones.	Alta	Media
2	Gestionar roles de usuarios.	Alta	Baja
3	Gestionar las actividades académicas y de intercambio de información entre participantes.	Media	Media
4	Gestionar la comunicación entre usuarios, por ejemplo a través foros, cuestionarios, chats, cartelera informativa y encuestas.	Alta	Baja
5	Gestión de evaluaciones de los objetivos de aprendizaje.	Alta	Media
6	Gestionar ayuda en línea mediante diferentes recursos.	Alta	Baja
7	Registro y seguimiento de las actividades de docentes y estudiantes.	Media	Media
8	Notificar por correo electrónico sobre eventos, asignaciones, evaluaciones y actividades en general.	Media	Baja
9	Generar reporte de actividad en la plataforma y reportes estadísticos.	Alta	Baja
10	Cumplimiento de normativas y estándares institucionales para garantizar la disponibilidad del servicio.	Alta	Alta
11	Funcionamiento sobre plataforma de software libre.	Alta	Media
12	El usuario es el único autorizado para compartir su información.	Alta	Media
13	Cumplimiento de normas y estándares que permitan el acceso a los datos centralizados.	Alta	Alta
14	Garantizar la conexión al sistema al iniciar sesión.	Alta	Media
15	Al completar una transacción el sistema garantiza que los datos se almacenaron correctamente en la base de datos.	Alta	Media
16	La interacción entre el usuario y el sistema debe ser fluida, fácil de usar.	Alta	Alta
17	Se debe implementar ayudas en línea que respondan a las dudas del uso de la aplicación.	Alta	Baja
18	Cada usuario debe tener acceso solo a los servicios y cursos según su rol.	Alta	Media
19	En situaciones inesperadas de fallos, el sistema debe disponer de rutinas de recuperación y respaldo de los datos.	Alta	Alta

Tabla 38. Ejemplificación: Elementos de diseño conceptual.
Fuente: Autora de la investigación (2015).

Actividad 8: Escoger patrones arquitecturales candidatos.

En esta actividad se seleccionan y ordenan jerárquicamente los patrones arquitecturales candidatos que satisfacen los atributos de calidad. Ver Tabla 39.

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	Patrones Arquitecturales Candidatos
Constructor:	Arquitecto de software
Nro. Identificación ACTIVIDAD:	A_08
Nro. Identificación ARTEFACTO:	12

Nro. Patrón	Nombre	Descripción	Favorece	Componentes Conectores	Ejemplo
1	Proxy	Es un intermediario responsable del control de acceso a recursos.	Seguridad		Proxy Web
2	Adapter	Encapsula funciones y datos proporcionados por API no orientadas a objeto, en clases con interfaces más robustas y mantenibles.	Portabilidad Fiabilidad	Componentes: Aplicación, API Conectores: Mecanismo de comunicación de las clases	Librerías Java, Php.
3	Network Comunication Protocol	Sistema de reglas para gestionar la comunicación entre dos o más entidades.	Seguridad Fiabilidad		
4	Multi-database	Organiza bases de datos distribuidas sobre un modelo cliente/servidor, donde agentes o mediadores, aceptan queries de usuario, los reconducen a las bases de datos disponibles y devuelven respuesta adecuada.	Portabilidad	Componentes: Repositorio de datos centrales Conectores: Mecanismo y protocolo de comunicación entre componente	Implementación de Corba, ODBC, JDBC, Sql-Net
5	Local Proxy	Instancia local que provee una interface a un servicio local o remoto	Seguridad		Proxy Web
6	Facade	Proporciona una interfaz simple para un subsistema complejo	Adecuación Funcional	Componentes: Facade, Subclasses	Clases Java
7	Data Transfer Object	Es un objeto para transportar datos entre procesos	Eficiencia Adecuación Funcional	Componentes: Paquete de datos Conectores: Mecanismos y protocolos de comunicación entre componentes	Objetos simples (POJO) en java
8	Server Session State	Guarda el estado de la sesión en un sistema con control de acceso a los recursos	Seguridad	Componentes: Context, State, Subclase ConcreteState	
9	Chain of Responsibi-lity	Disminuye el acoplamiento de la petición de un emisor hacia su receptor dando la posibilidad a otro objeto de responder la petición	Fiabilidad Eficiencia	Componentes: Client, Handler, ConcreteHandler	Sistema de ayuda en el contexto de sistemas web

Tabla 39. Ejemplificación: Patrones Arquitecturales Candidatos
Fuente: Autora de la investigación (2015).

Actividad 9: Instanciar elementos arquitecturales para los elementos del diseño del dominio.

Ahora se seleccionan patrones arquitecturales desde el artefacto patrones arquitecturales candidatos, ver Tabla 40.

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	Elementos arquitecturales (componentes y conectores)
Constructor:	Arquitecto de software
Nro. Identificación ACTIVIDAD:	A_09
Nro. Identificación ARTEFACTO:	13

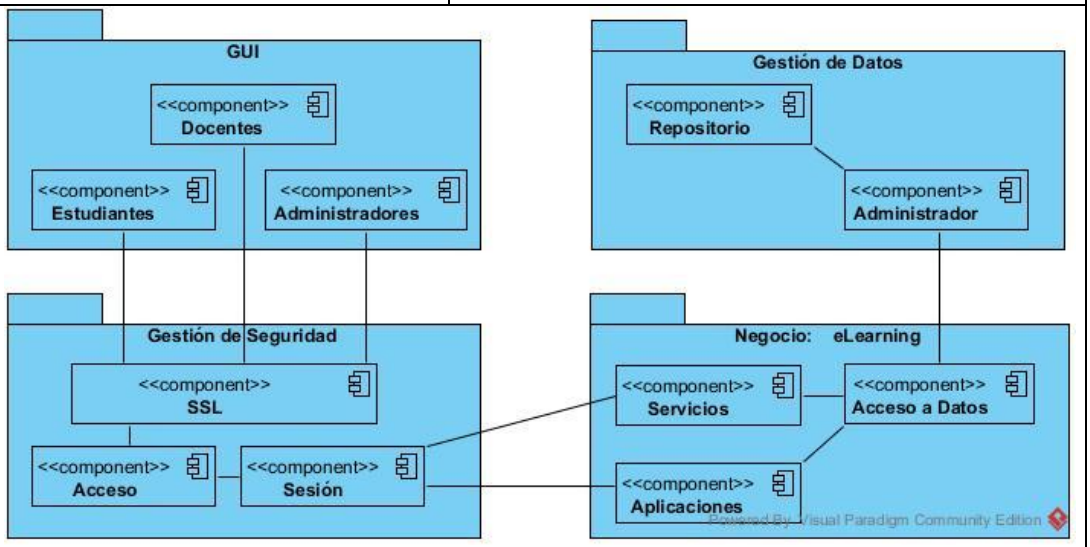
Nro. Ident.	Descripción del requisito	Nombre del patrón escogido	Nro. Patrón esc
1	Administrar objetos de aprendizaje en diferentes presentaciones.	Adapter, Multi-database, Data Transfer Object	2, 4, 7
2	Gestionar roles de usuarios.	Server Session State, Data Transfer Object, Multi-database	8, 7, 4
3	Gestionar las actividades académicas y de intercambio de información entre participantes.	Proxy, Data Transfer Object, Multi-database, Facade	1, 7, 4, 6
4	Gestionar la comunicación entre usuarios, por ejemplo a través foros, cuestionarios, chats, cartelera informativa y encuestas.	Data Transfer Object, Multi-database, Facade	7, 4, 6
5	Gestión de evaluaciones de los objetivos de aprendizaje.	Data Transfer Object, Multi-database	7, 4
6	Gestionar ayuda en línea mediante diferentes recursos.	Chain of Responsibility, Multi-database, Facade	9, 4, 6
7	Registro y seguimiento de las actividades de docentes y estudiantes.	Data Transfer Object, Multi-database	7, 4
8	Notificar por correo electrónico sobre eventos, asignaciones, evaluaciones y actividades en general.	Network Comunication Protocol	3
9	Generar reporte de actividad en la plataforma y reportes estadísticos.	Multi-database, Facade	4, 6
10	Cumplimiento de normativas y estándares institucionales para garantizar la disponibilidad del servicio.	Network Comunication Protocol, Local Proxy	3, 5
11	Funcionamiento sobre plataforma de software libre.	Network Comunication Protocol, Local Proxy	3, 5
12	El usuario es el único autorizado para compartir su información.	Multi-database	4
13	Cumplimiento de normas y estándares que permitan el acceso a los datos centralizados.	Data Transfer Object, Multi-database, Local Proxy	7, 4, 5
14	Garantizar la conexión al sistema al iniciar sesión.	Server Session State, Network Comunication Protocol, Multidatabase	8, 3, 4
15	Al completar una transacción el sistema garantiza que los datos se almacenaron correctamente en la base de datos.	Data Transfer Object, Multi-database	7, 4
16	La interacción entre el usuario y el sistema debe ser fluida, fácil de usar.	Data Transfer Object, Facade	7, 6
17	Se debe implementar ayudas en línea que respondan a las dudas del uso de la aplicación.	Chain of Responsibility, Multi-database	9, 4
18	Cada usuario debe tener acceso solo a los servicios y cursos según su rol.	Multi-database, Facade	4, 6
19	En situaciones inesperadas de fallos, el sistema debe disponer de rutinas de recuperación y respaldo de los datos.	Multi-database, Data Transfer Object	4, 7

Tabla 40. Ejemplificación: Elementos arquitecturales (componentes y conectores)
Fuente: Autora de la investigación (2015).

Actividad 10: Identificar similitudes entre elementos arquitecturales instanciados.

Esta actividad agrupa elementos arquitecturales para generar los componentes principales de la arquitectura, ver Tabla 41.

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	Elementos arquitecturales similares
Constructor:	Arquitecto de software
Nro. Identificación ACTIVIDAD:	A_10
Nro. Identificación ARTEFACTO:	14



GUI: Representa la capa de interacción con los usuarios de la plataforma, esta va a estar conformada por tres componentes según sus roles dentro del sistema.

Gestión de Seguridad: Es el responsable de gestionar el cifrado y descifrado de datos (SSL), de autorizar el acceso mediante la autenticación de usuarios, y la verificación de la sesión.

Negocio eLearning: Esta capa está compuesta por tres elementos, el componente de servicios que dispone las utilidades de la plataforma como ayuda, notificaciones, chats, y otros. El componente de aplicaciones tiene las funciones propias de la aplicación, como asignaciones, evaluaciones, cursos, objetos de aprendizaje y otros. Adicionalmente se encuentra un componente de acceso a datos que dispone de la lógica de negocio para acceder a la información.

Gestión de Datos: Conformado por el administrador de la base de datos responsable de los cambios en la BD y del uso de procedimientos almacenados y triggers. El Repositorio es la base de datos relacional donde se almacenarán los datos de la plataforma.

Tabla 41. Ejemplificación: Elementos arquitecturales similares
Fuente: Autora de la investigación (2015).

Actividad 11: Decidir la selección de la arquitectura como solución arquitectural candidata.

Esta actividad selecciona las posibles soluciones arquitecturales candidatas que satisfacen el modelo de calidad del dominio, ver Tabla 42.

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	Conjunto de soluciones candidatas
Constructor:	Arquitecto de software
Nro. Identificación ACTIVIDAD:	A_11
Nro. Identificación ARTEFACTO:	15

Nro. Identificación	Descripción del requisito funcional	Nombre del patrón escogido	Nro. del patrón escogido
1	Administrar objetos de aprendizaje en diferentes presentaciones.	Adapter, Multi-database, Data Transfer Object	2, 4, 7
2	Gestionar roles de usuarios.	Server Session State, Data Transfer Object, Multi-database	8, 7, 4
3	Gestionar las actividades académicas y de intercambio de información entre participantes.	Proxy, Data Transfer Object, Multi-database, Facade	1, 7, 4, 6
4	Gestionar la comunicación entre usuarios, por ejemplo a través foros, cuestionarios, chats, cartelera informativa y encuestas.	Data Transfer Object, Multi-database, Facade	7, 4, 6
5	Gestión de evaluaciones de los objetivos de aprendizaje.	Data Transfer Object, Multi-database	7, 4
6	Gestionar ayuda en línea mediante diferentes recursos.	Chain of Responsibility, Multi-database, Facade	9, 4, 6
7	Registro y seguimiento de las actividades de docentes y estudiantes.	Data Transfer Object, Multi-database	7, 4
8	Notificar por correo electrónico sobre eventos, asignaciones, evaluaciones y actividades en general.	Network Comunication Protocol	3
9	Generar reporte de actividad en la plataforma y reportes estadísticos.	Multi-database, Facade	4, 6
10	Cumplimiento de normativas y estándares institucionales para garantizar la disponibilidad del servicio.	Network Comunication Protocol, Local Proxy	3, 5
11	Funcionamiento sobre plataforma de software libre.	Network Comunication Protocol, Local Proxy	3, 5
12	El usuario es el único autorizado para compartir su información.	Server Session State	8
13	El usuario podrá compartir información entre miembros de su grupo o comunidades.	Data Transfer Object, Facade	7, 6
14	Cumplimiento de normas y estándares que permitan el acceso a los datos centralizados.	Data Transfer Object, Multi-database, Local Proxy	7, 4, 5
15	Garantizar la conexión al sistema al iniciar sesión.	Server Session State, Network Comunication Protocol, Multi-database	8, 3, 4
16	Al completar una transacción el sistema garantiza que los datos se almacenaron correctamente en la base de datos.	Data Transfer Object, Multi-database	7, 4
17	La interacción entre el usuario y el sistema debe ser fluida, fácil de usar.	Facade	7, 6

18	Se debe implementar ayudas en línea que respondan a las dudas del uso de la aplicación.	Chain of Responsibility, Multi-database	9, 4
19	Cada usuario debe tener acceso solo a los servicios según su rol.	Server Session State, Facade	8, 6
20	Los docentes y estudiantes solo pueden tener acceso a cursos a los cuales pertenecen.	Multi-database, Server Session State	4, 8
21	En situaciones inesperadas de fallos, el sistema debe disponer de rutinas de recuperación y respaldo de los datos.	Multi-database, Server Session State	4, 8

Tabla 42. Ejemplificación: Conjunto de soluciones candidatas
Fuente: Autora de la investigación (2015).

A continuación el artefacto Soporte de Decisión Arquitectural, este contiene comentarios y consideraciones de la escogencia de los elementos arquitecturales, y posteriormente puede ser útil en el proceso de evaluación arquitectural, ver Tabla 43.

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	Soporte de decisión arquitectural
Constructor:	Arquitecto de software
Nro. Identificación ACTIVIDAD:	A_11
Nro. Identificación ARTEFACTO:	16

La estructura de la plataforma se separó en tres capas, la capa de presentación (GUI), la capa de la lógica del negocio (eLearnign) y la capa de datos, esto con la finalidad de controlar el acceso a los recursos, garantizar la integridad de los datos, y que la plataforma sea escalable en el tiempo. De igual manera se considera que esta estructura facilita el mantenimiento del sistema, afectando en menor grado a los usuarios.

La capa GUI consta de tres componentes de acuerdo a los roles admitidos por la plataforma, esto permite que cada interfaz evolucione por separado, esto es importante porque se pretende que la interfaz de los estudiantes sea accesible, amigable e interactiva, lo que se logrará observando la reacción de ellos ante cada recurso implementado. Se debe tomar en cuenta que hay un componente para los administradores, pero en realidad van a existir diferentes administradores o administradores con diferentes niveles de acceso.

Los componentes de seguridad fueron separados para definir mejor la función de cada uno, y proporcionarle mayor fortaleza a la plataforma, evitando accesos no autorizados. En este se identifican los usuarios, y se les permite disponer de los recursos adecuados a su rol.

La capa del negocio se dividió en tres componentes, el componente de servicios de la plataforma que son actividades complementarias al proceso de aprendizaje, como lo son chats, sistema de notificaciones, agenda, la ayuda, entre otros. También se tiene el componente de aplicaciones, que dispone de todas las funcionalidades directas del proceso de enseñanza-aprendizaje, allí se incluyen los cursos, los objetos de aprendizaje, actividades evaluativas y otros. De igual manera se dispone de un componente de acceso a la data, esto se incluye para tener una mejor definición y separación de los componentes que manejan la lógica del negocio.

Finalmente, la Gestión de Datos, donde se encuentra el administrador del repositorio, que gestionará el acceso a la base de datos, y el repositorio en sí, que representa a la o las bases de datos que almacenaran la información de la plataforma.

Tabla 43. Ejemplificación: Soporte de decisión arquitectural
Fuente: Autora de la investigación (2015).

Actividad 12: Validar modelo de calidad del dominio con arquitecturas candidatas.

Esta actividad revisa el cumplimiento de las características de calidad del modelo de calidad del dominio por parte de las arquitecturas candidatas, las que cumplen el modelo de calidad, se incorporan a la arquitectura validada, ver tabla Tabla 44.

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	Arquitectura validada a la familia producto
Constructor:	Arquitecto de software
Nro. Identificación ACTIVIDAD:	A_12
Nro. Identificación ARTEFACTO:	17

Nro. Identificación	**Requisitos no funcionales derivados de las reglas del negocio**	**Propiedades de Calidad asociadas a los requisitos no funcionales (ISO/IEC 25010)**	**Nombre del patrón escogido**
1	-Cumplimiento de normativas y estándares institucionales para garantizar la disponibilidad del servicio.	**Fiabilidad** -Disponibilidad **Portabilidad** -Adaptabilidad	Network Comunication Protocol, Local Proxy
2	-El usuario es el único autorizado para compartir su información.	**Seguridad** -Confidencialidad	Server Session State
3	-Cumplimiento de normas y estándares que permitan el acceso a los datos centralizados.	**Fiabilidad** -Madurez -Disponibilidad	Data Transfer Object, Multi-database, Local Proxy
4	-Garantizar la conexión al sistema al iniciar sesión.	**Fiabilidad** -Disponibilidad	Server Session State, Network Comunication Protocol, Multi-database
5	-Al completar una transacción el sistema garantiza que los datos se almacenaron correctamente en la base de datos.	**Fiabilidad** -Madurez **Adecuación Funcional** -Completitud	Data Transfer Object, Multi-database
6	-La interacción entre el usuario y el sistema debe ser fluida, fácil de usar.	**Usabilidad** -Operabilidad	Facade
7	-Se debe implementar ayudas en línea que respondan a las dudas del uso de la aplicación.	**Usabilidad** -Operabilidad -Estética	Chain of Responsibility, Multi-database
8	-Cada usuario debe tener acceso solo a los servicios según su rol, y actividades a las cuales está registrado.	**Seguridad** -Autenticidad	Server Session State, Facade
9	-En situaciones inesperadas de fallos, el sistema debe disponer de rutinas de recuperación y respaldo de los datos.	**Seguridad** -Autenticidad	Multi-database, Server Session State

Tabla 44. Ejemplificación: Arquitectura validada a la familia producto
Fuente: Autora de la investigación (2015).

Consecutivamente se elabora el Documento de razonamiento arquitectural (Tabla 45) para sustentar las decisiones tomadas.

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	Documento de razonamiento arquitectural
Constructor:	Arquitecto de software
Nro. Identificación ACTIVIDAD:	A_12
Nro. Identificación ARTEFACTO:	18

La arquitectura cliente servidor tres capas, permite separar la presentación (clientes), los procesos y los datos en componentes distintos. Esta separación es recomendable en casos en que los sistemas tienen muchos usuarios que van a realizar constantes accesos a la plataforma. La ventaja de esta arquitectura es su flexibilidad, la cual permite una más fácil escalabilidad del sistema, también proporciona mayor seguridad porque en cada capa se define el tipo de controles apropiado y mejor rendimiento debido a que las funciones se puede dividir en servidores diferentes. Adicionalmente, es más fácil ampliar a n-capas según los nuevos requisitos de la plataforma.

Por otro lado, la arquitectura MVC, fundamentada en la separación de conceptos y la reutilización de código, busca simplificar el desarrollo y mantenimiento de las aplicaciones; es muy utilizada en sistemas que contemplan el almacenamiento de grandes volúmenes de información.

Estos dos modelos arquitecturales tienen ventajas y desventajas, por tanto, la solución optima parece tomar lo bueno de cada una. Es decir, la separación de conceptos de la arquitectura MVC y la seguridad y escalabilidad de la arquitectura cliente-servidor tres capas.

Por lo anterior se plantea implementar la arquitectura cliente-servidor tres capas, donde, se utilizará un servidor para la GUI (capa de presentación web), un servidor para la aplicación (capa de negocios) y un servidor para el repositorio (datos). De igual forma, se plantea el uso de un firewall entre el servidor de aplicación y de datos, y otro entre el servidor web y el servidor de aplicación.

Los otros elementos arquitecturales seleccionados son de distinto nivel de granularidad, pero el objetivo principal es definir lo mejor posible la arquitectura genérica de la línea de producto, el enfoque arquitectural permite facilitar el desarrollo de un producto de forma expedita, los cambios en un elemento no afectan significativamente las funciones de los demás, de igual forma, se implementan técnicas de reutilización en los distintos niveles del sistema.

Fortalezas:

La implementación de firewall brinda mayor seguridad y robustez al sistema.

Esta arquitectura es escalable, lo que permitiría en el futuro balancear la carga de algún servidor, implementando otro, por ejemplo un servidor de datos solo para los objetos de aprendizaje y otro para el resto de los datos del sistema.

Debilidades:

Esta arquitectura pone mayor carga en la red, debido al mayor tráfico. Esto incidiría en la ralentización del servicio.

Al haber problema en un servidor intermedio, se puede interrumpir el servicio de la plataforma.

Nro. Identificación	Requisitos no funcionales derivados de las reglas del negocio	Propiedades de Calidad asociadas a los requisitos no funcionales (ISO/IEC 25010)	Nombre del patrón escogido	Comentarios Decisiones consideradas, rechazadas
1	-Cumplimiento de normativas y estándares institucionales para garantizar la disponibilidad del servicio.	**Fiabilidad** -Disponibilidad **Portabilidad** -Adaptabilidad	Network Comunication Protocol, Local Proxy	Manejo apropiado de la red, lo que optimiza el servicio a los usuarios. El proxy local es un mecanismo de seguridad adicional dentro de la intranet.

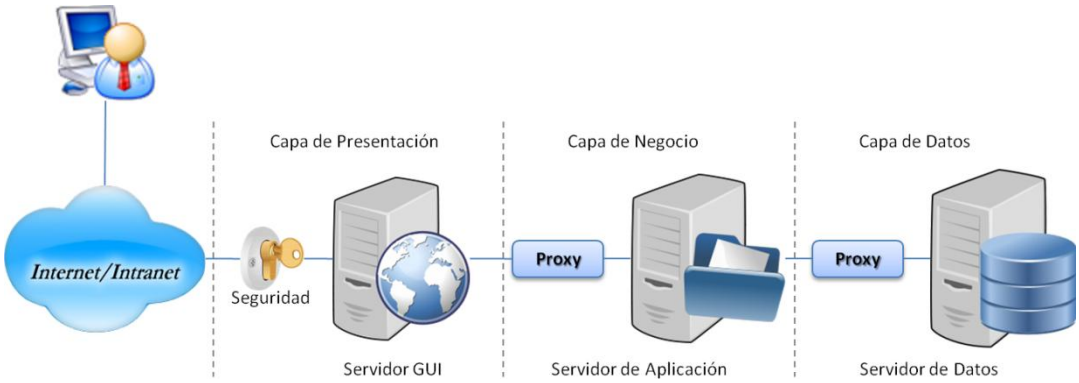
2	-El usuario es el único autorizado para compartir su información.	**Seguridad** -Confidencialidad	Server Session State	Al usuario iniciar sesión se garantiza que tiene acceso solo a sus datos y que cualquier cambio en la confidencialidad de su información será decisión propia.
3	-Cumplimiento de normas y estándares que permitan el acceso a los datos centralizados.	**Fiabilidad** -Madurez -Disponibilidad	Data Transfer Object, Multi-database, Local Proxy	Se pretende garantizar el acceso a los datos de forma segura y eficiente.
4	-Garantizar la conexión al sistema al iniciar sesión.	**Fiabilidad** -Disponibilidad	Server Session State, Network Comunication Protocol, Multi-database	El sistema debe estar siempre disponible, disminución de ocasiones donde es sistema está offline por mantenimiento.
5	-Al completar una transacción el sistema garantiza que los datos se almacenaron correctamente en la base de datos.	**Fiabilidad** -Madurez **Adecuación Funcional** -Completitud	Data Transfer Object, Multi-database	Se busca garantizar que las transacciones se realicen completa y correctamente.
6	-La interacción entre el usuario y el sistema debe ser fluida, fácil de usar.	**Usabilidad** -Operabilidad	Facade	La implementación de fachadas permite brindar interfaces sencillas al usuario, separando la complejidad del sistema, y ayudando al posterior mantenimiento o actualización.
7	-Se debe implementar ayudas en línea que respondan a las dudas del uso de la aplicación.	**Usabilidad** -Operabilidad -Estética	Chain of Responsibility, Multi-database	El sistema de ayuda debe estar siempre disponible, al no encontrarse un tópico, se mostrará el aspecto relacionado más cercano.
8	-Cada usuario debe tener acceso solo a los servicios según su rol, y actividades a las cuales está registrado.	**Seguridad** -Autenticidad	Server Session State, Facade	Cada rol debe tener un tipo de recursos disponibles, para que el sistema sea apropiado la presentación debe ser adaptable.
9	-En situaciones inesperadas de fallos, el sistema debe disponer de rutinas de recuperación y respaldo de los datos.	**Seguridad** -Autenticidad	Multi-database, Server Session State	Ante cierres inesperados la sesión caduca, las transacciones completadas se almacenaran, se notificará al usuario de su ultimo inicio de sesión y si hubo problemas en la misma.

Tabla 45. Ejemplificación: Documento de razonamiento arquitectural
Fuente: Autora de la investigación (2015).

Actividad 13: Escoger la arquitectura base para la familia.

En esta actividad se define la arquitectura del dominio, que es una arquitectura genérica para plataformas eLearning. Ver la Tabla 46.

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	Arquitectura base para la línea de producto
Constructor:	Arquitecto de software
Nro. Identificación ACTIVIDAD:	A_13
Nro. Identificación ARTEFACTO:	19



Arquitectura base para la línea de productos:
La arquitectura está conformada por tres capas, la capa de presentación, la capa de negocio (eLearning) y la capa de datos.
La capa de presentación está compuesta por dos elementos:
-Un componente de seguridad (SSL, autenticación y autorización de acceso, y sesión): este elemento es responsable de autenticar los usuarios y permitirle el acceso al sistema.
-Un componente de interfaz gráfica de usuario (GUI): es el medio que interactúa con el usuario final, es responsable de mostrar y capturar la información generada por el usuario.
La capa de negocio:
Está constituida por el servidor de aplicaciones, donde residen dos tipos de servicios, las aplicaciones propias del sistema (cursos, objetos de contenido, evaluaciones, y otros) y las utilidades (chats, notificaciones, agenda, y otros).
También contiene un elemento proxy para establecer una única vía de comunicación entre los servidores de presentación y de negocios, esto con la finalidad de dar mayor seguridad a la plataforma.
La capa de datos:
Está conformada por el servidor de datos, donde reside la base de datos y el componente administrador de la base de datos.

Tabla 46. Ejemplificación: Arquitectura base para la línea de producto
Fuente: Autora de la investigación (2015).

Posterior a la definición de la arquitectura del dominio, se elabora el Informe sobre la decisión arquitectural, donde se explican observaciones, ventajas, desventajas de la estructura de la arquitectura de dominio definida. Ver Tabla 47.

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	Informe sobre la decisión arquitectural
Constructor:	Arquitecto de software
Nro. Identificación ACTIVIDAD:	A_13
Nro. Identificación ARTEFACTO:	20
Ventajas de la propuesta arquitectural. El sistema es escalable, es decir, puede sostener su crecimiento en el tiempo sin perder la calidad de sus servicios. Esta estructura es más segura, la separación de los componentes arquitecturales en servidores separados y la utilización de proxy de comunicación disminuye la probabilidad de accesos no autorizados. Con esta arquitectura es un poco más sencillo realizar mantenimiento a algún elemento del sistema, sin interrumpir el funcionamiento de los otros componentes, de igual manera se pueden atender muchos clientes (usuarios de la plataforma) diariamente, proporcionándole recursos y servicios en línea.	

Tabla 47. Ejemplificación: Informe sobre la decisión arquitectural
Fuente: Autora de la investigación (2015).

Aplicación de InALPro

Disciplina Ingeniería de Requisitos de la Aplicación

Esta disciplina contrasta los requisitos funcionales y no funcionales del dominio con los requisitos funcionales y no funcionales según el cliente, para garantizar que se están cumpliendo todos los aspectos esperados por este. Esta disciplina contempla las siguientes actividades:

Actividad A_14: Análisis de Requisitos de la Aplicación.

Se identifican los requisitos funcionales y no funcionales de la aplicación (requisitos según el cliente), ver Tabla 48 y Tabla 49.

InALPro	
ARTEFACTO	**DESCRIPCIÓN**
Nombre	Lista de requisitos funcionales de la aplicación.
Constructor	Analista de requisitos
Nro. Identificación ACTIVIDAD	A_14
Nro. Identificación ARTEFACTO	21

Nro. Identificación	Descripción del requisito funcional
1	Administrar diferentes tipos de contenido, audio, multimedia, texto, entre otros.
2	Gestionar los de perfiles de usuario.
3	Gestionar los cursos, creación, configuración de sus características, la inscripción de los participantes.
4	Gestionar la comunicación entre usuarios, por ejemplo a través foros, cuestionarios, chats, cartelera informativa y encuestas.
5	Gestión de evaluaciones de los contenidos administrados.
6	Proporcionar ayuda en línea a través de secciones de preguntas frecuentes, glosarios, enlaces a otros recursos.
7	Registro y seguimiento de las actividades de docentes y estudiantes.
8	Notificar por correo electrónico y mensajes de texto sobre eventos, asignaciones, evaluaciones y actividades en general.
9	Compartir en redes sociales los logros alcanzados por los estudiantes.
10	Gestionar la comunicación de la plataforma eLearning con el sistema de control de estudios para obtener los datos de estudiantes y docentes.
11	Generar reportes de estudiantes por cursos, de actividades realizadas, de la interacción entre estudiantes y con los docentes, reportes estadísticos.

Tabla 48. Ejemplificación: Lista de requisitos funcionales de la aplicación
Fuente: Autora de la investigación (2015).

InALPro	
ARTEFACTO	**DESCRIPCIÓN**
Nombre	Lista de requisitos no-funcionales de la aplicación.
Constructor	Analista de requisitos
Nro. Identificación ACTIVIDAD	A_14
Nro. Identificación ARTEFACTO	22

Nro. Identificación	Reglas del negocio asociadas a la aplicación	Descripción del requisito funcional
	Políticas	
1	Uso de protocolos de comunicación que permitan la disponibilidad de los servicios a través de la internet e intranet institucional.	-Cumplimiento de normativas y estándares institucionales para garantizar la disponibilidad del servicio. -Funcionamiento sobre plataforma de software libre.
	Procesamiento	
2	Los usuarios seleccionan que información desean compartir.	-El usuario es el único autorizado para compartir su información. -El usuario podrá compartir información entre miembros de su grupo, comunidades o redes sociales.
3	Disponibilidad de acceso a los datos en todo momento.	-Cumplimiento de normas y estándares que permitan el acceso a los datos centralizados. -Garantizar la conexión al sistema al iniciar sesión.
4	Garantizar la integridad de la información.	-Al completar una transacción el sistema garantiza que los datos se almacenaron correctamente en la base de datos.
	Implementación	

5	Interfaz de usuario sencilla y amigable	-La interacción entre el usuario y el sistema debe ser fluida, fácil de usar. -Se debe implementar ayudas en línea que respondan a las dudas del uso de la aplicación.
6	Control de acceso a los recursos.	-Cada usuario debe tener acceso solo a los servicios y cursos según su perfil.
7	Mecanismos de soporte de fallos.	-En situaciones inesperadas de fallos, el sistema debe disponer de rutinas de recuperación y respaldo de los datos.

Tabla 49. Ejemplificación: Lista de requisitos no-funcionales de la aplicación
Fuente: Autora de la investigación (2015).

A continuación se comparan los requisitos funcionales/no-funcionales de la aplicación con los funcionales/no-funcionales del dominio, con la finalidad de verificar aspectos no contemplados, ver Tabla 50 y Tabla 51.

InALPro	
ARTEFACTO	**DESCRIPCIÓN**
Nombre	Matriz de requisitos funcionales de la aplicación Vs. requisitos funcionales del dominio.
Constructor	Analista de requisitos
Nro. Identificación ACTIVIDAD	A_14
Nro. Identificación ARTEFACTO	23

Requisitos funcionales del dominio. / Requisitos funcionales de la aplicación.		Nro. Identificación	1	2	3	4	5	6	7	8	9
Nro. Identificación	Descripción del requisito funcional	Descripción del requisito funcional	Administrar objetos de aprendizaje en diferentes presentaciones.	Gestionar roles de usuarios.	Gestionar las actividades académicas y de intercambio de información entre participantes.	Gestionar la comunicación entre usuarios, por ejemplo a través foros, cuestionarios, chats, cartelera informativa y encuestas.	Gestión de evaluaciones de los objetivos de aprendizaje.	Gestionar ayuda en línea mediante diferentes recursos.	Registro y seguimiento de las actividades de docentes y estudiantes.	Notificar por correo electrónico sobre eventos, asignaciones, evaluaciones y actividades en general.	Generar reporte de actividad en la plataforma y reportes estadísticos.
1	Administrar diferentes tipos de contenido, audio, multimedia, texto, entre otros.		X								
2	Gestionar los de perfiles de usuario.			X							
3	Gestionar los cursos, creación, configuración de sus características, la inscripción de los participantes.				X						
4	Gestionar la comunicación entre usuarios, por ejemplo a través foros, cuestionarios, chats, cartelera informativa y encuestas.					X					
5	Gestión de evaluaciones de los contenidos administrados.						X				
6	Proporcionar ayuda en línea a través de secciones de preguntas frecuentes, glosarios, enlaces a otros recursos.							X			
7	Registro y seguimiento de las actividades de docentes y estudiantes.								X		
8	Notificar por correo electrónico y mensajes de texto sobre eventos, asignaciones, evaluaciones y actividades en general.									X	
9	Compartir en redes sociales los logros alcanzados por los estudiantes.										
10	Gestionar la comunicación de la plataforma eLearning con el sistema de control de estudios para obtener los datos de estudiantes y docentes.										
11	Generar reportes de estudiantes por cursos, de actividades realizadas, de la interacción entre estudiantes y con los docentes, reportes estadísticos.										X

Observaciones:
Los requisitos 9 y 10 de la aplicación no han sido contemplados en los requisitos del dominio.

Tabla 50. Ejemplificación: Matriz de requisitos funcionales de la aplicación Vs. requisitos funcionales del dominio
Fuente: Autora de la investigación (2015).

InALPro	
ARTEFACTO	**DESCRIPCIÓN**
Nombre	Matriz de requisitos no-funcionales de la aplicación Vs. requisitos no-funcionales del dominio.
Constructor	Analista de requisitos
Nro. Identificación ACTIVIDAD	A_14
Nro. Identificación ARTEFACTO	24

Requisitos no-funcionales del dominio. / **Requisitos no-funcionales de la aplicación.**		**Nro. Identificación**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Nro. Identificación	**Descripción del requisito no funcional**	**Descripción del requisito no funcional**	Cumplimiento de normativas y estándares institucionales para garantizar la disponibilidad del servicio.	Funcionamiento sobre plataforma de software libre.	El usuario es el único autorizado para compartir su información.	Cumplimiento de normas y estándares que permitan el acceso a los datos centralizados.	Garantizar la conexión al sistema al iniciar sesión.	Al completar una transacción el sistema garantiza que los datos se almacenaron correctamente en la base de datos.	La interacción entre el usuario y el sistema debe ser fluida, fácil de usar.	Se debe implementar ayudas en línea que respondan a las dudas del uso de la aplicación.	Cada usuario debe tener acceso solo a los servicios y cursos según su perfil.	En situaciones inesperadas de fallos, el sistema debe disponer de rutinas de recuperación y respaldo de los datos.
1	Cumplimiento de normativas y estándares institucionales para garantizar la disponibilidad del servicio.		X									
2	Funcionamiento sobre plataforma de software libre.			X								
3	El usuario es el único autorizado para compartir su información.				X							
4	Cumplimiento de normas y estándares que permitan el acceso a los datos centralizados.					X						
5	Garantizar la conexión al sistema al iniciar sesión.						X					
6	Al completar una transacción el sistema garantiza que los datos se almacenaron correctamente en la base de datos.							X				
7	La interacción entre el usuario y el sistema debe ser fluida, fácil de usar.								X			
8	Se debe implementar ayudas en línea que respondan a las dudas del uso de la aplicación.									X		
9	Cada usuario debe tener acceso solo a los servicios según su rol.										X	
10	Los docentes y estudiantes solo pueden tener acceso a cursos a los cuales pertenecen.										X	
11	En situaciones inesperadas de fallos, el sistema debe disponer de rutinas de recuperación y respaldo de los datos.											X

Observaciones: Los requisitos no funcionales del dominio cubren los requisitos no funcionales de la aplicación.

Tabla 51. Ejemplificación: Matriz de requisitos no-funcionales de la aplicación Vs. requisitos no-funcionales del dominio
Fuente: Autora de la investigación (2015).

La comparación anterior da como resultado los requisitos no contemplados en el dominio, ver Tabla 52.

InALPro	
ARTEFACTO	**DESCRIPCIÓN**
Nombre	Lista de requisitos funcionales de la aplicación no contemplados en el dominio con sus propiedades de calidad asociadas.
Constructor	Analista de requisitos
Nro. Identificación ACTIVIDAD	A_14
Nro. Identificación ARTEFACTO	25

Nro. Identificación	Descripción del requisito funcional	Características de calidad ISO/IEC 25010	Atributos (Características)
1	Compartir en redes sociales los logros alcanzados por los estudiantes.	**Compatibilidad**: -Interoperabilidad	-Atributo: presencia de mecanismo. Métrica: valor booleano.
2	Gestionar la comunicación de la plataforma eLearning con el sistema de control de estudios para obtener los datos de estudiantes y docentes.	**Compatibilidad**: -Interoperabilidad **Seguridad**: -Confidencialidad	-Atributo: presencia de mecanismo. Métrica: valor booleano.

Tabla 52. Ejemplificación: Lista de requisitos funcionales de la aplicación no contemplados en el dominio con sus propiedades de calidad asociadas
Fuente: Autora de la investigación (2015).

A continuación, se ajusta el Modelo de calidad del dominio (artefacto 8 de InDoCaS), obteniendo el Modelo de calidad de la aplicación (Tabla 53).

InALPro	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	Modelo de calidad de la aplicación
Constructor:	Analista de Requisitos
Nro. Identificación ACTIVIDAD:	A_14
Nro. Identificación ARTEFACTO:	27

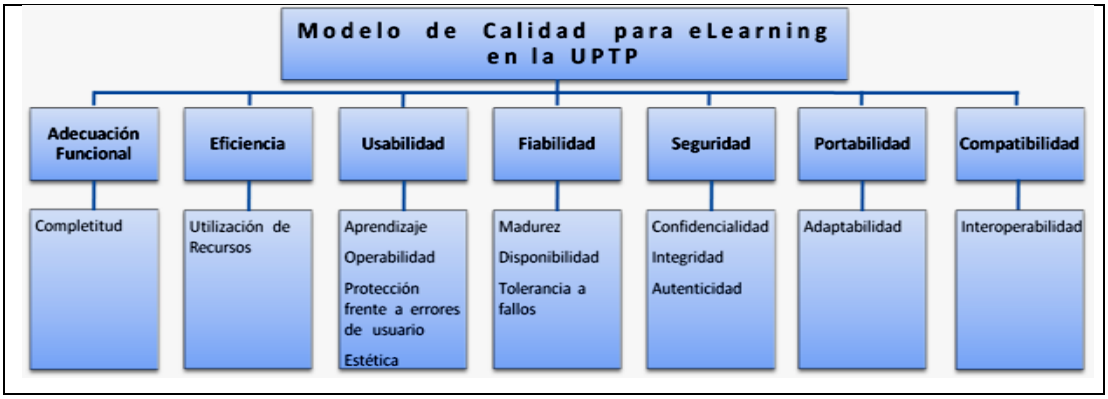


Tabla 53. Ejemplificación: Modelo de calidad de la aplicación
Fuente: Autora de la investigación (2015).

Finalmente, se elabora el Resumen sobre el Análisis de Requisitos de la Aplicación, para sustentar y/o documentar las decisiones tomadas en esta actividad, ver Tabla 54.

InALPro	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	Resumen sobre el Análisis de Requisitos de la Aplicación
Constructor:	Arquitecto de software
Nro. Identificación ACTIVIDAD:	A_14
Nro. Identificación ARTEFACTO:	28
-Los requisitos funcionales de la aplicación no están totalmente contemplados dentro del conjunto de requisitos funcionales del dominio, esto se evidenció mediante la Matriz de requisitos funcionales de la aplicación Vs. requisitos funcionales del dominio (artefacto 23). -Los requisitos no funcionales de la aplicación están totalmente contemplados dentro del conjunto de requisitos no funcionales del dominio, esto se evidenció mediante la Matriz de requisitos no funcionales de la aplicación Vs. requisitos funcionales del dominio (artefacto 24). -Los requisitos funcionales no contemplados en el dominio se presentan en el artefacto 25: Lista de requisitos funcionales de la aplicación no contemplados en el dominio con sus propiedades de calidad asociadas. -La Lista de requisitos funcionales de la aplicación no contemplados en el dominio con sus propiedades de calidad asociadas es pasada a InDoCaS para incluir dichos requisitos en el dominio como opcionales. -Dado que no surgieron nuevos requisitos no funcionales, no fue necesario elaborar el artefacto 26: Lista de requisitos no funcionales de la aplicación no contemplados en el dominio con sus propiedades de calidad asociadas	

Tabla 54. Ejemplificación: Resumen sobre el Análisis de Requisitos de la Aplicación
Fuente: Autora de la investigación (2015).

Actividad A_15: Validación de Requisitos de la Aplicación.

Para validad los requisitos de la aplicación se construyen los diagramas de casos de uso de la aplicación, de manera tal que se pueda analizar con el cliente/usuario los posibles escenarios de uso de la aplicación, ver Tabla 55.

InALPro	
ARTEFACTO	**DESCRIPCIÓN**
Nombre	Diagrama de Casos de uso de la aplicación
Constructor	Analista de requisitos
Nro. Identificación ACTIVIDAD	A_15
Nro. Identificación ARTEFACTO	29

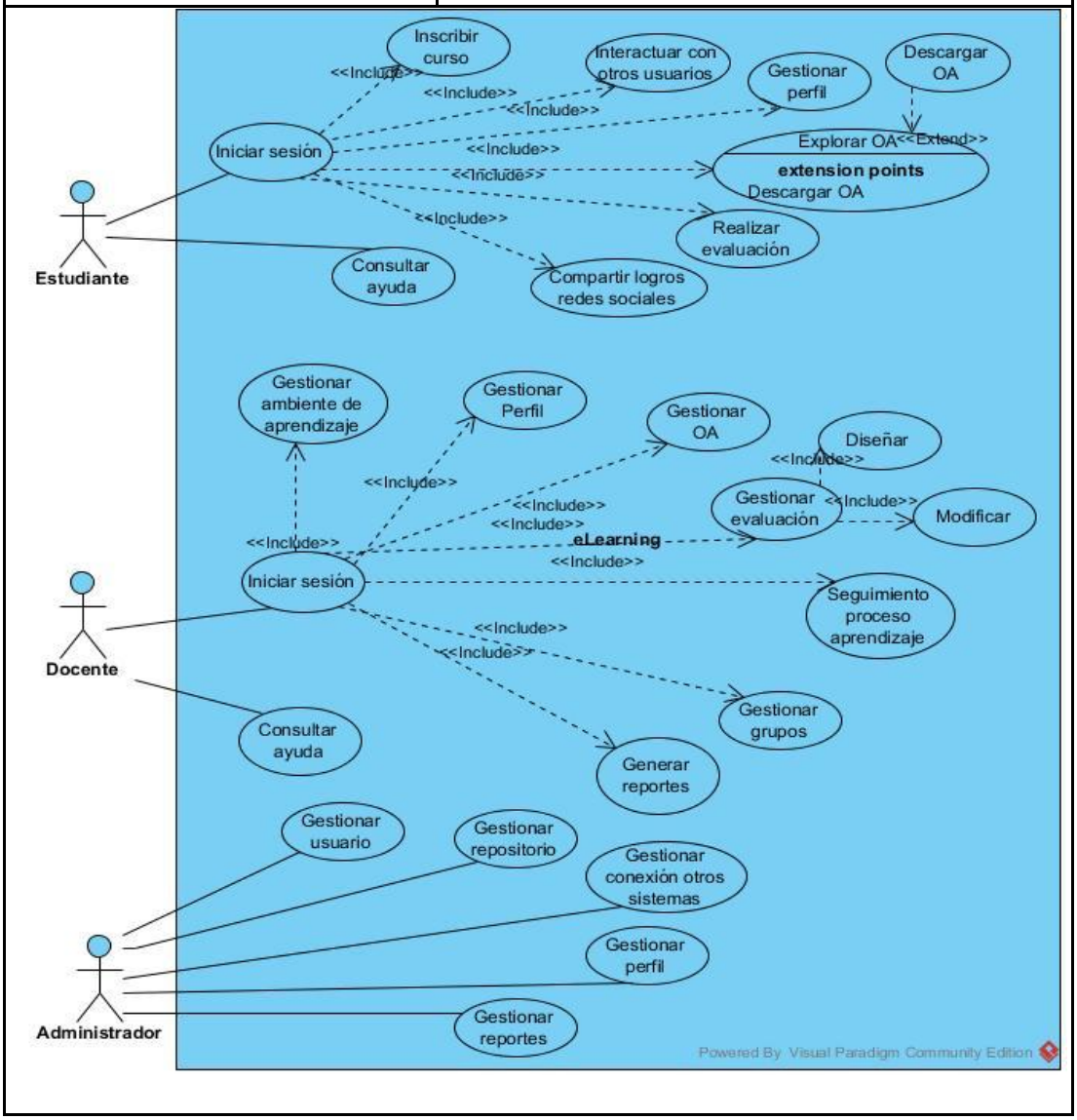


Tabla 55. Ejemplificación: Diagrama de Casos de uso de la aplicación
Fuente: Autora de la investigación (2015).

A continuación se construye una Lista de chequeo de requisitos de la aplicación (Tabla 56), para conjuntamente con los Diagramas de Casos de uso de la aplicación se verifiquen los requisitos propuestos para la aplicación, y en caso de encontrarse requisitos no contemplados, deberán incluirse.

InALPro	
ARTEFACTO	**DESCRIPCIÓN**
Nombre	Lista de chequeo de requisitos de la aplicación.
Constructor	Analista de requisitos
Nro. Identificación ACTIVIDAD	A_15
Nro. Identificación ARTEFACTO	30

Nro. Identificación	Descripción del requisito	Conforme (Si ó No)	Observaciones
1	Administrar objetos de aprendizaje en diferentes presentaciones.	Si	
2	Gestionar roles de usuarios.	Si	
3	Gestionar las actividades académicas y de intercambio de información entre participantes.	Si	
4	Gestionar la comunicación entre usuarios, por ejemplo a través foros, cuestionarios, chats, cartelera informativa y encuestas.	Si	
5	Gestión de evaluaciones de los objetivos de aprendizaje.	Si	
6	Gestionar ayuda en línea mediante diferentes recursos.	Si	
7	Registro y seguimiento de las actividades de docentes y estudiantes.	Si	
8	Notificar por correo electrónico sobre eventos, asignaciones, evaluaciones y actividades en general.	Si	
9	Generar reporte de actividad en la plataforma y reportes estadísticos.	Si	
10	Compartir en redes sociales los logros alcanzados por los estudiantes.	No	
11	Gestionar la comunicación de la plataforma eLearning con el sistema de control de estudios para obtener los datos de estudiantes y docentes.	No	
12	Cumplimiento de normativas y estándares institucionales para garantizar la disponibilidad del servicio.	Si	
13	Funcionamiento sobre plataforma de software libre.	Si	
14	El usuario es el único autorizado para compartir su información.	Si	
15	Cumplimiento de normas y estándares que permitan el acceso a los datos centralizados.	Si	
16	Garantizar la conexión al sistema al iniciar sesión.	Si	
17	Al completar una transacción el sistema garantiza que los datos se almacenaron correctamente en la base de datos.	Si	
18	La interacción entre el usuario y el sistema debe ser fluida, fácil de usar.	Si	
19	Se debe implementar ayudas en línea que respondan a las dudas del uso de la aplicación.	Si	
20	Cada usuario debe tener acceso solo a los servicios según su rol.	Si	

21	Los docentes y estudiantes solo pueden tener acceso a cursos a los cuales pertenecen.	Si	
22	En situaciones inesperadas de fallos, el sistema debe disponer de rutinas de recuperación y respaldo de los datos.	Si	

Tabla 56. Ejemplificación: Lista de chequeo de requisitos de la aplicación
Fuente: Autora de la investigación (2015).

Disciplina Diseño Arquitectónico de la Aplicación

El Diseño Arquitectónico de la Aplicación íntegra requisitos arquitecturales no contemplados en el dominio y válida que la arquitectura represente una solución adecuada para la aplicación de acuerdo a su complejidad, en caso de diferencia de componentes, se agrega el faltante y la información se envía a InDoCaS (elemento opcional) para ampliar el conocimiento del dominio. A continuación las actividades correspondientes a esta disciplina:

Actividad A_16: Revisión de la arquitectura de referencia.

A continuación se realiza una revisión de la Arquitectura de referencia (baseline) para la línea de producto (artefacto 19 de InDoCaS), como esta arquitectura es genérica, se busca definir su alcance y limitación para la aplicación según las reglas del negocio. Para esto se revisan el Soporte de decisión arquitectural (artefacto 16 de InDoCaS), el Documento de razonamiento arquitectural (artefacto 18 de InDoCaS), la Arquitectura base para la línea de productos (artefacto 19 de InDoCaS) y el Informe sobre la decisión arquitectural (Tabla 57). El resultado de esta revisión se obtiene en la tabla:

InALPro	
ARTEFACTO	**DESCRIPCIÓN**
Nombre	Alcance y limitación de la arquitectura de referencia.
Constructor	Arquitecto de software.
Nro. Identificación ACTIVIDAD	A_16
Nro. Identificación ARTEFACTO	31

Nro. Identificación	Aspecto alcanzado
1	La arquitectura de referencia plantea un esquema de seguridad conforme a lo solicitado por el cliente.
2	La centralización, acceso y control de los datos es un requisito exigido por el cliente.

3	La escalabilidad de la plataforma es un aspecto positivo dadas las condiciones de desarrollo del proyecto.
4	El cliente considera que es factible la implementación de esta arquitectura.

Nro. Identificación	Limitación observada
1	La arquitectura no contempla un medio de integración del sistema con otros sistemas existentes en la institución.
2	No se contempla la posibilidad de que los usuarios (estudiantes) tengan un medio para compartir sus logros a través de las redes sociales.
3	En casos de alto tráfico en la red la calidad del servicio puede verse afectada.

Tabla 57. Ejemplificación: Alcance y limitación de la arquitectura de referencia.
Fuente: Autora de la investigación (2015).

Actividad A_17: Ajustar la arquitectura de referencia.

Para realizar el ajuste de la arquitectura de referencia se toman en cuenta los puntos de variación, ya que estos referencian posibles cambios en la arquitectura de referencia que deben reflejarse en la arquitectura de la aplicación.

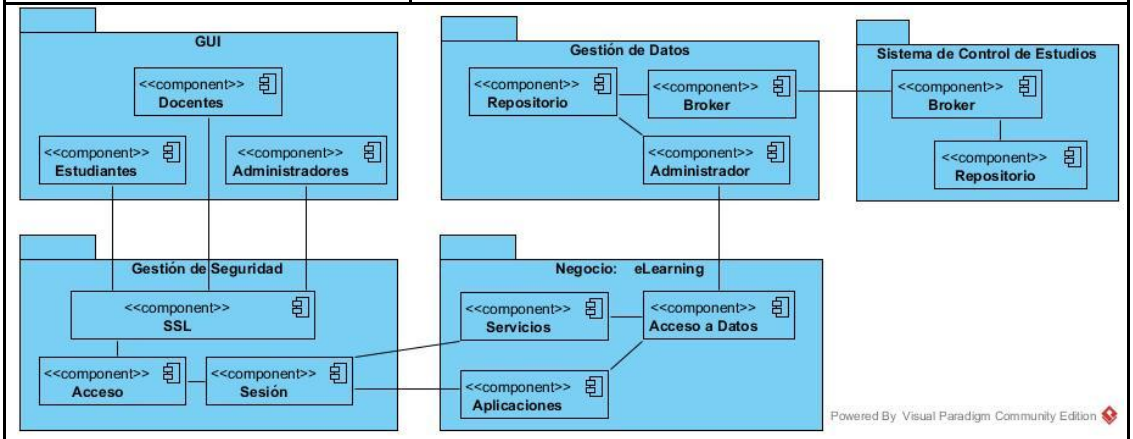
InALPro	
ARTEFACTO	**DESCRIPCIÓN**
Nombre	Lista de decisiones sobre los puntos de variación
Constructor	Arquitecto de software.
Nro. Identificación ACTIVIDAD	A_17
Nro. Identificación ARTEFACTO	32

Nro. Identificación	Característica	Decisión seleccionada
1	Publicación de información en las redes sociales.	Se decidió no incorporar esta característica al sistema debido a que no hubo consenso (por parte del cliente) respecto al uso de las redes sociales. Principalmente porque cuando se accedería a la plataforma desde la intranet institucional la conexión a las redes social estaría inhabilitado, es decir, este servicio funcionaría solo cuando el estudiante accede desde fuera de la institución.
2	Comunicación con el sistema de control de estudios.	Se decidió incorporar esta característica al sistema por ser de suma importancia para la migración y validación de datos de estudiantes y docentes.

Tabla 58. Ejemplificación: Lista de decisiones sobre los puntos de variación
Fuente: Autora de la investigación (2015).

Tomados en cuenta los puntos de variación, los requisitos funcionale/no-funcionales, se decide el siguiente planteamiento como solución arquitectural para el sistema eLearning en UPTP, ver Tabla 59.

InALPro	
ARTEFACTO	**DESCRIPCIÓN**
Nombre	Modelo arquitectural propuesto para la aplicación
Constructor	Arquitecto de software.
Nro. Identificación ACTIVIDAD	A_17
Nro. Identificación ARTEFACTO	33



GUI: Representa la capa de interacción con los usuarios de la plataforma, esta va a estar conformada por tres componentes según sus roles dentro del sistema.

Gestión de Seguridad: Es el responsable de gestionar el cifrado y descifrado de datos (SSL), de autorizar el acceso mediante la autenticación de usuarios, y la verificación de la sesión.

Negocio eLearning: Esta capa está compuesta por tres elementos, el componente de servicios que dispone las utilidades de la plataforma como ayuda, notificaciones, chats, y otros. El componente de aplicaciones tiene las funciones propias de la aplicación, como asignaciones, evaluaciones, cursos, objetos de aprendizaje y otros. Adicionalmente se encuentra un componente de acceso a datos que dispone de la lógica de negocio para acceder a la información.

Gestión de Datos: Conformado por el administrador de la base de datos responsable de los cambios en la BD y del uso de procedimientos almacenados y triggers. El Repositorio es la base de datos relacional donde se almacenarán los datos de la plataforma.

Broker: Es un conector que permite una comunicación asíncrona de acoplamiento flexible entre servidores de datos, garantizando que la información intercambiada entre brokers se entregue completa, es decir, es confiable la transmisión de datos.

Tabla 59. Ejemplificación: Modelo arquitectural propuesto para la aplicación
Fuente: Autora de la investigación (2015).

Actividad A_18: Evaluar arquitectura propuesta para la aplicación.

Definida la arquitectura de la aplicación, es evaluada y se decide implementar o no cambios en su estructura, el resultado de esta actividad se muestra en la Tabla 60.

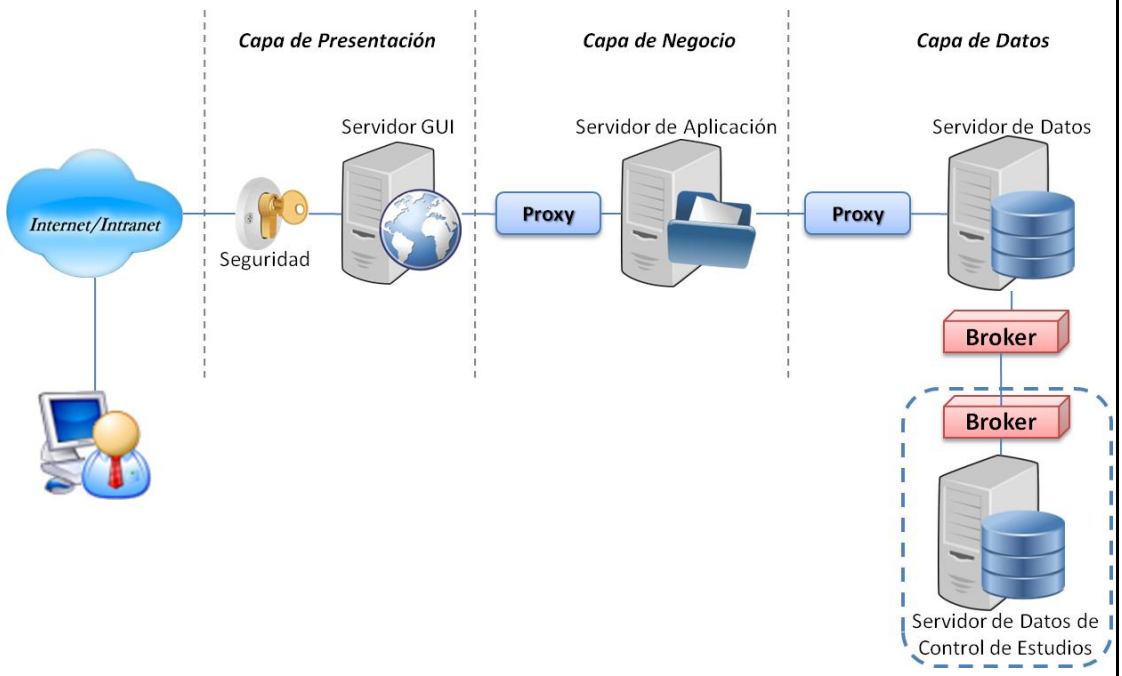
InALPro	
ARTEFACTO	**DESCRIPCIÓN**
Nombre	Lista de adaptaciones o incorporaciones a la arquitectura de la aplicación
Constructor	Arquitecto de software.
Nro. Identificación ACTIVIDAD	A_18
Nro. Identificación ARTEFACTO	34

Nro. Identificación	Modificación propuesta
1	Incorporar un conector para intercomunicar el Servidor de Control de Estudios con el Servidor de la Aplicación. Este conector propuesto es un bróker de servicio que permite una comunicación asíncrona entre el Servidor de la aplicación y el Servidor de control de estudios.

Tabla 60. Ejemplificación: Lista de adaptaciones o incorporaciones a la arquitectura de la aplicación
Fuente: Autora de la investigación (2015).

A continuación se incorporan las modificaciones a la arquitectura de la aplicación, Tabla 61.

InALPro	
ARTEFACTO	**DESCRIPCIÓN**
Nombre	Arquitectura para la aplicación
Constructor	Arquitecto de software.
Nro. Identificación ACTIVIDAD	A_18
Nro. Identificación ARTEFACTO	35



Arquitectura para la plataforma eLearning en UPTP:
La arquitectura está conformada por tres capas, la capa de presentación, la capa de negocio (eLearning) y la capa de datos.

La capa de presentación está compuesta por dos elementos:

-Un componente de seguridad (SSL, autenticación y autorización de acceso, y sesión): este elemento es responsable de autenticar los usuarios y permitirle el acceso al sistema.

-Un componente de interfaz gráfica de usuario (GUI): es el medio que interactúa con el usuario final, es responsable de mostrar y capturar la información generada por el usuario.

La capa de negocio:

Está constituida por el servidor de aplicaciones, donde residen dos tipos de servicios, las aplicaciones propias del sistema (cursos, objetos de contenido, evaluaciones, y otros) y las utilidades (chats, notificaciones, agenda, y otros).

También contiene un elemento proxy para establecer una única vía de comunicación entre los servidores de presentación y de negocios, esto con la finalidad de dar mayor seguridad a la plataforma.

La capa de datos:

Está conformada por el servidor de datos, donde reside la base de datos y el componente administrador de la base de datos.

Broker:

Es el conector que permitirá la comunicación entre el sistema de Control de Estudios y el sistema eLearning en UPTP, esto es necesario para la obtención de datos esenciales de docentes, estudiantes, cursos a los cuales pertenecen tanto estudiantes como docentes, y otros datos muy relevantes para el sistema eLearning. Mediante este bróker se pondrá contar con una comunicación asíncrona de acoplamiento flexible entre servidores de datos, garantizando que la información intercambiada es confiable.

Tabla 61. Ejemplificación: Arquitectura para la aplicación
Fuente: Autora de la investigación (2015).

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

La propuesta de un proceso para la ingeniería de la aplicación en líneas de producto de software, apoyado en la ingeniería de método situacional y enmarcado dentro del enfoque investigativo de ciencia del diseño permite llegar a las siguientes conclusiones:

1. La ingeniería del software es una disciplina que puede catalogarse como joven, es decir, a pesar de haber evolucionado a pasos agigantados requiere aún, bastante investigación en muchos de sus ámbitos, como por ejemplo en la reutilización de artefactos del proceso de desarrollo de software. Es el caso de la ingeniería de la aplicación, donde se revisaron métodos conocidos para su implementación y pocos de ellos están específicamente documentados, en este sentido, se considera que Gray Watch es el más apropiado para adaptar a este propósito.
2. La ingeniería de método situacional es un enfoque bastante versátil para el desarrollo de métodos y/o procesos de software, ya que es adaptable al entorno de desarrollo del proyecto y permite reutilizar partes o fragmentos de métodos ya probados en otros contextos. De esta manera facilita creación de nuevas propuestas metodológicas. Este enfoque fue el utilizado para especializar las disciplinas de la ingeniería de la aplicación, y el cual resultó muy adecuado, relativamente sencillo y practico de usar.
3. La incorporación de características de calidad a la ingeniería de la aplicación no garantiza la producción de software infalible, pero sí que la secuencia de sus actividades contemplen aspectos que influirán en la producción de un software más satisfactorio para el usuario. Incluir el modelo de calidad ISO/IEC 25010 en el proceso InALPro fue sencillo, debido a que el proceso InDoCaS ya lo contenía, lo que se hizo fue especializar el modelo de calidad del dominio para la aplicación.

4. La definición de un método de desarrollo es un proceso complejo, sin embargo, se considera la propuesta InALPro un aporte significativo al ambiente empresarial y académico en el ámbito de las líneas de producto de software donde se encuentran pocos planteamientos de este tipo. Lo anterior se afirma, puesto que de los métodos estudiados pocos son realmente orientados a las líneas de producto de software, la mayoría son adaptaciones poco explicitas de sus disciplinas.
5. Es factible aplicar InDoCaS e InALPro para implementar una línea de producto de software en otros dominios, debido al resultado satisfactorio en el dominio eLearning.
6. El proceso investigativo enmarcado dentro de la ciencia del diseño es ventajoso para estudios en el ámbito de la ingeniería del software, esto se debe a las características propias del proceso de software, es decir, los trabajos dentro de enfoques tradicionales de investigación son difíciles de realizar por la intangibilidad de los insumos y características de los artefactos producidos. Los fundamentos de la ciencia del diseño orientan la investigación al diseño de un producto que solventa un problema de la vida real.
7. La propuesta metodológica de integración de la ciencia del diseño y la ingeniería de método situacional, se considera útil y apropiada por su implementación versátil en el desarrollo de métodos o procesos de desarrollo de software.
8. La conjugación de la ciencia del diseño, la ingeniería de método situacional, el método Gray Watch e InDoCaS permitieron proponer en la disciplina de análisis de la aplicación dos (2) actividades que generan diez (10) artefactos y en la disciplina de diseño de la aplicación tres (3) actividades que generan cinco (5) artefactos para la ingeniería de la aplicación en líneas de producto de software. Esto constituye InAlPro, una ingeniería de la aplicación mejor documentada y con más posibilidades de reutilización de sus artefactos.

Recomendaciones

En base al presente estudio es recomendable lo siguiente:

1. Crear un repositorio de artefactos reutilizables que pueda ser compartido por InDoCaS e InALPro, de esta manera facilitar el acceso, organización y adaptación de activos de la línea de producto de software.
2. Evaluar otros métodos de diseño arquitectural que simplifiquen este proceso y estudiar la posibilidad de incorporarlo a InALPro.
3. Aplicar InDoCaS e InALPro a otros dominios para comparar resultados sobre su actuación.
4. La utilización de la ciencia del diseño como paradigma investigativo en el contexto de la ingeniería de software y la convergencia del uso de la ciencia del diseño con la ingeniería de método situacional para el desarrollo de nuevos métodos de producción de software.

GLOSARIO DE TÉRMINOS

Actividad: es la acción o tarea realizada por uno o más actores para alcanzar un objetivo definido.

Actor: responsable de una actividad o tarea dentro de un proceso de software.

Arquitectura de Software: según el documento de IEEE Std 1471-2000[14], es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución.

Artefacto: son todos los elementos o productos tangibles derivados del proceso de líneas de producto de software.

Atributo de calidad: característica de un producto o de un proceso mediante el cual se puede medir o comparar sus beneficios respecto a otro similar.

Desarrollo tradicional: De acuerdo a Piattini, M. y Garzás, J. (2007), enfoque donde cada producto se desarrolla con independencia del resto, con su propio equipo, ciclo de vida; ya sea con una aproximación estructurada u orientada a objetos.

Dominio: área de conocimiento que es estudiada a fin de caracterizarla para crear aplicaciones de software que proporcionan soluciones a sus necesidades de información.

Modelo: es la representación de una abstracción de una parte del mundo real.

Producto de la línea: cada uno de los sistemas de software que genera una línea de producto de software implantada.

Proceso de software: según Sommerville, I. (2005) es un conjunto de actividades y resultados asociados cuya meta es el desarrollo o evolución del software.

[14] http://standards.ieee.org/findstds/standard/1471-2000.html

REFERENCIAS BIBLIOGRÁFICAS

America, P., Obbink, H., Ommering, R. y Linden, F. (2000). CoPAM: A Component-Oriented Platform Architecting Method Family for Product Family Engineering. The First Software Product Line Conference (SPLC), Kluwer International Series in Software Engineering and Computer Science, Denver, Colorado, USA, 2000, p. 15. [En línea] Disponible: http://www.cin.ufpe.br/~esa2/papers/abstracts/art63.htm [Consulta: Julio 18, 2013]

Atkinson, C., Bayer, J., y Muthig, D. (2000). Component-Based Product Line Development: The KobrA Approach. [En línea] Disponible: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.25.8030&rep=rep1&type=pdf [Consulta: Julio 18, 2013]

Ávila, O., Estévez, A., Sánchez, E. y Roda, J., (2007). Combinando Modelos de Procesos y Activos Reutilizables en una Transición poco Invasiva hacia las Líneas de Producto de Software. XII Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2007), España: Thomson Editorial. [En línea] Disponible: http://lucentia.dlsi.ua.es/labcss/sites/default/files/Actas/ActasJISBD2007/JISBD-07-avila-combinando.pdf ISBN 978-84-9732-595-0 [Consulta: Marzo 14, 2012]

Balestrini, M., (2002). Cómo se elabora el Proyecto de Investigación. Tercera Edición. Caracas: B.L Consultores.

Barros, O., (1995). Reingeniería de Procesos de negocio: Un enfoque metodológico. Chile: Editorial Dolmen.

Bass, L., Clements, P., Cohen, S., Northrop, L. y Withey, J. (1997). "Product Line Practice Workshop Report". Technical Report CMU/SEI-97-TR-003 (ESC-TR-97-003), Software Engineering Institute. Carnegie Mellon University, Pittsburgh, Pennsylvania 15213 (USA). [En línea] Disponible: http://www.dtic.mil/cgi-bin/GetTRDoc?Location=U2&doc=GetTRDoc.pdf&AD=ADA327610 [Consulta: Abril 7, 2012]

Bass, L., Clements, P. y Kazman, R. (2003). Software Architecture in the practice. Second Edition. USA: Adisson-Wesley. P

Bayer, J., Flege, O., Knauber, P., Laqua, R., Muthig, D., Schmid, K., Widen, T., y DeBaud, J.-M. (1999) "PuLSE: A methodology to develop software product lines", Proceedings of the Symposium on Software Reusability. [En línea] Disponible: http://www.informatik.hs-mannheim.de/~knauber/Publications/1999.SSR_LosAngeles.pdf [Consulta: Julio 18, 2013]

Bermejo, M. Jesús (2007). Ingeniería de Líneas de Productos. Beneficio, Adopción, Experiencia y Futuro. Memorias de investigación de tesis doctoral. Universidad de Sevilla. España. [En línea] Disponible:

http://www.lsi.us.es/docs/doctorado/memorias/Jesus%20Bermejo%20%28Periodo%20Investigacion%29.pdf [Consulta: Julio, 17 2013]. (Pag. 10)

Botero, A., Giraldo, H. y Moyano, A. (2003) Limitaciones del desarrollo de aplicaciones en dispositivos móviles. Articulo. Pontificia Universidad Javeriana. Bogotá. [En línea] Disponible: http://pegasus.javeriana.edu.co/~mad/Articulo.pdf [Consulta: Julio, 18 2013] [En línea] Disponible: http://doc.utwente.nl/18012/1/Brinkkemper96method.pdf [Consulta: agosto, 26 2014]

Brinkkemper, S. (1996). Method Engineering: engineering of information system development method and tools. Information and Software Technology 38. (pp. 275-280).

Bukvova, H. (2009). "Research as a Process: A Comparison between Different Research Approaches". Sprouts: Working Papers on Information Systems, 9(29). [En línea] Disponible: http://sprouts.aisnet.org/9-29 [Consulta: agosto, 26 2014]

Canelón, R., (2010). Un proceso para la Ingeniería del Dominio basado en Calidad de Software. Una aplicación al dominio del aprendizaje móvil sensible al contexto. Tesis Doctoral. Universidad Central de Venezuela. Caracas.

Camarero, M., (2014). Análisis de metodologías para el ciclo de vida y desarrollo de productos. Proyecto fin de carrera/grado. Universidad Politécnica de Madrid. España. [En línea] Disponible: http://oa.upm.es/22319/ [Consulta: Marzo, 06 2014]

Cervera, M., Albert, M., Torres, V. y Pelechano, V. (2011, mayo). A State-of-the-Art Review on Method Engineering. Technical Report. Universidad Politécnica de Valencia. España. (pp. 7-8). [En línea] Disponible: http://www.pros.upv.es/technicalreports/PROS-TR-2011-13.pdf [Consulta: Marzo, 06 2015]

Chirinos, L., Losavio, F., y Matteo, A. (2004). Identifying Quality-Based Requeriments. Information systems Mangement. Editorial: Auerbach Publications.

Clements, P., y. Krueger, C., (2002) Being proactive pays / Eliminating the adoption barrier. IEEE Software, pages 28_31, July/August 2002. [En línea] Disponible: http://www.biglever.com/papers/PointCounterPoint.pdf [Consulta: Abril, 8 2012]

Clements, P., y Northrop, L., (2002) Software Product Lines: Practices and Patterns. USA: Addison–Wesley. (pp. 5-48)

Cross, N., Naughton, J. y Walker, D. (1981, Octubre) "'Design method and scientific method". Design Studies. Vol. 2. No. 4. pp. 195-201 [En línea] Disponible:http://designstudiesdiscourses.files.wordpress.com/2013/09/designmethodandscience.pdf [Consulta: Agosto, 8 2014]

Eriksson, H-E. y Penke, M. (2000) Business Modeling with UML. [En línea] Disponible:http://www.utm.mx/~caff/poo2/Business%20Modeling%20with%20UML.pdf [Consulta: Agosto, 8 2014]

Gacek, K., Schmid, K. y Clements, P. (2001). Successful Software Product Line Development in a Small Organization. A Case Study, Technical Report, Fraunhofer Institut for Experimental Software Engineering (IESE)

Gacenga, F., Cater-Steel, A., Toleman, M. y Tan, W-G. (2012) A Proposal and Evaluation of a Design Method in Design Science Research. The Electronic Journal of Business Research Methods Volume 10 Issue 2 2012 (pp 89-100) [En línea] Disponible: http://eprints.usq.edu.au/21624/ [Consulta: agosto, 26 2014]

Gacitúa, R. (2003). Métodos de desarrollo de software: el desafío pendiente de la estandarización. Theoria, Vol. 12: 23-42, 2003. [En línea] Disponible: http://www.soldance.com.ar/nqi/nqifiles/Metodologias.pdf [Consulta: agosto, 26 2014]

Gallardo, Y., y Moreno, A. (1999). Serie Aprender a Investigar. Modulo 3: Recolección de la Información. Bogotá: Arfo Editores Ltda.

Gehlert, A. y Pfeiffer, D. (2007) Utilizing Theories to Reduce the Subjectivity of Method Engineering Processes. IFIP Working Group 8.1 Working Conference on Situational Method Engineering: Fundamentals and Experiences. Switzerland (pp. 30-32). [En línea] Disponible: http://cuiwww.unige.ch/~ralyte/publications/ME07_PosterProceedings.pdf [Consulta: agosto, 27 2014]

Gomaa, H. (2011). Software Modeling & Desing. UML, use cases, patterns, and software architectures. USA: Cambridge University Press.

Gómez, E. (2012) Modelo arquitectural para aplicaciones móviles usando el enfoque de líneas de producción dinámica de software. Trabajo de Grado de Maestría. Universidad Centroccidental "Lisandro Alvarado". Barquisimeto.

González, J., Abrahão, S. y Insfrán, E (2011: Septiembre). Un enfoque Multi-modelo para la Introducción de Atributos de Calidad en el Desarrollo de Líneas de Producto Software. Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2011), Madrid. [En línea] Disponible: http://www.sistedes.es/ficheros/actas-conferencias/JISBD/2011.pdf [Consulta: Abril, 02 2013]

González, W. (2007) Las ciencias de diseño: racionalidad limitada, predicción y prescripción. (pp. 3-7) España: Netbiblio. [En línea] Disponible: http://goo.gl/lpy6RA [Consulta: Agosto, 12 2014]

Guendouz, A., y Bennouar, D. A (2013). Software Product Line For E-Learning Applications. The International Arab Conference on Information Technology. Sudan.

[En línea] Disponible: http://www.acit2k.org/ACIT/2013Proceedings/141.pdf [Consulta: enero, 04 2015]

Gutiérrez, J., Salazar, J., y López, N., (2008). Integración de una Herramienta de Administración de Configuraciones en una Línea de Producto de Software. Paradigma. En construcción del software. [Revista Electrónica] Universidad de los Andes. Colombia. Disponible: http://paradigma.uniandes.edu.co/index.php?option=com_content&view=article&catid=40:articulos&id=82:integracion-de-una-herramienta-de-administracion-de-configuraciones-en-una-linea-de-producto-de-software&lang=es [Consulta: Abril 01, 2012]

Günther, S. y Berger, T. (2008). Software Product Line Conference 2008. 12th International. [Presentación] Limerick, Ireland. Disponible: http://splc.net/prev-conferences/soapl-2008.pdf [Consulta: Abril 04, 2012]

Hamar, V. (2003). Aspectos metodológicos del desarrollo y reutilización de componentes de software. Trabajo de grado de maestría. Universidad de Los Andes. Mérida.

Hanssen, G., (2010). From Agile Software Product Line Engineering Towards Software Ecosystems. Tesis Doctoral. Norwegian University of Science and Technology. Trondheim, Noruega. [En línea] Disponible: http://www.idi.ntnu.no/research/doctor_theses/ghanssen.pdf [Consulta: Abril 2, 2012].

Harmsen, A. (1997). Situational Method Engineering. Tesis Doctoral. University of Twente. Pp. 22-39. [En línea] Disponible: http://eprints.eemcs.utwente.nl/17266/01/af_harmsen%5B1%5D.pdf [Consulta: Agosto 22, 2014].

Harmsen, A., Brinkkemper, S. y Oei, H. (1994, Septiembre). Situational Method Engineering for Information System Project Approaches. In: A.A. Verrijn Stuartand T.W. Olle (Eds.), Methods and Associated Tools for the Information Systems Life Cycle. Netherlands, September 1994, (pp. 169-194). [En línea] Disponible: http://goo.gl/Pd65jj [Consulta: Agosto 12, 2014].

Henderson-Sellers, B. y Ralyté, J. (2010) Situational Method Engineering: State-of-the-Art Review. Journal of Universal Computer Science, vol. 16, no. 3 (2010), (pp. 424-478) [En línea] Disponible: http://goo.gl/KnJqOz [Consulta: Agosto 2, 2014].

Henderson-Sellers, B., Ralyté, J., Ågerfalk, P. y Rossi, M. (2014). Situational Method Engineering. Springer: Berlin. [En línea] Disponible: http://goo.gl/LZwHc2 [Consulta: Agosto 12, 2014].

Hevner, A., y Chatterjee, S. (2010) Design Research in Information Systems. Theory and Practice. Springer. New York. pp. 9-21 [En línea] Disponible:

http://www.springer.com/business+%26+management/business+information+systems/book/978-1-4419-5652-1 [Consulta: Agosto 2, 2014].

Hevner, A., March, S., Park, J. y Ram, S. (2004). Design Science in Information Systems Research. MIS Quarterly Vol. 28 No. 1, pp. 75-105. [En línea] Disponible: http://www.brian-fitzgerald.com/wp-content/uploads/2014/05/Hevner-et-al-2004-misq-des-sci.pdf [Consulta: agosto, 26 de 2014]

Hill, G. (2009). "A Framework for Valuing the Quality of Customer Information" Tesis Doctoral. The University of Melbourne. Australia. [En línea] Disponible: https://docs.google.com/file/d/0Bzp-pBJZfEMJYTk0ZGEzNGYtMGZlNC00NGMxLTljNDktZmFhZDg4OTE3YTU0/edit?ddrp=1&hl=en [Consulta: agosto, 26 2014]

INTECO (2009, marzo). Ingeniería del Software: Metodologías y Ciclos de Vida. Guía del Laboratorio Nacional de Calidad del Software del Instituto Nacional de Tecnologías de la Comunicación. (p. 41).

Jones, C. (1996, Febrero). Software change management. Computer, Vol. 29, nro. 2, pp. 80-82. [En línea] Disponible: http://people.eecs.ku.edu/~saiedian/Teaching/Sp08/816/Papers/Background-Papers/change-mgmt.pdf [Consulta: septiembre, 21 de 2014]

Knoernschild, K. (2012) Java Application Architecture: Modularity Patterns with Examples Using OSGi. U.S: Prentice Hall.

Lehman, M.M. (1996). Laws of Software Evolution Revisited. In Proceedings of the 5th European Workshop on Software Process Technology (EWSPT '96). [En línea] Disponible: http://www.rose-hulman.edu/Users/faculty/young/CS-Classes/csse575/Resources/Lehman-LawsRevisited-96.pdf [Consulta: septiembre, 26 2014]

Leyvan, E., Prieto, J., Sampalo, M., y Garzón, M., (2006). Sistemas y aplicaciones informáticas. Temario. Primera Edición. España: Editorial Mad, S.L.

Linden, F., Schmid, K. & Rommes, E. (2007). Software Product Lines in Action. USA: Springer.

López, E., (2010). Programación Orientada a Característica Usando Clases Parciales en C#. Trabajo final de máster. Universidad de Málaga. España. [En línea] Disponible: http://tentecsharp.googlecode.com/svn/trunk/Elio/Doc/MemoriaTrabajoTutelado/Trabajo%20Final%20MasterISIA_Definitivo.pdf [Consulta: Marzo 10, 2012].

Losavio, F., y Guillen, C., (2010). Comparación de Métodos para la Arquitectura del Software: Un Marco de Referencia para un Método Arquitectónico Unificado. Revista de la Facultad de Ingeniería U.C.V., Vol. 25, N° 1, pp. 71–87 [En línea]

Disponible: http://www.scielo.org.ve/scielo.php?pid=S0798-40652010000100008&script=sci_arttext [Consulta: Marzo 10, 2012].

Lujan, S. (2002). Programación de aplicaciones web: Historia, principios básicos y clientes web. España: Editorial Club Universitario, pp. 48-59. [Libro en línea] Disponible: http://hdl.handle.net/10045/16995 [Consulta: mayo, 06 2014]

Montagud, S. (2009). Un Método para la Evaluación de la Calidad de Líneas de Productos Software basado en SQuaRE. Tesis de Máster en Ingeniería del Software. Universidad Politécnica de Valencia. España.

Montilva, J. (2006). Desarrollo de Software Basado en Líneas de Productos de Software. IEEE Computer Society Region 9. Capitulo Argentina. Programa DVP. [En línea] Disponible: http://www.ieee.org.ar/downloads/2006-montilva-productos.pdf [Consulta: Marzo 28, 2012].

Montilva, J., Arapé N. y Colmenares J. (2003, Noviembre). Desarrollo de Software Basado en Componentes. Artículo publicado en la actas del IV Congreso de Automatización y Control, Mérida.

Montilva, J., Barrios, J. y Rivero, M. (2008). GRAY WATCH: Método de desarrollo de software para aplicaciones empresariales. Proyecto METHODIUS. FONACIT 2005000165. Mérida. [En línea] Disponible: http://www.codecompiling.net/files/slides/IS_clase_13_GRAY_WATCH_Octubre_2008_V1.pdf [Consulta: Marzo 28, 2013].

Montilva, J. Montilva, W. y Barrios, J. (2011). Blue WATCH: Un marco metodológico balanceado para el desarrollo de software en Pequeñas Empresas. Latin American and Caribbean Conference. artículos en actas o proceedings de conferencias internacionales arbitradas Medellín Colombia. En línea] Disponible: http://www.laccei.org/LACCEI2011-Medellin/RefereedPapers/It192_Montilva.pdf [Consulta: Marzo 8, 2014].

Nandhakumar, J. y Avison, J. (1999). The fiction of methodological development: a field study of information system development. Information Technology & People 12(2): 176-191. [En línea] Disponible: http://www.interruptions.net/literature/Nandhakumar-ITP99.pdf [Consulta: septiembre 21, 2014]

Offermann, P., Levina, O., Schonherr, M. y Bub, U. (2009). "Outline of a Design Science Research Process" Disponible: http://www.researchgate.net/publication/221581320_Outline_of_a_design_science_research_process [Consulta: agosto, 26 2014]

Parnas, D. y Clements, P. (1986). A rational design process: How and why to fake it. IEEE Transactions on Software Engineering. 12(2): 251-257. [En línea]

Disponible: http://www.ics.uci.edu/~taylor/classes/121/IEEE86_Parnas_Clement.pdf [Consulta: septiembre 21, 2014]

Parra A., C. (2007). Una línea de productos basada en modelos. Tesis de Maestría. Universidad de los Andes. Colombia. (p. 5). [En línea] Disponible: http://researchers.lille.inria.fr/~cparra/publications/thesismaster07.pdf [Consulta: Febrero 21, 2014]

Pérez, J. y Sánchez, I. (2012). Hacia la extensión del Método Gray Watch basado en el Estándar de Calidad ISO/IEC 25010. Publicaciones en Ciencias y Tecnología. Año 2012. Enero-Junio. Vol 6, Nro 1, pp 5-19. [En línea] Disponible: http://bibcyt.ucla.edu.ve/edocs_bciucla/pcyt/pcyt06/pcyt060101.pdf [Consulta: septiembre, 06 2014]

Piattini, M. y Garzás, J. (Eds.) (2007). Fábricas de Software: experiencias, tecnologías y organización. Segunda Edición. Madrid: Editorial Ra-Ma. (pp. 61-79)

Pfeiffer, D. (2008) "Semantic Business Process Analysis. Building Block-based Construction of Automatically Analizable Business Process Models" Tesis Doctoral. Westaflische Wihelms-Universitat. Munster. [En línea] Disponible: http://udoo.uni-muenster.de/downloads/publications/2076.pdf [Consulta: agosto, 26 2014]

Pohl, K., Böckle, G., y Linden, F. (2005). Software Product Line Engineering: Foundations, Principles, and Techniques. Alemania: Springer-Verlag.

Pressman, R., (2010). Ingeniería del Software. Un enfoque práctico. Séptima Edición. México: McGraw-Hill Interamericana.

Product Line Engineering (2013). [Página Web] Disponible: http://www.productlineengineering.com/index.html [Consulta: Febrero 21, 2014]

Ralyté, J., (2002). Requirements Definition for the Situational Method Engineering. Proceedings of the IFIP WG8.1 Working Conference on Engineering Information Systems in the Internet Context (EISIC'02). Japon, septiembre 2002. pp.127-152. [Articulo en línea] Disponible: http://cuiwww.unige.ch/~ralyte/publications/jralyte_EISIC02.pdf [Consulta: marzo, 31 2015]

Ralyté, J., Deneckère, R. y Rolland, C. (2003). Towards a Generic Model for Situational Method Engineering. CAISE, Velden: Austria. [En línea] Disponible: http://hal.archives-ouvertes.fr/docs/00/70/36/07/PDF/SME_CAISE03-submited_.pdf [Consulta: agosto, 24 2014]

Rivero, L., (2011). Una línea de producción de Software para sistemas transaccionales. Una aplicación al proceso de desarrollo de software en la Coordinación Nacional de Tecnología de Información de la UNEXPO. Trabajo de Grado de Maestría. Universidad Centroccidental “Lisandro Alvarado”. Barquisimeto.

Rivero, M., Montilva, J., Barrios, J., Murúa, M. y Granados, G. (2009). Un análisis del desarrollo de software en empresas venezolanas. Seventh LACCEI Latin American and Caribbean Conference for Engineering and Technology. San Cristobal, Venezuela.

Ruiz-Rube, I., Dodero, J., y Ruiz, M. (2014). Ingeniería Dirigida por Modelos como soporte a la gestión de procesos software. Trabajo incluido en las actas de las II Jornadas Predoctorales de la Escuela Superior de Ingeniería. Universidad de Cádiz (JORPRESI 2010). [En línea] Disponible: http://hdl.handle.net/10498/15861 [Consulta: agosto, 26 2014]

Sametinger, J. (1997) Software engineering with reusable components. Germany: Springer Verlag. (p. 11). [Libro en línea] Disponible: http://books.google.co.ve/books?id=AxPQvGRs2wUC&printsec=frontcover#v=onepage&q&f=false [Consulta: Julio, 22 2013]

Sánchez, P., García-Saiz, D. y Zorrilla, M. (2013). Construyendo Familias de Productos Software para Plataformas e-Learning: Un Caso de Estudio. IEEE - VAEP-RITA. Volumen 1, Número 2 Págs. 105-112. [Artículo en línea] Disponible: http://rita.det.uvigo.es/VAEPRITA/index.php?content=Num_Pub&idiom=Es&visualiza=3&volumen=1&numero=2&articulo=5 [Consulta: Enero, 08 2015]

Simon, H. (1996) The sciences of the artificial. 3rd edition. The MIT Press, Cambridge, MA. (p. 11). [Libro en línea] Disponible: http://m.friendfeed-media.com/092e5a73c91e0838eeb11e0fe90edaf9e9afc065 [Consulta: Agosto, 20 2014]

Shaw, M., y Garlan, D. (1996). Introduction to Software Architectures. New perspectives on an emerging discipline. New Jersey: Prentice Hall.

Software Engineering Institute (SEI). Carnegie Mellon. [Página Web] Disponible: http://www.sei.cmu.edu/ [Consulta: Febrero 1, 2012]

Soler, J., Prados, F., Poch, J. y Boada, I. (2012). ACME: Plataforma de Aprendizaje Electrónico (e-learning) con Funcionalidades Deseables en el Ámbito de la Ingeniería. Formación universitaria. [En línea] Disponible: http://www.scielo.cl/scielo.php?pid=S0718-50062012000300002&script=sci_arttext&tlng=pt [Consulta: Mayo 19, 2014]

Sommerville, I., (2005). Ingeniería del software. Séptima Edición. Madrid: Pearson Addison-Wesley.

Takeda, H., Veerkamp, P., Tomiyama, T., y Yoshikawam, H. (1990). "Modeling Design Processes" AI Magazine 1990 Winter, (pp. 37-48). [En línea] Disponible: http://aaaipress.org/ojs/index.php/aimagazine/article/download/855/773 [Consulta: Agosto 09, 2014]

The Standish Group. Chaos Manifesto 2013, Think Big, Act Small. [En línea] Disponible: https://secure.standishgroup.com/reports/reports.php [Consulta: Mayo 29, 2014]

Valdezate, A. (2010). "Gestión de la variabilidad en líneas de producto dinámicas" Trabajo fin de master. Universidad Rey Juan Carlos. España.

Vaishnavi, V. y Kuechler, W. (2004). "Design Science Research in Information Systems," January 20, 2004 (Última actualización: 23 de octubre de 2013); [En línea] Disponible: http://www.desrist.org/design-research-in-information-systems/ [Consulta: Agosto 09, 2014]

Vaishnavi, V., Kuechler, W. y Kuechler, W. Jr. (2007). "Design Science Research Methods and Patterns: Innovating Information and Communication" USA: Taylor & Francis Group. [En línea] Disponible: http://books.google.co.ve/books?id=sI2Y9Jh8tq8C&printsec=frontcover#v=onepage&q&f=false [Consulta: Agosto 09, 2014]

Winter, R. (2008). Design science research in Europe. European Journal of Information Systems. (2008) 17, (pp. 470–475). doi:10.1057/ejis.2008.44 [En línea] Disponible:http://www.palgrave-journals.com/ejis/journal/v17/n5/pdf/ejis200844a.pdf [Consulta: Agosto 19, 2014]

Universidad Pedagógica Experimental Libertador (2006). Manual de Trabajos de Grado de Especialización y Maestría y Tesis Doctorales. Caracas: FEDUPEL.

Weiss, D. y Lai, R. (1999). Software Product Line Engineering: A Family Based Software Development Process; USA: Addison-Wesley.

Zabala, S. (2013). Perfil por competencias del profesional de la informática: una visión desde la perspectiva del mercado empleador venezolano. Revista Espacios. Vol. 34 (6) 2013. Pág. 7. 44 [En línea] Disponible: http://www.revistaespacios.com/a13v34n06/13340607.html [Consulta: Agosto 29, 2014]

CURRÍCULO VITAE DEL AUTOR

Sol Mirella Ovalles Hernández

Cursante del postgrado en Ciencias de la Computación de la Universidad Centroccidental "Lisandro Alvarado". Nació en Acarigua, estado Portuguesa, el 13 de abril de 1974. Estudió educación primaria en la Escuela Básica "Miguel Otero Silva", seguidamente, estudió educación básica-diversificada en la Unidad Educativa Nacional "Eduardo Cholle Boada" ambas en Acarigua, en esta última obtuvo el título de Bachiller en Ciencias, recibe diploma por excelente rendimiento académico al destacarse por tener el segundo mejor promedio de su promoción. A continuación, estudia en el Instituto Universitario Politécnico de las Fuerzas Armadas Nacionales (IUPFAN), donde obtiene el título de Técnico Superior en Análisis y Diseño de Sistemas en el año 1994. En el año 2003 culmina Capacitación para profesionales no Docentes (Componente Docente) en la Universidad Pedagógica Experimental Libertador. Más adelante, continua sus estudios en la Universidad Centroccidental "Lisandro Alvarado" donde obtiene el título de Ingeniero en Informática, destacándose al obtener la distinción "Cum Laude", integrar el Cuadro de Honor en los lapsos académicos 2006-II, 2006-I, 2005-I, 2004-II, 2004-I, 2003-II, 2003-I, 2002-I y recibe certificados por calificación sobresaliente (19 o 20 puntos) en las asignaturas: Programación no Numérica II, Programación no Numérica I, Laboratorio II, Investigación de Operaciones I, Sistemas III, Estructuras Discretas II, Ingles II, Técnicas de Estudio y Problemática del Desarrollo. Posee experiencia laboral como Analista del Registro de Importaciones y posteriormente como Analista de Compras en el Central Azucarero Portuguesa, C.A., desde agosto de 1994 a agosto de 1998, desde agosto de 1998 hasta la fecha se ha desempeñado como Programador I y Programador II en la UPTP "Juan de Jesús Montilla", en esta misma institución ejerció como Docente contratado a tiempo convencional desde octubre 2010 a octubre 2014. Continuando con la actividad docente ejerce como Docente Contratado medio tiempo en UCLA desde julio 2014.