UNIVERSIDAD CENTROCCIDENTAL
"LISANDRO ALVARADO"

MODELO ARQUITECTURAL PARA UNA LÍNEA DE PRODUCCIÓN DE SOFTWARE QUE INTEGRA LAS INGENIERÍAS DEL DOMINIO Y APLICACIÓN USANDO InDoCaS

DEYANIRETH DUARTE MARIN

Barquisimeto, 2012

UNIVERSIDAD CENTROCCIDENTAL "LISANDRO ALVARADO"
DECANATO DE CIENCIAS Y TECNOLOGÍA
POSTGRADO EN CIENCIAS DE LA COMPUTACIÓN

MODELO ARQUITECTURAL PARA UNA LÍNEA DE PRODUCCIÓN DE SOFTWARE QUE INTEGRA LAS INGENIERÍAS DEL DOMINIO Y APLICACIÓN USANDO InDoCaS

Proyecto de Trabajo Grado presentado para optar al título de
Magister Scientiarum en Ciencias de la Computación

Por: DEYANIRETH DUARTE MARIN
TUTOR: DR. RODOLFO CANELÓN OSAL

Barquisimeto, 2012

MODELO ARQUITECTURAL PARA UNA LÍNEA DE PRODUCCIÓN DE SOFTWARE QUE INTEGRA LAS INGENIERÍAS DEL DOMINIO Y APLICACIÓN USANDO InDoCaS

Por: DEYANIRETH COROMOTO DUARTE MARIN

Trabajo de grado aprobado

_____ _____

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

_____

(Jurado 3)

Barquisimeto, ___ de _____ de 2012

DEDICATORIA

A Dios,
Mi esposo,
Mi hijo,
Mis padres y
Hermano

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE CUADROS

UNIVERSIDAD CENTROCCIDENTAL "LISANDRO ALVARADO"
DECANATO DE CIENCIAS Y TECNOLOGÍA
POSTGRADO EN CIENCIAS DE LA COMPUTACIÓN

MODELO ARQUITECTURAL PARA UNA LÍNEA DE PRODUCCIÓN DE SOFTWARE QUE INTEGRA LAS INGENIERÍAS DEL DOMINIO Y APLICACIÓN USANDO InDoCaS

Autor: Ing**.** Deyanireth Duarte Marin

Tutor: Dr. Rodolfo Canelón Osal

RESUMEN

La presente investigación tiene como objetivo plantear un modelo arquitectural para una línea de producción que permita la integración de las ingenierías del dominio y de la aplicación usando al proceso InDoCaS como guía en el enfoque de calidad. El proceso InDoCaS propone un modelo de desarrollo para familias de productos en un dominio y la Ingeniería de la Aplicación se basa en la reutilización de componentes existentes y los planes de producción para el desarrollo de aplicaciones; las aplicaciones que componen las ingenierías del dominio y de la aplicación son desarrolladas con tecnologías diferentes haciendo inmanejable la comunicación entre las mismas. La arquitectura de software para una línea de producción debe cumplir con ciertos atributos de calidad como son la integrabilidad y la interoperabilidad. En este sentido, se estudiará la definición de un Middleware como solución arquitectural para facilitar la integración e interoperabilidad entre las diferentes aplicaciones utilizadas en la construcción de líneas de producción de software, tecnologías basadas en servicios y las mejores prácticas de la ingeniería de software, serán incorporadas como componentes y conectores para la mediación de servicios de enrutamiento, transformación de mensajes y soportes para los diversos protocolos, simplificándose el trabajo de desarrolladores. Asimismo, se definirá un repositorio que permitirá implementar las estructuras, formatos de los artefactos y técnicas de las aplicaciones de las ingenierías del dominio representada por el proceso InDoCaS y de la aplicación. Esta investigación está enmarcada dentro de la modalidad de proyecto factible.

Palabras Claves: Líneas de producción de software, ingeniería de dominio, ingeniería de la aplicación, InDoCaS, calidad de software, integración de aplicaciones, arquitectura de software, middleware.

INTRODUCCIÓN

La sociedad y las organizaciones en general, están definiendo la forma de integrarse con su entorno en un ambiente que tiende a ser global, con el uso de la tecnología de información donde los resultados dependen cada vez menos del tiempo y la distancia, considerando al intercambio de información como uno de los elementos claves para el logro de los objetivos. Por tal razón, uno de los aspectos más desafiantes de cualquier organización es lograr el intercambio de información dentro, desde y hacia fuera de la misma a través de las tecnologías de información.

Los sistemas de información han evolucionado con el fin de responder a los nuevos requisitos organizacionales. Sin embargo, a pesar de los avances producidos, todavía no responde totalmente a las necesidades de los usuarios. Cabe destacar que la mayoría de los productos de software están basados en plataformas de software heterogéneos, por lo tanto, no tienen la capacidad requerida para integrarse e interoperar. Es por ello que la mayoría de las aplicaciones no alcanzan sus objetivos, y como consecuencia de esto, existen altos porcentajes de rechazo de los mismos y disminución en gran medida de su operatividad. Esta situación puede afectar a la industria de desarrollo de software en cuanto a la capacidad de las aplicaciones y la necesidad de crear eficientes productos para la satisfacción de sus clientes. En consecuencia, seguir con el mismo esquema de tecnología, puede generar retrasos en la entrega del producto, pérdidas de información, duplicidad de esfuerzo, costos excesivos y limitaciones para la distribución del trabajo. Es por ello que la construcción de la arquitectura del software es fundamental en el desarrollo del mismo, ya que representa una propuesta de solución para los requisitos funcionales y no funcionales establecidos en las aplicaciones que determinarán la calidad del software.

Las líneas de productos de software son una forma de implementar la variabilidad del software que consiste en crear eficientemente diferentes productos o aplicaciones

de un dominio, establecido por la ingeniería del dominio la cual permite reutilizar un conjunto de artefactos centrales de una manera pre-establecida que abarca todas las actividades para la creación de modelos de dominios, arquitecturas, componentes y artefactos de software reutilizables. Asimismo, la Ingeniería de la aplicación se basa en la reutilización de componentes existentes, en el conocimiento del dominio y en las necesidades del cliente para construir nuevas aplicaciones. La arquitectura de una línea de productos es el activo reutilizable más valioso para la organización porque es una referencia para la creación de productos de software en un dominio. Una de las alternativas de solución al problema antes mencionado consiste en definir un middleware que incorpore tecnologías basadas en servicios y un componente de mediación para la integración de las ingenierías del dominio y de la aplicación de la línea de producción.

Desde donde surge la presente investigación enfocada en obtener un modelo arquitectural de software que cumpla con los requisitos de integración e interoperabilidad establecidos en el middleware, el cual permitirá la configuración de la línea de producción mediante aplicaciones que constituyen la Ingeniería del dominio representada por el proceso InDoCaS y la ingeniería de la aplicación con el objetivo fundamental de intercambiar los activos de software compuestos de: requisitos, diseños, componentes, casos de prueba, arquitectura, documentación, entre otros. Al mismo tiempo, las técnicas y los activos de software necesitan ser almacenados, intercambiados y transformados para ser usados por las aplicaciones.

La integración de las aplicaciones de la ingeniería del dominio representada por el proceso InDoCaS y de la Ingeniería de la aplicación se considera importante porque aporta los activos para la línea de productos de software como factor principal para satisfacer las necesidades de la academia, la industria del software y los desarrolladores; asegurando el éxito de los proyectos emprendidos, así como de su compromiso para suministrar productos de calidad. A continuación, se describe la estructura del presente proyecto la cual está constituida de la siguiente manera:

Capítulo I El Problema. Se presenta el planteamiento del problema, se establece objetivo general y objetivos específicos, se expone la justificación, alcances y limitaciones de la investigación.

Capítulo II Marco Teórico. Se exponen los antecedentes del estudio, las bases teóricas, en este caso se describen: Arquitectura de Software, Estilos Arquitecturales, Patrones Arquitecturales, Patrones de Diseño, Líneas de Producción de Software (LPS), Ingeniería del Dominio, Ingeniería de la Aplicación, Gestión de Variabilidad, InDoCaS, Integración de Aplicaciones, Middleware, Plataformas Middleware, Servicios Web, Arquitectura Orientada a Servicios (SOA), Bus de Integración de Aplicaciones (ESB)., Repositorios de Datos. Además se presenta la operacionalización de la variable, sobre los cuales se fundamentará la descripción, el análisis y la evaluación de los hallazgos de la investigación.

Capítulo III Marco Metodológico. Se describe la metodología que se empleará para la ejecución del estudio.

Capítulo IV Propuesta del Estudio. Se indica la justificación, los objetivos tanto general como los específicos, descripción de la propuesta y posteriormente se desarrollan los objetivos específicos.

Capítulo V Conclusiones y Recomendaciones. Se presentan las conclusiones y recomendaciones derivadas del presente trabajo de investigación.

CAPÍTULO I

EL PROBLEMA

Planteamiento del Problema

Actualmente, la sociedad está transitando por nuevos escenarios sistémicos que traspasan las fronteras de las organizaciones, donde la globalización, el trabajo colaborativo, la transformación del mercado mundial, la integración, constituyen el nuevo ordenamiento de las estructuras sociales de avanzada en el siglo XXI.

De esta manera, la relación entre el hombre, la sociedad y la tecnología, produce un nuevo sentido a las instituciones, debido a que el momento socio histórico actual, permite a las organizaciones humanas sistematizar los modos de comunicación, adaptación, desarrollo de habilidades y destrezas de nuevos métodos de trabajo y conocimiento, tanto en la sociedad como en las organizaciones, el nuevo hombre socialmente es cibernético, donde el compromiso e interés por las tecnologías se proyecta de modo creciente e indetenible.

Según Rivero (2011), uno de los activos más importantes de cualquier organización es la información, es por ello, que éstas deben tener una infraestructura de información estable a través del uso de las tecnologías de información. Según el mismo autor, "los sistemas de información buscan la forma de mejorar el uso de la tecnología que soporta el flujo de información dentro de la organización" (p.16).

Evidentemente, la mayoría de las empresas tienen diversos sistemas de información que en opinión de Rivero (ob. cit.), "son y serán la piedra fundamental de toda organización para la toma de decisiones" (p.66), los cuales dan soporte a los procesos de negocios, que son la base clave para el logro de los objetivos planteados.

Por su parte, Grimán et al. (2007), expresan que los sistemas de información se han convertido en uno de los componentes críticos del negocio. Acotan además, que estos en los Sistemas Empresariales, se han desarrollado con base en diversas tecnologías, produciendo así "islas de aplicaciones" que requieren de su integración para mejorar el flujo de información entre los procesos involucrados. Todo ello, se transcribe en una ventaja competitiva para la empresa al reducir el tiempo de respuesta de estos procesos, por lo que es deseable que en muchos casos críticos, las aplicaciones integradoras de estos sistemas de información cumplan con ciertos atributos de calidad.

Cabe destacar, que la calidad de los productos de software representa un reto muy importante para la ingeniería de software. Según Gómez (2011), la calidad del software "se define como el conjunto total de características de un sistema que le confieren la capacidad de satisfacer las necesidades establecidas y las necesidades implícitas de los usuarios" (p.47).

Para muchas empresas es una necesidad la integración de las aplicaciones, debido al crecimiento en el desarrollo de sistemas para automatizar sus operaciones y a la proliferación de distintas plataformas tecnológicas que se utilizan para tal fin.

Asimismo, Rodríguez et al. (2010) indicó que los problemas de integración se deben a múltiples y diferentes: aplicaciones, plataformas, bases de datos, procesadores de transacciones, entradas de datos, versiones de los mismos datos y datos del negocio incompatibles. El autor también refiere, que los factores que contribuyen a los problemas son: la construcción de aplicaciones en tiempos diferentes y con grupos diferentes que desarrollan independientemente; organizaciones con arquitecturas incompatibles y aplicaciones legadas difíciles de mantener y eliminar; organizaciones que prefieren la adquisición de aplicaciones empaquetadas sobre el desarrollo interno.

Considerando la información anterior, es aquí donde cobra importancia un Middleware. Garimella et al. (2008), señalan que el propósito de un Middleware es "facilitar la comunicación y movilidad de los datos entre diferentes aplicaciones de

tecnologías de la información" (p.41). En efecto, las plataformas Middleware permiten dar solución a la integración de sistemas inter-empresa o intra-empresas con el objetivo de intercambiar información entre las aplicaciones heterogéneas de manera confiable. Del mismo modo, Mondejar (2010) define un Middleware como la "integración e interoperabilidad de la aplicación y los servicios que se ejecutan en dispositivos de computación y comunicación heterogéneos" (p.09).

De acuerdo a lo planteado por Rodríguez et al. (2010), La integración de aplicaciones mejora los sistemas de información existentes, conecta las diversas "islas de automatización", adopta un enfoque de cambios e integra sistemas a través de plataformas tecnológicas; y la interoperabilidad "es la habilidad de diversos sistemas y organizaciones para trabajar juntos" (p.22), a nivel tecnológico esta capacidad se puede obtener a través de un Middleware que posibilite el intercambio de información entre las aplicaciones involucradas.

Ante este hecho, las tecnologías de información son construcciones que tienen sus características en el entender, predecir e interpretar los fenómenos humanos, lo que significa, que la virtualidad representa una forma de manifestación de la cultura del hombre postmoderno.

De acuerdo a Cabello (2008), la Ingeniería del Software debe proporcionar herramientas y métodos para desarrollar un conjunto de sistemas con distintas capacidades y adaptables a situaciones variables. Ante esta situación, surge el concepto de las líneas de producción de software "se refiere a las técnicas de ingeniería para la creación de una cartera de sistemas de software similares de un conjunto común de activos de software utilizando un medio común de producción" (Montilva, 2007). Los procesos que deben ser desarrollados en la construcción de una línea de producción de software son: la ingeniería del dominio y la ingeniería de la aplicación. Ver figura 1.

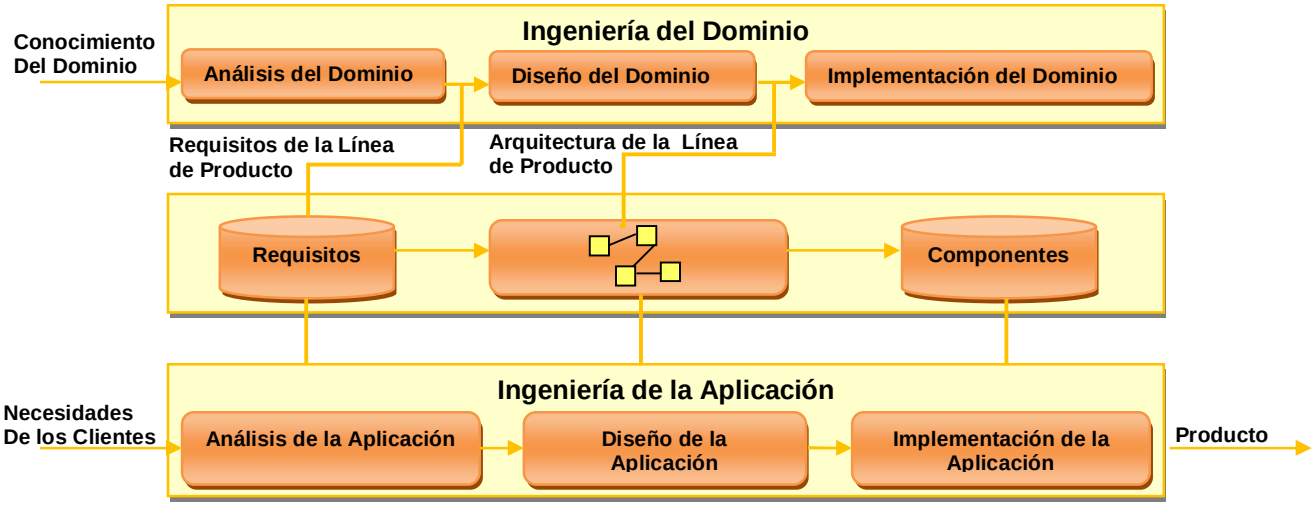


Figura 1. *Modelo de procesos para el desarrollo de software basado en componentes.*

A través de la figura anterior, puede apreciarse que la ingeniería del dominio está compuesta por tres subprocesos (análisis, diseño e implementación del dominio), asimismo la ingeniería de la aplicación abarca tres subprocesos (análisis, diseño e implementación de la aplicación).

El Proceso para la Ingeniería de Dominios basado en la Calidad de Software (InDoCaS), es una especialización de la ingeniería del dominio con el objetivo de obtener una arquitectura base para una familia de productos; las principales disciplinas de este proceso son análisis, diseño del dominio (Canelón, 2010); y la implementación del dominio como una expansión al proceso InDoCaS (Rivero, 2011). Mientras que la Ingeniería de la Aplicación, se encarga del desarrollo de aplicaciones basado en la reutilización de componentes existentes y los planes de producción; abarca tres subprocesos: análisis, diseño e implementación (Montilva, 2007). Cada subproceso tiene actividades, artefactos de entrada, artefactos de salida y técnicas. Los artefactos de salida de un subproceso pueden ser entradas de otro subproceso, es decir, existe cierta dependencia entre subprocesos para realizar algunas actividades. El aporte de InDoCaS para el área del desarrollo de software

basado en líneas de producción de software es un proceso para la ingeniería del dominio basado en calidad de software, que incorpora subprocesos en las fases de análisis y diseño del dominio y presenta detalles en las actividades de análisis y diseño de la arquitectura conceptual.

A través de los planteamientos anteriores, puede apreciarse que la mayoría de las aplicaciones son diseñadas de manera diferente, es decir, no se tiene una infraestructura común, son heterogéneas a nivel hardware y software (base de datos, lenguaje de programación, middleware de integración, sistema operativo, entre otros).

De lo anterior, es propio mencionar que las aplicaciones de la Ingeniería del dominio representada por el proceso InDoCaS y la Ingeniería de la Aplicación, no se encuentran integradas y no pueden interoperar debido al uso de tecnologías de hardware y software heterogéneas. Asimismo, las técnicas y artefactos utilizados en dichas aplicaciones no se encuentran disponibles en repositorios de datos centralizados lo cual impide que las aplicaciones no puedan intercambiar información.

Esta situación afecta a la industria de desarrollo de software en cuanto a la capacidad de las aplicaciones de las ingenierías del dominio representada por el proceso InDoCaS y de la aplicación en crear eficientes productos para la satisfacción de sus clientes. En consecuencia, seguir con el mismo esquema de tecnología, puede generar retrasos en la entrega del producto, pérdidas de información, duplicidad de esfuerzo, costos excesivos y limitaciones para la distribución del trabajo.

De igual forma, se reconoce la importancia que tiene la integración de la Ingeniería del dominio representada por el proceso InDoCaS, la Ingeniería de la Aplicación y los repositorios de las técnicas y artefactos, lo cual se considera como factor principal para lograr la interoperabilidad y así seguir el curso normal de sus funcionalidades con el propósito de generar activos de software para una línea de productos de software.

También, se reconoce la importancia de incorporar un modelo arquitectónico que defina la estructura para la construcción de un middleware para obtener la integración

e interoperabilidad de las aplicaciones, ya que según lo indicado por Gómez (2010) la arquitectura de software es la estructura de un sistema de software, que al construir debe satisfacer las necesidades actuales, también le facilita al software las capacidades necesarias para permitir su mantenimiento y evolución de acuerdo a las necesidades del negocio y requisitos del cliente.

El interés principal de este trabajo de investigación es dar respuesta a las siguientes interrogantes, ¿Cuál es el modelo arquitectural que define el Middleware de integración para las ingenierías del dominio representado por el proceso InDoCaS y de la aplicación?, ¿Cómo estructurar un componente de mediación que provea servicios de enrutamiento, transformación de mensajes y soportes para los diversos protocolos?, ¿Cómo es la estructura, formatos de los artefactos y técnicas involucradas en la integración de las ingenierías del dominio representado por el proceso InDoCaS y de la aplicación?, ¿Cuáles son los repositorios y la meta-data para implementar almacenar las estructuras establecidas en las técnicas y artefactos utilizados en las aplicaciones de las ingenierías del dominio representado por el proceso InDoCaS y de la aplicación?.

Objetivos

Objetivo General

Proponer un modelo arquitectural para una línea de producción que permita la integración de las ingenierías del dominio y de la aplicación usando al proceso InDoCaS como guía en el enfoque de calidad.

Objetivos Específicos

1. Estudiar el enfoque de líneas de producción de software.

2. Definir un Middleware que implemente tecnologías de integración para las ingenierías del dominio representado por el proceso InDoCaS y de la aplicación.

3. Estructurar un componente de mediación que provea servicios de enrutamiento, transformación de mensajes y soportes para los diversos protocolos.

4. Analizar la estructura y formatos de los artefactos y técnicas involucradas en la integración de las ingenierías del dominio representado por el proceso InDoCaS y de la aplicación.

5. Definir repositorios y su meta-data que permitan almacenar las estructuras establecidas en las técnicas y artefactos utilizados en las aplicaciones de las ingenierías del dominio representado por el proceso InDoCaS y de la aplicación.

Justificación e Importancia

En la actualidad, la sociedad y las organizaciones en general, requieren herramientas tecnológicas como sistemas de información que puedan crear y mantener la información de forma completa y consistente, que pueda intercambiar la información que se genera, independientemente de su estructura o formato.

Ante este hecho, la presente investigación basa su justificación en un modelo arquitectural que proporcionará un marco de referencia para guiar la construcción de

aplicaciones que incluyan características de calidad ajustadas a un estándar como: la integrabilidad y la interoperabilidad. Según Canelón (2010), la integrabilidad es la habilidad para hacer que componentes del sistema desarrolladas separadamente trabajen juntas correctamente; y la interoperabilidad es la habilidad de que el sistema, visto como un conjunto de componentes, pueda trabajar con otros sistemas.

El presente estudio facilitará una línea de productos de software integrada en sus fases fundamentales que permitirán a la industria de desarrollo del software reutilizar los activos de la línea para que los desarrolladores puedan manejar la variabilidad existente en los diversos dominios de aplicación y producir software de una manera más rápida, económica y de calidad. Así mismo, proporcionará un modelo de referencia a la academia, con el fin de difundir a la comunidad nuevas alternativas para el desarrollo de sus aplicaciones. En general para la sociedad y las organizaciones se incorporaran una infraestructura para desarrollar sistemas de información de calidad, con el fin de dar respuestas rápidas a solicitudes continuas de clientes y usuarios.

El modelo arquitectural propuesto debe integrar las ingenierías del dominio representado por InDoCaS y de la aplicación y los repositorios de artefactos y técnicas para obtener una línea de producción de software que permita manejar la variabilidad de un dominio.

La importancia de esta investigación radica en mejorar la comunicación entre las aplicaciones que constituyen las ingenierías del dominio representado por el proceso InDoCaS y de la aplicación, de esta manera puedan intercambiar información de los artefactos y técnicas, y seguir con el curso normal de sus funcionalidades de una manera eficiente y eficaz, con el fin de aportar activos a la líneas de producción de software para mejorar la gestión del desarrollo y la calidad del software que produce.

Alcances y Limitaciones

El alcance del presente estudio está orientado en primer lugar en definir un middleware de integración para la interoperabilidad de las aplicaciones de las ingenierías del dominio representada por el proceso InDoCaS como guía en el enfoque de calidad y la ingeniería la aplicación. Este middleware implementa tecnologías de integración de aplicaciones en un contexto orientado a servicios, incorporando un componente de mediación que provea servicios de enrutamiento, transformación de mensajes y soportes para los diversos protocolos. Asimismo, permitirá definir un repositorio y su meta-data, que permitirá implementar las estructuras, formatos de los artefactos y técnicas involucrados en la integración de las aplicaciones establecidas en las aplicaciones que constituyen las ingenierías del dominio representada por el proceso InDoCaS y de la aplicación en el presente trabajo.

CAPÍTULO II

MARCO TEÓRICO

Es necesario señalar el marco de referencia teórico que orienta el presente estudio, ya que éste determina la perspectiva de análisis y la visión del problema que se asume en esta investigación. Para ello, este capítulo se estructura de la siguiente manera: antecedentes de la investigación, las bases teóricas que fundamentan la investigación y la operalización de las variables.

Antecedentes

Los antecedentes permiten dar al lector toda la información posible acerca de las investigaciones que se han realizado sobre el problema que se investiga. En este sentido, es de vital importancia reflejar lo que se ha investigado en la temática de interés para el estudio, por cuanto da una proyección del cómo se ha manejado la integración de las aplicaciones.

Gómez (2011) en su trabajo titulado “Modelo arquitectural para aplicaciones móviles usando el enfoque de líneas de producción dinámica de software”, realizó la especialización del proceso para la ingeniería del dominio basado en calidad de software InDoCaS, agregó para ello actividades y artefactos a su disciplina análisis del dominio para ajustar el proceso a las líneas de producción dinámica de software. Finalmente, presentó una arquitectura para el dominio de Aprendizaje de idiomas asistido por móviles en inglés Mobile Assisted Language Learning (MALL), para lo

cual estudió algunos requisitos dinámicos y concluyó que es viable su uso en otros dominios o en otras investigaciones similares.

Destacó, además, que el activo de software denominado "Requisitos de referencia", con el cual se comparan los requisitos minimales del dominio y decidir si se trata de una línea habitual o dinámica, se puede extender a las necesidades propias de cada proyecto, incluso se pueden sugerir soluciones arquitecturales para cada requisito. Concluyo que la pericia de un arquitecto es de suma importancia en todo el proceso de ingeniería de dominio, lo que representa un inconveniente al momento de ejecutar el proceso InDoCaS, puesto que ciertas decisiones como la escogencia de los patrones y estilos arquitecturales dependen del punto de vista del arquitecto y su experiencia.

Este antecedente se considera importante porque da un gran aporte teórico a este trabajo de investigación; además se obtienen buenas prácticas para el diseño de un modelo arquitectural que es relevante para esta investigación.

Por su parte, Rivero (2011), en su trabajo titulado "Una línea de producción de software para sistemas transaccionales con una aplicación al proceso de desarrollo de software en la Coordinación Nacional de Tecnología de Información de la Universidad Nacional Experimental Politécnica "Antonio José de Sucre" (UNEXPO)", expandió el proceso general denominado InDoCaS agregando la disciplina de implementación del dominio; incorporó las actividades correspondientes a la implementación del dominio e instanció el método WATCH – Component para crear el proceso InDoCaS Extendido. Finalmente, utilizó el proceso InDoCaS Extendido se para llevar a cabo las actividades que componen la línea de producción de software para sistemas transaccionales utilizando como caso de estudio la Coordinación Nacional de Tecnología de información de la UNEXPO.

El trabajo anterior se toma como antecedente ya que aporta una amplia información sobre las actividades agregadas a la disciplina de implementación del

dominio del proceso InDoCaS y los artefactos que se generan como activos de software que formará parte de la línea de producción de software.

En el mismo orden de ideas, Canelón (2010) en su tesis doctoral "Un proceso para la ingeniería del dominio basado en calidad de software (InDoCaS)", desarrolló una especialización de las actividades de análisis y diseño del subproceso de la ingeniería del dominio del proceso de las líneas de producción de software, con el propósito de obtener una arquitectura base para una familia de productos de software. Hizo uso de la notación SPEM para especificar los aspectos dinámicos a través de diagramas de actividad donde se muestra el flujo de actividades y artefactos, en el aspecto estático se utilizó paquetes. Presentó un esquema para describir actividades y artefactos de cada disciplina. InDoCaS fue aplicado al dominio de aplicaciones móviles sensibles al contexto y al aprendizaje móvil como familia del dominio y se obtuvo la caracterización y arquitectura conceptual (baseline). Concluyó que el enfoque utilizado para la construcción de InDoCaS fue realizado desde las aplicaciones móviles sensibles al contexto y las familias de aplicaciones pertenecientes a este dominio.

El trabajo expuesto, aporta a esta investigación las actividades, artefactos de software y técnicas de las disciplinas de análisis y diseño del dominio, así como, elementos del proceso InDoCaS que deben contemplarse en el diseño del modelo de arquitectural.

Asimismo, Pernalete et al. (2010) en su artículo titulado, "IMS – Learning Design y el Modelo Arquitectural de AMBAR", presentaron un Framework para AMBAR (Sistema Generador de AMBientes de Enseñanza-ApRendizaje Constructivistas basados en Objetos de Aprendizaje (OA)) obteniendo una arquitectura inicial modular, escalable e interoperable; utilizaron las especificaciones IMS Learning Design (IMS LD) para apoyar la estandarización y reusabilidad de diseños de aprendizaje; también utilizaron tecnologías basadas en objetos distribuidos

representadas en una Arquitectura Orientada a Servicios (SOA), implementada sobre el Enterprise Server Bus (ESB) para la integración de aplicaciones. Concluyeron que la integración de la arquitectura orientada a servicios (SOA) sobre un EBS da origen a la propuesta arquitectural para AMBAR asegurando la interoperabilidad, accesibilidad y escalabilidad de sus aplicaciones. También resaltaron que los componentes son integrables en tiempo de ejecución lo que da la flexibilidad de añadir funcionalidad a un sistema en producción sin tener que pararlo, y además por su naturaleza desacoplada sin comprometer el funcionamiento de otros componentes del sistema.

La investigación referida, ofrece un cúmulo de información pertinente para este estudio, por cuanto describe de manera general las tecnologías usadas, lo cual servirá de apoyo para la realización de la propuesta del presente trabajo. Además, es importante conocerlos principales inconvenientes que se presentaron en el diseño de la arquitectura para AMBAR, a fin de que el modelo arquitectural pueda responder eficazmente a los problemas de integración de aplicaciones y adecuarse lo más posible a los requisitos.

Silveira y Pastor (2009) en su artículo titulado: “Servicios de Integración de Sistemas de Información Empresariales: Rol e Importancia de los Procesos de Negocio”, definieron un método para proyectos de integración de Sistemas de Información Empresariales a partir de la investigación previa en Integración de Aplicaciones Empresariales o EIA, el análisis de errores y éxitos en casos reales y la experiencia en proyectos de integración del primer autor, identificaron los principales procesos que intervienen en el modelo propuesto. Concluyeron que para lograr una integración eficiente, una buena práctica es el uso de un método que se adecue a la naturaleza de los proyectos de integración.

Sin dudas que el estudio citado es relevante para la investigación en curso, debido a que plasma ideas de buenas prácticas en integración de sistemas de información empresarial, que son elementales para el diseño del modelo arquitectural.

Caponi et al. (2008), En su trabajo titulado: “Mensajería en sistemas de información”, estudiaron el uso de patrones de mensajería y evaluaron las alternativas para la implementación de los patrones en un contexto orientado a servicios, mostraron las herramientas Web Services (WS) como mecanismo de interacción de las aplicaciones a integrar, su familia de estándares WS y Bus de Servicio Empresariales (ESB) como plataforma base. Desarrollaron un caso de estudio en donde integraron componentes heterogéneos y aplicaron patrones de mensajería, analizaron la implementación de estos en base a las herramientas y tecnologías antes mencionadas; clasificaron los patrones de mensajería basada en las implementaciones propuestas; concluyeron que las herramientas elegidas eran adecuadas para el contexto de trabajo y que con ellas se logra resolver la mayoría los patrones.

El estudio presentado, constituye un aporte significativo para esta investigación, por cuanto sirve como guía para la fase de diseño de la propuesta, ya que aporta información referente a las herramientas y tecnologías como Web Services (WS) y Bus de Servicio Empresariales (ESB) que hacen posible integración de aplicaciones y el desarrollo un caso de estudio en donde integraron componentes heterogéneos y aplicaron patrones de mensajería, analizaron la implementación de estos en base a las tecnologías de integración .

Por último, es conveniente explicar, que las investigaciones presentadas tienen relación con el objetivo general de este trabajo de grado y son fundamentales en la realización de esta investigación.

En el cuadro 1 se resumen los antecedentes planteados en esta sección, se describen los objetivos, aportes y limitaciones de cada antecedente asociado con cada variable del estudio.

Cuadro 1. Resumen de los antecedentes

ANTECEDENTES	APORTE	LIMITACIONES
Rivero (2011). Una línea de producción de software para sistemas transaccionales con una aplicación al proceso de desarrollo de software en la Coordinación Nacional de Tecnología de Información de la Universidad Nacional Experimental Politécnica "Antonio José de Sucre" (UNEXPO). Trabajo de Grado. DCyT. UCLA.	Aporta una amplia información sobre las actividades agregadas a la disciplina de implementación del dominio del proceso InDoCaS y los artefactos que se generan como activos de software que formará parte de la línea de producción de software.	Se enfoca únicamente en los sistemas transaccionales. No está desarrollado el proceso de Ingeniería de la Aplicación de la línea de producción de software.
Gómez (2011) Modelo arquitectural para aplicaciones móviles usando el enfoque de líneas de producción dinámica de software. Caso de estudio: Aprendizaje de idiomas asistido por móviles en inglés Mobile Assisted Language Learning (MALL). Trabajo de Grado. DCyT. UCLA	Realizo la especialización al proceso InDoCaS, agregó para ello actividades y artefactos a su disciplina análisis del dominio. Finalmente, presentó una arquitectura para el dominio de MALL.	Se enfoca únicamente en el dominio Aprendizaje de idiomas asistido por móviles en inglés Mobile Assisted Language Learning (MALL).
Canelón R. (2010). Un proceso para la ingeniería del dominio basado en calidad de software. Caso de estudio: Una aplicación al dominio del aprendizaje móvil sensible al contexto. Tesis Doctoral. Universidad Central de Venezuela.	Aporta a esta investigación las actividades, artefactos de software y técnicas de las disciplinas de análisis y diseño del dominio, así como, elementos del proceso InDoCaS que deben contemplarse en el diseño del modelo de arquitectural.	No está desarrollada la disciplina de Implementación del dominio del proceso de Ingeniería de Dominio.
Pernalete et al. (2010) IMS – Learning Design y el Modelo Arquitectural de AMBAR (Sistema Generador de AMBientes de Enseñanza-ApRendizaje Constructivistas basados	Presentaron un Framework para AMBAR obteniendo una arquitectura inicial modular, escalable, interoperable a través de tecnologías de integración	No se muestra en descripción de los componentes del middleware y los patrones de diseño.

en Objetos de Aprendizaje (OA)).	como ESB, SOA, Servicios web.	
Caponi et al. (2008), Mensajería en sistemas de información. Facultad de Ingeniería. Universidad Montevideo-Uruguay	Desarrollaron un caso de estudio en donde integraron componentes heterogéneos y aplicaron patrones de mensajería, analizaron la implementación de estos en base a las tecnologías de integración.	No está orientado a un enfoque de línea de producción de software

Fuente: Autor de la investigación.

Bases Teóricas

Las bases teóricas de una investigación según Ramírez (ob.cit.), ilustran al lector sobre diferentes teorías que se han elaborado para interpretar el objeto de estudio y sus relaciones con otros fenómenos de la realidad. Comprenden un conjunto de conceptos y proposiciones que constituyen un punto de vista o enfoque determinados, dirigido a explicar el fenómeno o problema planteado. En este caso pretenden describir: Arquitectura de Software, Estilos Arquitecturales, Patrones Arquitecturales, Patrones de Diseño, Líneas de Producción de Software (LPS), Ingeniería del Dominio, Ingeniería de la Aplicación, Gestión de Variabilidad, InDoCaS, Integración de Aplicaciones, Middleware, Plataformas Middleware, Servicios Web, Arquitectura Orientada a Servicios (SOA), Bus de Integración de Aplicaciones (ESB)., Repositorios de Datos.

Arquitectura de Software

Gómez (2010), afirma que la arquitectura representa la base de un sistema de software que debe ser construida pensando en satisfacer tanto las necesidades actuales, como en proporcionar al software las capacidades necesarias para permitir

su mantenimiento y evolución de acuerdo a las necesidades del negocio y las solicitudes de los clientes.

A medida que los sistemas de información son cada vez más complejos y amplios, crece en importancia y dificultad en la organización del trabajo y la estructura de sus componentes. Por lo tanto, la arquitectura de software de las empresas debe ser: simple (para todas las partes interesadas pueden entender y utilizar), flexible (para que pueda adaptarse a los cambios dinámicos en el tiempo requerido por el entorno empresarial), reutilizable (especialmente en los bloques de software) y ser capaz de desvincular la funcionalidad de negocio de las tecnologías utilizadas para su ejecución.

La arquitectura software trata el diseño e implementación de la estructura de alto nivel del software. Es el resultado de ensamblar un cierto número de elementos arquitectónicos para satisfacer la funcionalidad y ejecución de los requisitos del sistema; así como los requisitos no funcionales tales como: fiabilidad, escalabilidad, portabilidad, disponibilidad, entre otros. De acuerdo al autor, no es fácil de obtener una arquitectura de software en un sólo modelo (o diagrama), sin embargo, para describir una arquitectura software se tiene el modelo 4+1 compuesto de múltiples vistas o perspectivas.

El modelo 4+1 Vistas

El Modelo 4+1 Vistas organiza la descripción de la arquitectura de un software usando cinco (5) vistas concurrentes, cada una de las cuales está dirigida a un conjunto específico de conceptos. Los arquitectos exponen sus decisiones de diseño en cuatro (4) vistas y usan la quinta vista para ilustrar y validar dichas decisiones. Véase figura 2.

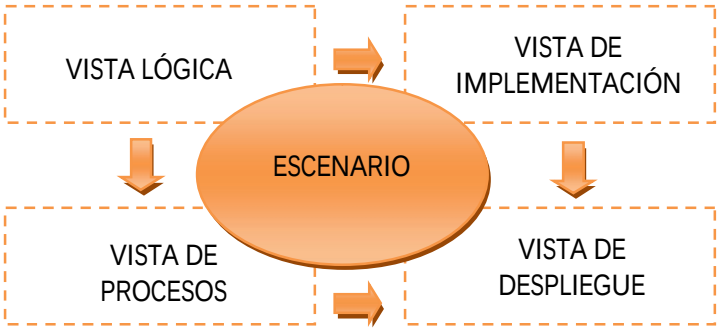


Figura 2. *Modelo 4+1 (Kruchten 1995).*

Dichas vistas son definidas por el autor como:

Arquitectura Lógica: soporta el análisis y la especificación de los requisitos funcionales: lo que el sistema debería proporcionar en términos de servicios a sus usuarios. El sistema se descompone en un conjunto de abstracciones clave tomadas mayormente del dominio del problema, en forma de objetos o clases.

Arquitectura de Procesos: se tratan algunos requisitos no funcionales (Ejecución, disponibilidad, tolerancia a fallos, integridad, entre otros). Esta vista también especifica que hilo de control ejecuta cada operación identificada en cada clase identificada en la vista lógica; se centra por tanto en la concurrencia y distribución de procesos.

Arquitectura de Desarrollo: la vista de desarrollo o despliegue se enfoca en la organización de los módulos software en el entorno de desarrollo. El software es empaquetado en pequeños trozos (librerías de programa, subsistemas, componentes,

entre otros), los subsistemas se organizan en capas jerárquicas, y cada capa proporciona una interfaz bien definida a sus capas superiores.

Arquitectura Física: la vista física se centra en los requisitos no funcionales, tales como la disponibilidad del sistema, la fiabilidad (tolerancia a fallos), ejecución y escalabilidad.

Escenarios: los diseñadores de software organizan la descripción de sus decisiones arquitecturales alrededor de estas cuatro (4) vistas, y las ilustran con una pequeña selección de casos de uso o escenarios, constituyendo así la quinta vista. La arquitectura está parcialmente producida por esos escenarios.

Estilo arquitectural

El estilo arquitectónico es un patrón de construcción. Los estilos conjugan componentes, conectores, configuraciones y restricciones (constraints). La descripción de un estilo se puede formular en lenguaje natural o en diagramas, pero lo mejor es hacerlo en un Lenguaje de Descripción Arquitectónica (ADL) o en Lenguajes Formales de Especificación (LFE).

Patrón de Software

Los patrones facilitan la reutilización del diseño y de la arquitectura, capturando las estructuras estáticas y dinámicas de colaboración de soluciones exitosas a problemas que surgen al construir aplicaciones. Si se hace una primera clasificación de los patrones software atendiendo al nivel de abstracción de éstos, pudieran quedar: los patrones arquitectónicos, de diseño o centrados en el código, entre otros.

Patrón Arquitectónico

Un patrón de arquitectura es una descripción de elementos y los tipos de relación, junto con un grupo de restricciones en cómo deben ser usados. Los patrones de arquitectura se pueden ver como la descripción de un problema en particular y recurrente de diseño, que aparece en contextos de diseño arquitectónicos específicos, y representa un esquema genérico demostrado con éxito para su solución. El esquema de solución se especifica mediante la descripción de los componentes que la constituyen, sus responsabilidades y desarrollos, así como también la forma en que estos colaboran entre sí.

Los patrones arquitectónicos se definen sobre aspectos fundamentales de la estructura del sistema software, donde se especifican un conjunto de subsistemas con sus responsabilidades y una serie de recomendaciones para organizar los distintos componentes. Posteriormente, el diseño hará uso de patrones de diseño para resolver problemas específicos.

Patrón de Diseño

Un patrón de diseño provee un esquema para refinar los subsistemas o componentes de un sistema de software, o las relaciones entre ellos. Describe la estructura comúnmente recurrente de los componentes en comunicación, que resuelve un problema general de diseño en un contexto particular.

Son menores en la escala de abstracción que los patrones arquitectónicos, además tienden a ser independientes de los lenguajes y paradigmas de programación. Su aplicación tiene efectos sobre subsistemas, debido a que especifica a un mayor nivel de detalle, sin llegar a la implementación, el comportamiento de los componentes del subsistema.

En conclusión, una arquitectura de software es la estructura general de un sistema que comprenden elementos de software, las propiedades visibles de dichos elementos y las relaciones entre los mismos. Estos elementos son componentes del sistema que pueden implementar patrones dar solución a un problema. La arquitectura de software es la base para el entendimiento, consenso, negociación, comunicación; además puede ser un activo de software para la línea de productos de software.

Líneas de Producción de software (LPS)

Líneas de productos de software representa las técnicas de ingeniería para constituir de un portafolio de sistemas de software relacionados a un conjunto de activos de software con el uso de un medio común de producción (Montilva, 2007).

El alcance de la línea de productos, está determinado por el conjunto de productos de software que pueden ser producidos a partir de los activos de software y del modelo de decisión, Véase figura 3. (Clements and Northrop, 2002)

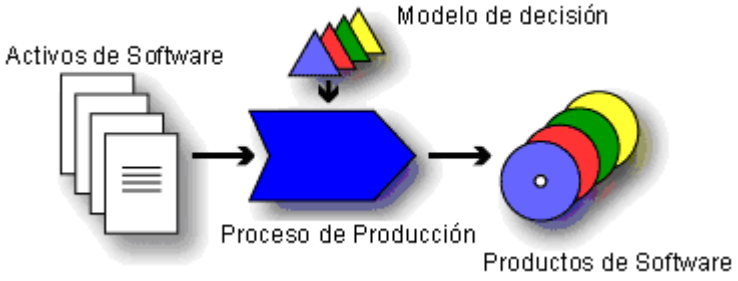


Figura 3. *Modelo básico de una línea de productos de software (Clements and Northrop 2002).*

Activos de software: colección de activos o partes de software (requisitos, componentes de código fuente, casos de prueba, arquitectura y documentación) que

puede ser configurado y compuesto de diferentes maneras para producir todos los productos de la línea.

Modelo de decisión: describe aspectos opcionales y características variables de los productos de la línea. Cada producto de la línea está definido por decisiones de producto.

Proceso de producción: establece los mecanismos para la composición y configuración de los productos procedentes de las entradas de activos de software. Las decisiones de producto se utilizan durante el proceso de producción para establecer qué insumos de activos de software a utilizar y la forma de configurar los puntos de variación dentro de esos activos.

Productos de Software: conjunto de todos los productos que pueden ser producidos para la línea de productos.

Según Montilva (2006), los beneficios tácticos y estratégicos de la línea de producción son:

- Reducción en el tiempo promedio de creación y entrega de nuevos productos.
- Reducción en el número promedio de defectos por producto.
- Reducción en el esfuerzo promedio requerido para desarrollar y mantener los productos.
- Reducción en el costo promedio de producción de los productos.
- Incremento en el número total de productos que pueden ser efectivamente desplegados y mantenidos.
- Reducción en el tiempo de entrega (time-to-market) y el tiempo de retorno (time-to-revenue) de nuevos productos.
- Mejoras en el valor competitivo del producto.
- Márgenes mayores de ganancias.

- Mejor calidad de los productos.
- Mejoras en la reputación de la empresa.
- Mayor escalabilidad del modelo de negocios en términos de productos y mercados.
- Mayor agilidad para expandir el negocio a nuevos mercados.
- Reducción de riesgos en la entrega de productos.

Para esta investigación se detallaran las dos principales actividades que deben ser desarrolladas en la construcción de una línea de producto, estas son la ingeniería del dominio y la ingeniería de la aplicación, véase figura 4.

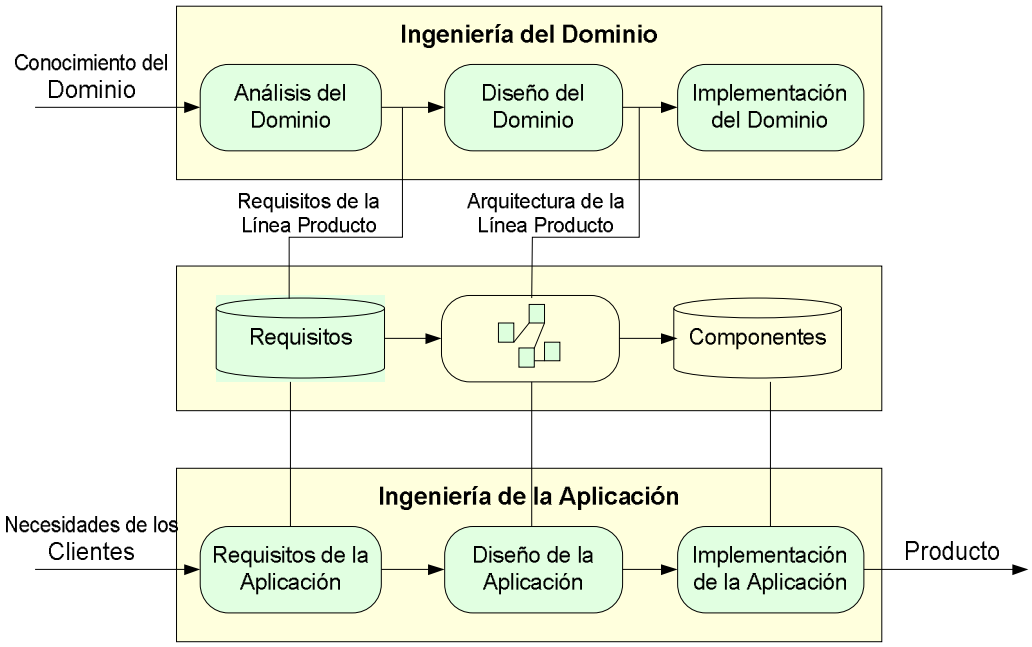


***Figura 4**. Proceso General para las Líneas de Productos. (Montilva, 2007).*

Ingeniería de dominio

De acuerdo a Montilva (2007), la ingeniería de dominio obtiene información y representa el conocimiento sobre un dominio determinado, con el propósito de establecer activos de software que sean reutilizables en el desarrollo de los nuevos productos de una LPS. Las actividades principales de la ingeniería de dominio son: análisis de aspectos para determinar los requisitos que son comunes, opcionales y diferentes a todos sus miembros.

Diseño de la arquitectura para producir una arquitectura de dominio la cual tiene componentes comunes a todos los miembros de la familia, componentes opcionales que son requeridos por algunos miembros y componentes variantes de los cuales algunos miembros de la familia emplean distintas versiones, las cuales tienen puntos de variación que permiten configurarlos.

Implementación del dominio para crear y almacenar los activos de software que se emplearán para producir los productos de software.

La ingeniería del dominio, en opinión de Canelón (2010), permite identificar similitudes y diferencias entre miembros de una familia de productos e implementa un conjunto de artefactos de software (componentes) que son compartidos entre los productos. El proceso consiste de actividades para analizar sistemas en un dominio y la creación de una arquitectura referencial y componentes reutilizables basados en el análisis. La arquitectura referencial y los componentes reutilizables son amoldados a las similitudes y diferencias de los sistemas en el dominio.

En conclusión, la ingeniería de dominio se dirige a la creación sistemática de modelos de dominios, arquitecturas, componentes y artefactos de software que son reutilizados por la ingeniería de aplicaciones.

Ingeniería de la aplicación

Según Montilva (2007), la ingeniería de aplicaciones tiene como objetivo desarrollar productos de la LPS haciendo uso de los activos de software y planes de producción. La ingeniería de la aplicación es la parte de la ingeniería de software que se basa en la reutilización de componentes existentes y en el conocimiento del dominio. Las aplicaciones son construidas mediante el ensamblaje de componentes.

Las actividades del proceso de la ingeniería de la aplicación incluye la utilización de un modelo del dominio para identificar los requisitos del cliente, un diseño particular para especificar una configuración del producto y aplicaciones generadoras y componentes de software para producir código de la aplicación.

Gestión de Variabilidad

Según Gómez (2011), Uno de los elementos clave para una línea de producción de software es la representación y gestión de la variabilidad. La variabilidad ha sido descrita como la habilidad de un sistema software o artefacto para ser cambiado, personalizado o configurado para usarse en un contexto particular. La ingeniería de la línea de producción de software se pretende soportar un grupo de productos. Esos productos pueden atender diferentes clientes individuales o diferentes segmentos de mercado. La variabilidad es un concepto clave en cualquiera de esas aproximaciones.

En lugar de entender cada uno de los sistemas individuales, se observa a la línea de producto como un todo y la variación entre los sistemas individuales. Esta variabilidad debe ser definida, representada, explotada e implementada, a través de la del proceso de desarrollo de la línea.

Cuando se gestiona la variabilidad en una línea, se necesitan distinguir tres tipos de características (*feature*) principales:

Comunes: una característica (funcional o no funcional) puede ser común a todos los productos en la línea de producto. Esta es implementada como parte de la plataforma.

Variables: una característica puede ser común a algunos productos, pero no a todos. Entonces debe ser explícitamente modelada como una posible variabilidad y debe ser implementada en una forma que permita tenerla en solamente los productos seleccionados.

Específicos del producto: una característica puede ser parte de solamente un producto. Tales especialidades son escasamente requeridas por el mercado, pero si por los intereses de los clientes individuales.

Mientras las características comunes y variables se manejan esencialmente en la ingeniería del dominio, las características específicas del producto son manejadas exclusivamente en la ingeniería de la aplicación.

FODA: Análisis del dominio Orientado a Rasgos

Algunos métodos, tales como FODA (análisis del dominio Orientado-Rasgo). El método de análisis FODA desarrollado por el (SEICMU, 1990) es uno de los principales métodos de partida en el terreno del desarrollo de Líneas de Producción. Examinando los diferentes sistemas de software relacionados, el análisis de dominio permite obtener una descripción genérica de los requisitos y una forma de implementarlos. El resultado de aplicar FODA es una familia de productos más que un producto individual, produciendo un modelo del dominio con la flexibilidad suficiente para reflejar las diferentes posibilidades que existan, y una arquitectura estándar para desarrollar componentes software. Las fuentes del análisis del dominio son, en FODA, la documentación publicada sobre el dominio, los estándares, las aplicaciones existentes y los expertos. FODA organiza las capacidades (lo que observa el usuario final) de la familia de productos constituyendo jerarquías en forma de árbol. Las capacidades o características comunes se colocan en los nodos raíz del

árbol. Cada variante sobre estas características comunes implica una rama en el árbol. La ventaja de esta técnica es que se recogen en un único modelo todas las posibles variaciones que hay en la familia de productos y que se identifican claramente.

InDoCaS: Un proceso para la Ingeniería del Dominio basado en Calidad de Software

InDoCaS es un proceso de ingeniería de dominio basado en la calidad de software cuyo objetivo final "es obtener una arquitectura base para una familia de productos, las disciplinas de este proceso son análisis y diseño del dominio" (Canelón, 2010; p. 65). El análisis utiliza un modelo de clasificación de requisitos (RECLAMO), ISO/IEC 25010 y FODA. El diseño está basado en un modelo general de diseño arquitectural que concreta síntesis y evaluación arquitectural.

Según (Canelón, ob.cit.), InDoCaS incorpora subprocesos en las fases de análisis y diseño del dominio dentro de la disciplina ingeniería de dominio. La fase de análisis es fundamental para el desarrollo de las líneas de productos de software porque se definen los aspectos comunes a todas las familias de dominio y los particulares de cada una de ellas. Este subproceso permite la caracterización del dominio, identifica las propiedades de calidad que deben ser garantizadas y define el modelo de calidad asociado al dominio que brinda soporte al resto de las fases. Además, proporciona una base o modelo de arquitectura que es individualizada cuando se considera hacer una familia del dominio. El subproceso de diseño arquitectural es llevado a cabo a través de actividades fundamentales para la síntesis y evaluación arquitectural de una familia del dominio.

Los esquemas que definen las actividades y artefactos que están dentro de las disciplinas de análisis y diseño del dominio son:

El esquema de la actividades contiene: nombre de la actividad, responsable, objetivo específico, número de identificación, tipo de documento generado (lista,

esquema, tabla, entre otros), técnicas utilizadas, artefactos de entrada y artefactos de salida.

El esquema de los artefactos contiene: nombre de la actividad, constructor, número de identificación de la actividad, número de identificación del artefacto, formato asociado (lista, esquema, modelo, diagrama, tabla, entre otros).

En conclusión, las ingenierías del dominio y de la aplicación de intercambiar información tales como artefactos de software y técnicas a través de la integración de sus aplicaciones.

Integración de Aplicaciones

Según Canelón (ob.cit.) la ingeniería de aplicaciones facilita una plantilla para una integración de un conjunto de tecnologías que juntas proporcionan integración en tiempo real. Esta plantilla incluye: integración de plataformas, integración de eventos a través de mensajes, transformación y enrutamiento a través de reglas, integración de interfaces, aplicaciones personalizadas.

La integración de aplicaciones empresariales, se refiere a los desarrollos de software orientados a crear medios para compartir datos y procesos entre distintas aplicaciones de una organización. Originalmente los programas de las empresas solían ser independientes, de tal forma que se tenía un programa para recursos humanos, otra para llevar el inventario, la contabilidad, las ventas, entre otros, sin interactuar entre sí. Por lo general eran desarrollados para resolver necesidades específicas, con la tecnología que en cada momento existiese y haciendo uso, casi siempre, de sistemas y tecnologías propietarias. Conforme las empresas crecieron, reconocieron la necesidad de que sus diferentes sistemas pudiesen compartir información e interactuar.

En conclusión, La integración de aplicaciones empresariales constituye un proceso

mediante el cual hardware, software y procedimientos de negocios se disponen para hacer posible el manejo de la información y los sistemas en un trabajo conjunto que pueda alcanzar la sinergia, todo esto se hace a través de un Middleware.

Middleware

Un middleware es el software que reside entre la aplicación y los sistemas operativos, protocolos y hardware subyacente, con el objetivo de permitir que componentes heterogéneos y distribuidos se interconecten y colaboren entre sí.

Según García et al. (2007), definen las plataformas Middleware como "una solución muy utilizada para el desarrollo de proyectos de software distribuido, que implican la integración de sistemas heterogéneos". (p.01).

Conforme a Mondejar (2010), un Middleware se sitúa sobre el nivel de red y debajo del nivel de aplicación y resumen la heterogeneidad y complejidad del entorno fundamental. Además proporciona un entorno integrado distribuido cuyo objetivo es simplificar la tarea de poner en práctica los sistemas distribuidos, y también para proporcionar valor añadido los servicios tales como ubicación o persistencia para habilitar desarrollo de aplicaciones distribuidas.

En conclusión, un middleware requiere de plataformas tecnológicas para dar solución al problema de interoperabilidad.

Plataforma Middleware

La arquitectura Middleware proporciona una plataforma escalable de despliegue donde los aspectos distribuidos han sido introducidos en la red y activados en máquinas individuales o en grupos de máquinas (Mondejar, ob.cit.).

La implementación a través de herramientas comerciales de integración como son los sistemas distribuidos middleware, proporciona un software que facilita y

simplifica la construcción de aplicaciones/servicios distribuidos, proporcionando mecanismos que los componentes distribuidos utilizan para comunicarse e interaccionar a través de la red.

El middleware es una capa situada entre la aplicación y la capa de transporte que independiza las distintas aplicaciones software de los PC´s (clientes y servidores) de la plataforma de comunicaciones subyacente. El programador puede proceder sin tener en cuenta la tecnología concreta de comunicaciones empleada, ni el lenguaje de programación utilizado en la capa de aplicación.

Ventajas de un middleware

Algunas de ventajas de un middleware son: los códigos del cliente y servidor están aislados de las capas de transporte y protocolos de comunicación inferiores, la gestión de conexión es transparente al usuario, las aplicaciones servidoras son transportables, es decir, pueden migrar sin cambios de una máquina a otra.

El middleware de la presente investigación estará basado en un Bus de Integración de Aplicaciones (ESB, Enterprise Service Bus), con la implementación de una Arquitectura Orientada a Servicios (SOA, Service Oriented Architecture).

Arquitectura Orientada a Servicios

Arquitectura Orientada a Servicios (SOA, Service Oriented Architecture) la que permite a distintos sistemas, escritos en diferentes lenguajes de programación y de distintas plataformas tecnológicas, transformarse en servicios débilmente acoplados y altamente interoperables.

Según Pernalete et al. (2010), definen SOA como una arquitectura que soporta servicios débilmente acoplados para posibilitar la flexibilidad en el negocio de una

manera interoperable e independiente de la tecnología. Consta de un conjunto de servicios de negocio que soportan la realización de procesos de negocio de principio a fin de una forma dinámica y reconfigurable utilizando descripciones de servicios basadas en interfaces. Utilizando SOA se puede descomponer la funcionalidad de la organización en partes reutilizables, más manejables, que pueden ser diseñadas, desarrolladas y gestionadas de forma independiente, como servicios.

SOA permite una implementación ligera y adaptable, que aprovecha la infraestructura existente, expandiéndola en lugar de sustituirla, basada en eficiencia a un bajo costo, con bajas tasas de tráfico y mecanismos estandarizados de seguridad soportados en servicios.

SOA es la nueva estrategia para construir sistemas de tecnologías de información que permite a los negocios la interoperabilidad entre los mismos a pesar de los cambios constantes entre sus negocios, como nuevos departamentos, nuevas reglas de negocios, nuevas legislaciones, nuevas plataformas, entre otros cambios.

Bus de Integración de Aplicaciones

Caponi et al. (2008) definieron Bus de Integración de Aplicaciones o Enterprise Service Bus (ESB) como "una plataforma de integración de aplicaciones basada en estándares, que combina: mensajera, servicios, ruteo y transformación de mensajes, con el objetivo de coordinar y conectar de manera confiable un número significativo de aplicaciones" (p. 28).

Esta construido sobre un canal común de mensajes (bus) altamente distribuibles, multiprotocolo, basado en estándares y provee la columna vertebral para implementar una Arquitectura Orientada a Servicios (SOA[7]). Véase figura 5, se ilustra la infraestructura de este tipo de plataformas.

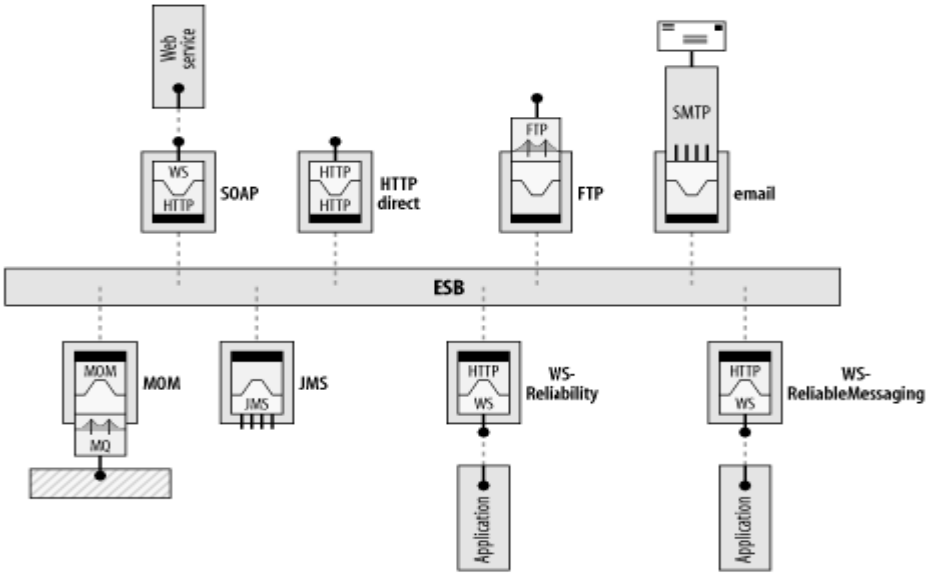


Figura 5. *Capacidades de integración multiprotocolo de un ESB (Caponi et al., 2008).*

Por su parte Garimella et al. (2008) indicaron que un ESB es un "elemento de la arquitectura de software que proporciona servicios fundamentales para los sistemas de información a través de un motor de mensajería controlado por eventos." (p. 71). También, es un bus de servicios corporativos y es parte de la categoría de infraestructura de Middleware.

Esta plataforma trae ventajas significativas en contextos en los que existan necesidades de integrar aplicaciones. Estas ventajas se ven con mayor claridad en organizaciones que presentan ambientes heterogéneos, con muchas aplicaciones ejecutando en plataformas diferentes, al tener una necesidad de integración, los ESBs resuelven gran parte del proceso permitiendo tener un buen control del mismo, según lo indicado por Caponi et al. (2008).

Según el mismo autor las funcionalidades básicas que un ESB son:

- Transparencia de localización: desacoplan a los servicios proveedores de los consumidores. Estos proveen una plataforma central para comunicar aplicaciones haciendo transparente al consumidor la localización del receptor.
- Conversión de protocolos de transporte. Un ESB es capaz de integrar aplicaciones por más que estas se comuniquen bajo diferentes protocolos de transporte (HTTP, SMTP, FTP, entre otros).
- Transformación de mensajes: proveen funcionalidades para transformar formatos de mensajes. Por lo general soportan transformaciones basadas en estándares como XSLT y XPath.
- Ruteo de mensajes. Determinar el destinatario de un mensaje, es una de las funcionalidades más importantes de los ESBs.
- Seguridad: proveen funcionalidades de autenticación, autorización y encriptación de mensajes.
- Monitoreo y administración: proveen ambientes de monitoreo y administración necesarios para el control de los flujos de mensajes.
- Soporte a ejecución de procesos de negocio. Permiten la especificación y ejecución de procesos de negocio que orquestan servicios accesibles a través del bus.
- Manejo de transacciones: proveen soporte transaccional en las interacciones con el bus de servicios, lo que permite que se pueda asegurar el envío o recepción de varios mensajes a través de este de forma atómica.

El ESB está integrado por aplicaciones proveedoras de servicios. Cada una de las aplicaciones puede ser independiente de las demás. Esta capacidad de integración es independiente del modo de licencia del software, libre o propietario. Esta arquitectura nos permite, por ejemplo, utilizar con una herramienta de software propietario unos servicios desarrollados sobre plataformas de software libre.

La presente investigación está enfocada en obtener un middleware basado en un ESB que comunique a las aplicaciones, los servicios web y repositorios de datos disponibles.

Repositorio

Un repositorio es un sitio centralizado donde se almacena y mantiene información digital, habitualmente bases de datos o archivos informáticos. Pueden ser de acceso público o estar protegidos y necesitar de una autentificación previa. Los depósitos más conocidos son los de carácter académico e institucional. La mayoría de los repositorios cuentan con sistemas de respaldo y mantenimiento preventivo y correctivo.

Según López (2007), un repositorio digital tiene las siguientes características: contiene objetos digitales, contiene metadatos, asegura la identificación persistente del objeto mediante un identificador único persistente, ofrece funciones de gestión, archivo y preservación de los objetos, proporciona un acceso fácil, controlado y estandarizado a los objetos, ofrece los sistemas adecuados de seguridad para los objetos y los metadatos, es sostenible en el tiempo, entre otros.

Finalmente, las bases teóricas brindaron a la presente investigación la oportunidad de comprender, familiarizarse y analizar las características de fondo relacionadas con Arquitectura de Software, Líneas de Producción de Software (LPS), Integración de Aplicaciones, Middleware, Plataformas Middleware, Repositorios de Datos, entre otros.

Operacionalización de las Variables

Cuadro 2. La operacionalización de las variables.

Proponer un modelo arquitectural para una línea de producción que permita la integración de las ingenierías del dominio y de la aplicación usando al proceso InDoCaS como guía en el enfoque de calidad.				
Variable	**Definición**		**Dimensiones**	**Indicadores**
	Conceptual	**Operacional**		
Modelo arquitectural para una línea de producción de software	Se define un Middleware que provee un marco de referencia para guiar la construcción de aplicaciones	La variable "modelo arquitectural para la integración de las aplicaciones" considerando dos dimensiones, la dimensión Arquitectura de Software y la dimensión Línea de producción dinámica de software	Arquitectura de Software.	Estilos Arquitecturales Patrones Arquitecturales Repositorio
			Línea de producción de software	Ingeniería de Dominio. InDoCaS Ingeniería de la aplicación.

Fuente: Autor de la investigación 2012

CAPÍTULO III

MARCO METODOLÓGICO

Naturaleza del estudio

De acuerdo con los objetivos propuestos el presente trabajo de investigación se enmarca en el paradigma positivista y bajo el enfoque cuantitativo, orientado hacia una investigación de campo bajo un diseño de Proyecto Factible.

En lo que al paradigma positivista se refiere, Hurtado (2008), señala que el paradigma positivista está marcado por un estilo de pensamiento sensorial, una orientación concreta-objetiva hacia las cosas, por un lenguaje impersonal, matemático, una vía hipotética deductiva del conocimiento y por unas referencias de validación situadas en la realidad objetiva.

En cuanto al enfoque cuantitativo, Hurtado (2008), destacan que en este enfoque predomina el uso de instrumentos de medición que proporcionan datos, los cuales requieren la aplicación de modelos matemáticos y de estadística para su interpretación.

Por su naturaleza, el presente estudio constituye una investigación de campo y se corresponde, tal como se establece en el Manual de Trabajo de Grado de Especialización, Maestría y Tesis Doctorales de la Universidad Pedagógica Experimental Libertador (2007), en un "análisis sistemático de problemas en la realidad, con el propósito de describirlos, interpretarlos, entender su naturaleza y

factores constituyentes…" (p.18). Atendiendo al diseño, se sustenta en el Proyecto Factible, que tal como lo señala dicho manual:

> Consiste en la investigación, elaboración y desarrollo de un modelo operativo viable para solucionar problemas, requisitos o necesidades de organizaciones o grupos sociales; puede referirse a la formulación de políticas, programas, tecnologías, métodos o procesos. El proyecto debe tener apoyo en una investigación documental, de campo o un diseño que incluya amabas modalidades. (p. 21).

De esta manera, tanto el paradigma positivista, como el tipo y diseño de investigación adoptado para el estudio, son pertinentes para dar respuestas a las interrogantes de investigación y al propósito implícito en el presente estudio, centrado en proponer un modelo arquitectural para la integración de las aplicaciones de la ingeniería del dominio representada por el proceso InDoCaS y la ingeniería de la aplicación, que permita la integración e interoperabilidad mediante tecnologías basadas en servicios y bus de integración.

El estudio se llevará a efecto por medio de tres fases: Diagnóstico, Factibilidad y Diseño, las cuales se describen a continuación:

Fases del Estudio

Fase I. Diagnóstica

Esta fase consiste en el desarrollo del estudio de campo que sustentará el proyecto de investigación, para lo cual se espera establecer los elementos que describan las necesidades reales del proyecto en curso. Dicha fase sienta sus bases en los siguientes aspectos:

Población

Según Hernández, Fernández y Baptista (2007), definen "una población como el conjunto de elementos que presentan una característica común" (p.240), por tanto es la totalidad de sujetos a estudiar que posean caracteres en común. En razón a ello, la población de estudio estará conformada por todas las pequeñas y medianas empresas, cooperativas, universidades, instituciones y comunidades pertenecientes al área de Informática, que son quienes realizan desarrollo de software, establecidas específicamente en la ciudad de Barquisimeto, Estado Lara.

Muestra

Hurtado (2008), expresa que la muestra es una porción de la población que se toma para realizar el estudio, la cual se considera representativa de la población, además, que los procedimientos de muestreo no son requeridos en toda investigación, lo importante es la representatividad de la muestra para que los hallazgos puedan ser generalizados, para lo cual el investigador puede tomar como criterio, entre otros, la posibilidad de tener acceso a la población, disponibilidad de tiempo, personal y de recursos. Por tanto, una muestra debe ser definida en base de la población determinada, y las conclusiones que se obtengan de dicha muestra sólo podrán referirse a la población en referencia.

La muestra estará representada por un total de veinte (20) jefes de proyectos de Software, que laboran en seis (6) organizaciones que corresponde a pequeñas y medianas empresas, dedicadas al desarrollo de software, establecidas en la ciudad de Barquisimeto, en donde han mostrado disposición para participar en el estudio. Estas organizaciones se listan en el siguiente cuadro:

Cuadro 3. Descripción de la Muestra de Estudio

NOMBRE	JEFES DE PROYECTOS DE SOFTWARE
Grupo Corporativo MARNA, S.A.	03
Alfa Asesores de Occidente, S.A.	04
Centro De Investigación Y Desarrollo Empresarial De Venezuela, C.A.	04
Laraweb Sistemas, C.A.	03
Universidad Centroccidental "Lisandro Alvarado" Decanato de Ciencias y Tecnologías	03
Universidad Nacional Experimental "Fuerzas Armadas"	03
Total	20

Fuente: Autora de la investigación 2012

Técnicas e Instrumentos de Recolección de Datos

Las técnicas son los medios o procedimientos de los cuales se valdrá el investigador para alcanzar los objetivos de la investigación. De allí que la recolección de los datos en este estudio se realizará a través de: consultas en Internet, investigación documental, técnicas de resumen, subrayado, ilustraciones, diagramas que apoyaran a la investigación de manera teórica.

Hurtado (ob.cit), señala que los instrumentos "constituyen un conjunto de pautas e instrucciones que orientan la atención del investigador hacia un tipo de información específica para impedir que se aleje del punto de interés" (p.409).

Procedimiento

Para llevar a efecto el estudio se procederá a realizar las siguientes actividades:

1. Arqueo y Revisión de la literatura pertinente al estudio. Con este propósito se ficharon textos especializados, consultas electrónicas, trabajos de investigación, artículos, revistas y fuentes legales.
2. Definir un Middleware que implemente tecnologías de integración basadas en servicios, un componente de mediación que provee enrutamiento y transformación de datos para interoperabilidad las ingenierías del dominio y de la aplicación.
3. Analizar las estructuras, formatos de los artefactos involucrados en la integración de las ingenierías del dominio representado por el proceso InDoCaS y de la aplicación.
4. Definir un repositorio y su meta-data que permita implementar las estructuras establecidas en los servicios de las aplicaciones de las ingenierías del dominio representado por el proceso InDoCaS y de la aplicación.

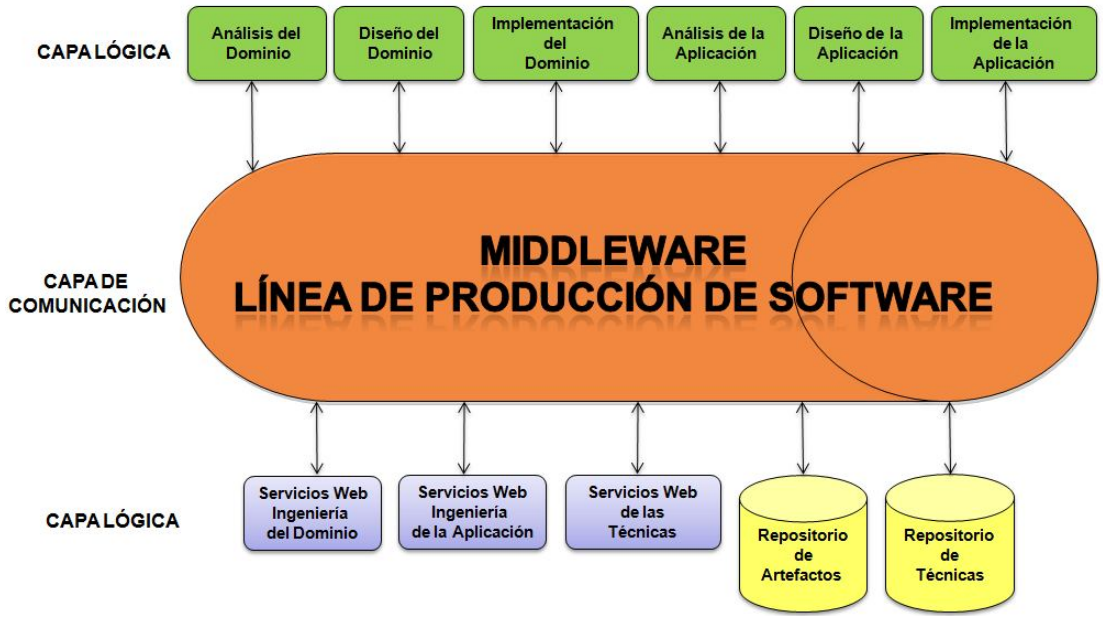


Figura 6. *Modelo Arquitectural para una línea de producción de software.*
Fuente: Autor de la Investigación 2012

Fase II. Estudio de Factibilidad

En esta fase se abordarán los aspectos que apoyan la posibilidad del desarrollo de la propuesta, inherente a la factibilidad teórica, técnica, económica y social:

Factibilidad Teórica

En esta fase se desarrollarán los elementos teóricos puntuales que sustentan la factibilidad de integración e interoperabilidad mediante tecnologías basadas en servicios y bus de integración.

Factibilidad Técnica

En esta fase se analizarán los aspectos técnicos involucrados en el desarrollo de un modelo arquitectural, a fin de detectar si la propuesta del estudio tiene factibilidad de uso con respecto a "licencias libres" propuestas por el Gobierno Nacional Venezolano para el presente año 2012.

Asimismo, se determinará la factibilidad en cuanto al Sistema Operativo GNU/GPL, como por ejemplo Mandriva 2008 o superior, licencia libre. Sin embargo podría instalarse en Sistemas Operativos de carácter propietario por ser un modelo transportable a cualquier plataforma (WINDOWS, SOLARIS, entre otros).

Factibilidad Económica

Esta fase se desarrollará con el propósito de constatar la factibilidad económica del modelo arquitectural enmarcado en un Middleware para la integración de las aplicaciones del proceso InDoCaS y la ingeniería de la aplicación para una línea de producción que permita el intercambio de información entre las aplicaciones, lo cual estará centrado en las condiciones para la adquisición de licencias de software para el

desarrollo del proyecto, costos, proyección del producto, entre otros aspectos. Se propone entonces, realizar el diseño de componentes y despliegue a través de herramientas de modelado para UML.

Factibilidad Social

En esta fase, se estudiarán los elementos factibles de la propuesta que brindarán beneficio a las empresas y a la sociedad en general en relación a los conocimientos de computación, tecnología y desarrollo de software, lo cual permitirá el crecimiento de la sociedad.

Fase III. Diseño de la Propuesta

Una vez desarrolladas las fases de Diagnóstico y Factibilidad, se procederá al diseño de la propuesta de un modelo arquitectural para una línea de producción que permita la integración de las ingenierías del dominio y de la aplicación usando al proceso InDoCaS como guía en el enfoque de calidad, que permita la integración e interoperabilidad mediante tecnologías basadas en servicios y bus de integración.

En este trabajo, el diseño de la propuesta incluirá los siguientes elementos que caracterizan el modelo arquitectural: un middleware para la línea de producción de software como un componente de mediación que provea servicios de enrutamiento, transformación de datos para interoperabilidad las ingenierías del dominio y de la aplicación; repositorios de técnicas y artefactos; servicios Web de las ingenierías del dominio y de la aplicación; servicios Web de las técnicas utilizadas en la ejecución de las funcionalidades de las ingenierías del dominio y de la aplicación.

CAPÍTULO IV

PROPUESTA DEL ESTUDIO

Justificación

Las líneas de producción de software, han aparecido como un enfoque cuyo objetivo es crear diferentes versiones de productos de software a partir de una infraestructura común. Este enfoque aplicado apropiadamente, puede producir múltiples beneficios y en última instancia, dar a las organizaciones ciertas ventajas competitivas estratégicas. Las líneas de producción de software a través de los procesos de la ingeniería del dominio y de la aplicación hacen una sistemática reutilización de la arquitectura de software.

El modelo arquitectural para una línea de producción de software, propuesto en la presente investigación representa un modelo de referencia para el desarrollo de un middleware que permita integrar las ingenierías del dominio y de la aplicación con el fin de aportar activos a la líneas de producción de software para mejorar la gestión del desarrollo y la calidad del software que produce.

Cabe destacar que los beneficios que se generan al integrar los procesos de la línea de producción de software son: disminución de costo de producción, reducción en el tiempo promedio de creación y entrega de nuevos productos, disminución de productos defectuoso y mejoras en el valor del producto que influyen sobre la competitividad de las organizaciones.

Una línea de productos de software integrada en sus fases fundamentales permitirá a la industria de desarrollo del software reutilizar los activos de la línea para que los desarrolladores puedan manejar la variabilidad existente en los diversos dominios de

aplicación y producir software de una manera más rápida, económica y de calidad. Asimismo, proporcionará un modelo de referencia a la academia, con el fin de difundir a la comunidad nuevas alternativas para el desarrollo de sus aplicaciones. En general para la sociedad y las organizaciones se incorporaran una infraestructura para desarrollar sistemas de información de calidad, con el fin de dar respuestas rápidas a solicitudes continuas de clientes y usuarios.

Los resultados derivados del presente estudio son un aporte significativo en la línea de investigación de Ciencias de la Computación mención Ingeniería de Software, lo cual servirá como elemento base para emprender otras investigaciones destinadas a solucionar otros problemas que afecten a la industria del software, organizaciones, la academia, entre otros.

El presente trabajo de investigación se realizó con la finalidad de brindar un aporte a la sociedad académica, programadores, arquitectos de software, integradores, centros de desarrollo de software, entre otros, que se interesan en el desarrollo de software específicamente en la integración de aplicaciones.

Objetivos

Objetivo General

Proponer un modelo arquitectural para una línea de producción que permita la integración de las ingenierías del dominio y de la aplicación usando al proceso InDoCaS como guía en el enfoque de calidad.

Objetivos Específicos

1. Analizar el enfoque de líneas de producción de software.

2. Definir un Middleware que implemente tecnologías de integración basadas en servicios para las ingenierías del dominio y de la aplicación.

3. Estructurar un componente de mediación que provea servicios de enrutamiento, transformación de mensajes y soportes para los diversos protocolos.

4. Analizar la estructura, formatos de los artefactos y técnicas involucradas en la integración de las ingenierías del dominio representado por el proceso InDoCaS y de la aplicación.

5. Definir repositorios y su meta-data que permita implementar las estructuras, formatos establecidas en los artefactos y técnicas de las ingenierías del dominio representado por el proceso InDoCaS y de la aplicación.

Estructura del Modelo Propuesto

1. Enfoque de líneas de producción de software.

La reutilización del software es el proceso que permite la utilización de los activos de software existentes para la construcción de otros nuevos o para modificar los existentes; actualmente existen diferentes técnicas y métodos de reutilización, una de las más efectivas es la creación de líneas de productos de software o familias de aplicaciones (Cabello, 2008).

La línea de producción de software es una familia de sistemas de software que comparten un conjunto común de características y son desarrolladas usando activos de software (modelo arquitectural, componentes de software, plan de pruebas,

modelos, entre otros) para un segmento en el mercado. Para la construcción de una línea de producción de software se tienen dos procesos fundamentales: a) la ingeniería del dominio se centra en el desarrollo de artefactos reutilizables que formarán la familia de productos, b) la ingeniería de la aplicación se encarga de desarrollar productos individuales, que pertenecen a la familia de productos, cumpliendo con los requisitos del usuario, reutilizando y adaptando los artefactos producidos en la ingeniería de dominio.

El proceso de desarrollo de la línea de producción de software utiliza técnicas y métodos para crear la línea de productos; Según Rivero (2011), los métodos más representativos de la industria son: Attribute-Driven Design (ADD), Siemens' 4 view (S4V), Rational Unified Process (RUP 4+1), Organización y Procesos de Arquitecturas de negocios (BAPO), Modelo de Separación de Arquitecturas (ASC), Modelo general para el diseño de arquitecturas y proceso para la ingeniería del dominio basado en calidad de software (InDoCaS).

El proceso InDoCaS usa el modelo general para el diseño de arquitecturas, que utiliza los métodos ADD, RUP 4+1, BAPO y ASC, y lo combina con el modelo de calidad ISO/IEC 25010, característica que no está explícita en otros modelos de procesos existentes.

InDoCaS es utilizado en el enfoque de desarrollo de líneas de productos de software y puede ser aplicado para un dominio específico; los activos de software producidos pueden ser reutilizados para generar un producto de una familia particular del dominio. Las disciplinas principales de InDoCaS son: Análisis y diseño del Dominio; el análisis utiliza un modelo de clasificación de requisitos (RECLAMO), ISO/IEC 25010 y FODA; y el diseño está basado en un modelo general de diseño arquitectural que concreta síntesis y evaluación arquitectural (Canelón, 2010). Mientras que la disciplina implementación del dominio fue desarrollada a través del proceso InDoCaSE instanciando el método WATCH – Component para construir actividades y generar los artefactos correspondientes (Rivero, 2011).

2. Modelo Arquitectural para una línea de producción de software.

El modelo arquitectural propuesto define la estructura jerárquica de los componentes de una línea de producción de software y la manera en que los componentes interactúan; asimismo se mostraran los servicios o funcionalidades que ofrecen a través de interfaces definidas.

El diseño arquitectónico se representa mediante dos modelos: el modelo estructural que muestra la arquitectura como una colección organizada de componentes; y el modelo marco de trabajo que define un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del middleware para una línea de producción de software.

Los componentes de una línea de producción de software son: Análisis del Dominio, Diseño del Dominio, Implementación del Dominio, Análisis de la Aplicación, Diseño de la Aplicación, Implementación de la Aplicación, Middleware para una Línea de Producción de Software, Servicios Web de la Ingeniería del Dominio, Servicios Web de la Ingeniería de la Aplicación, Servicios Web de la Ingeniería de la Aplicación, Servicios Web de Técnicas, Repositorio de Técnicas y Repositorio de Artefactos. Ver figura 7.

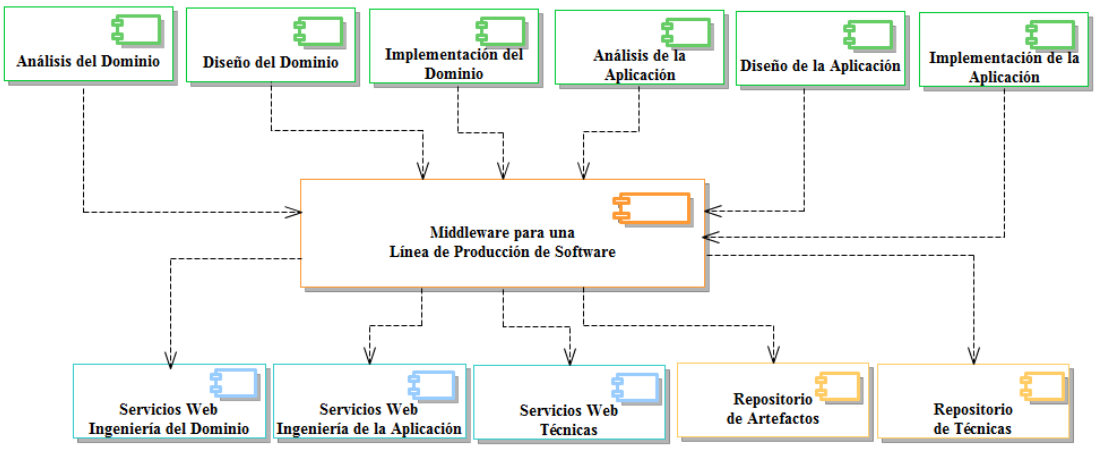


Figura 7. *Diagrama de Componentes para una línea de producción de software.*
Fuente: Autor de la investigación 2012

El diagrama de despliegue se muestra los nodos donde se ejecutan cada componente de una línea de producción de software.

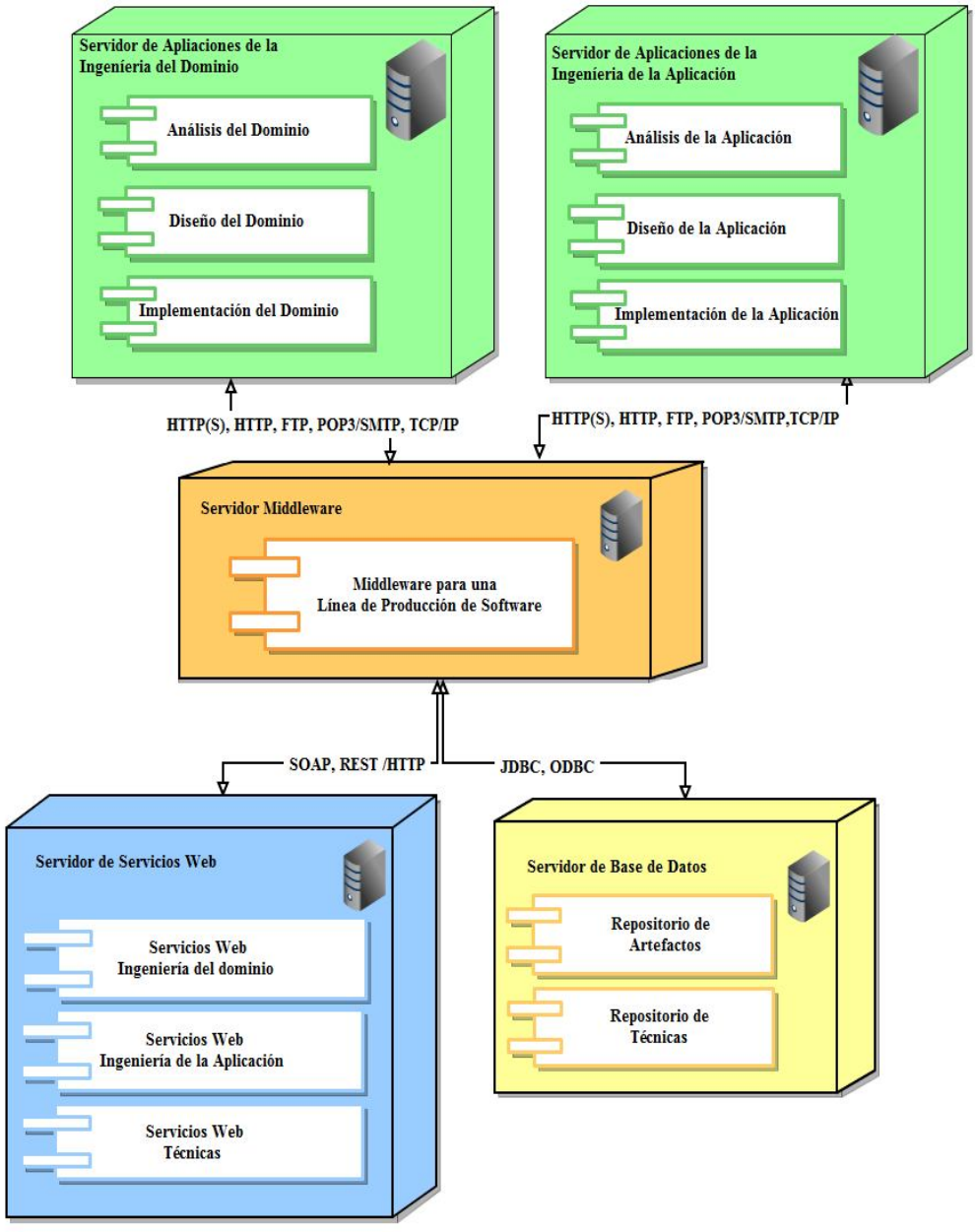


Figura 8. *Diagrama de Despliegue para la línea de producción de software.*
Fuente: Autor de la investigación 2012

A continuación se describen los diversos componentes de software que estructuran al diagrama de despliegue para una línea de producción de software:

2.1. Servidor de Aplicaciones de la Ingeniería del Dominio.

Este nodo contiene los siguientes componentes: análisis, diseño e implementación del dominio. Estos módulos se comunican con el componente "Middleware para una Línea de Producción de Software" utilizando los protocolos de transporte (HTTP(S), HTTP, FTP, POP3/SMTP, TCP/IP, entre otros) para solicitar los servicios que están publicados en el componente servicios web de la ingeniería del dominio y el componente servicios web de técnicas. Ver Figura 9.

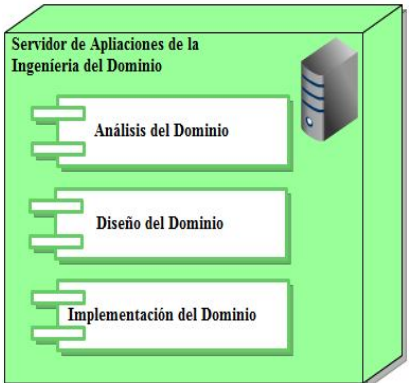
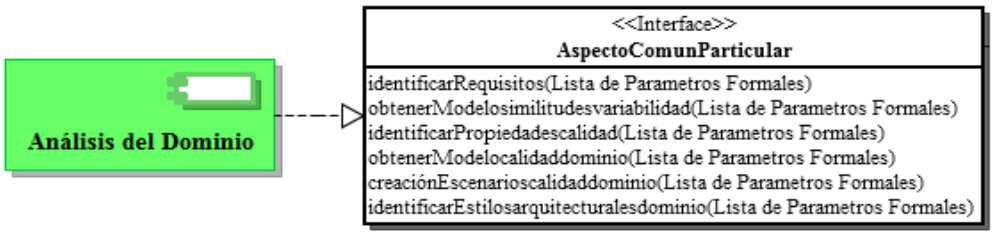


Figura 9. *Nodo "Servidor de las Aplicaciones de la Ingeniería del Dominio".*
Fuente: Autor de la investigación 2012

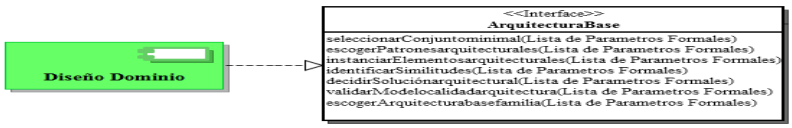
Cuadro 4. Componentes de la Ingeniería del Dominio

DESCRIPCIÓN DEL ANALISIS DEL DOMINIO
Análisis del dominio se refiere a los aspectos comunes a todas las familias del dominio y los aspectos particulares de cada una de ellas. Proporciona la interfaz AspectoComunParticular y requiere de la interfaz Mediación.    Este componente solicita al componente "Middleware para una Línea de Producción de Software" las técnicas (RECLAMO, FODA, ISO/IEC 25010 y ADD) para ser utilizadas en la ejecución de sus funcionalidades. Esta solicitud se realiza a través de los servicios web de técnicas (obtener RECLAMO, obtener FODA, obtener ISO/IEC 25010, obtener ADD). Las funcionalidades del componente análisis del dominio están basadas en: Identificación de requisitos, Obtener modelo de similitudes y variabilidad, Identificación de propiedades de calidad, Obtener modelo de calidad asociado al dominio, Creación de escenarios de calidad del dominio e Identificar estilos arquitecturales para el dominio    .

DESCRIPCIÓN DEL ANALISIS DEL DOMINIO

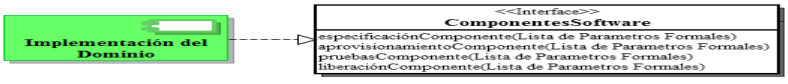
Diseño del Dominio define la arquitectura base para la familia. Proporciona la interfaz ArquitecturaBase y requiere de la interfaz Mediación.

Solicita al componente "Middleware para una Línea de Producción de Software" las técnicas (RECLAMO, FODA, ISO/IEC 25010 y ADD) y los artefactos generados por el componente análisis del dominio (Conjunto Minimal de Requisitos, Estilos Arquitecturales, Catalogo de Patrones Arquitecturales, una familia del dominio, Modelo de calidad del dominio, clasificación y asignación de prioridades y Arquitectura validadas a la familia). Esta solicitud se realiza a través de los servicios web de (técnicas, ingenierías del dominio y de la aplicación). Sus funcionalidades: Seleccionar los elementos del diseño del dominio que satisfagan el conjunto minimal de requisitos funcionales y no funcionales, Escoger patrones arquitecturales candidatos, Instanciar elementos arquitecturales, Identificar similitudes entre los elementos arquitecturales instanciados, Decidir la selección de la arquitectura como solución arquitectural candidata, Validar modelo de calidad del dominio con arquitectura candidatas, Escoger arquitectura base para la familia.



Implementación del Dominio define componentes de software a través de la arquitectura obtenida en el diseño del dominio. Proporciona la interfaz ComponentesSoftware y requiere de la interfaz Mediación.

Solicita al componente "Middleware para una Línea de Producción de Software" el método WATCH–COMPONENT que se encuentra almacenado en el componente repositorio de técnicas; solicita los artefactos generados por el componente diseño del dominio, para ser utilizados en la ejecución de sus funcionalidades. Estos artefactos son: Elementos del Diseño Conceptual, Patrones Arquitecturales Candidatos, Elementos Arquitecturales, Elementos Arquitecturales Similares, Solución Arquitectural, Informe de Decisión Arquitectural, Lista de requisitos de la familia, Modelo de calidad instanciado a la familia, Arquitecturas validadas a la familia y Documento de razonamiento arquitectural, Arquitectura de Referencia para la familia del dominio e Informe de decisión arquitectural. Esta solicitud se realiza a través de los servicios web de técnicas y artefactos. Sus funcionalidades: Especificación del componente, Aprovisionamiento del componente, Pruebas del Componente y Liberación del Componente.



Fuente: Autor de la investigación 2012

2.1. Servidor de las Aplicaciones de la Ingeniería de la Aplicación.

Este nodo contiene los siguientes componentes: análisis, diseño e implementación de la aplicación.

Estos módulos construyen productos individuales usando un subconjunto de artefactos de software disponibles en el repositorio de artefactos que son obtenidos a través del componente "Middleware para una Línea de Producción de Software"; se utiliza el protocolo de transporte (HTTP(S), HTTP, FTP, POP3/SMTP, TCP/IP, entre otros) para solicitar servicios que están publicados en el componentes servicios web de la ingeniería del dominio y en el componente servicios web de la ingeniería de la aplicación. Ver Figura 10.

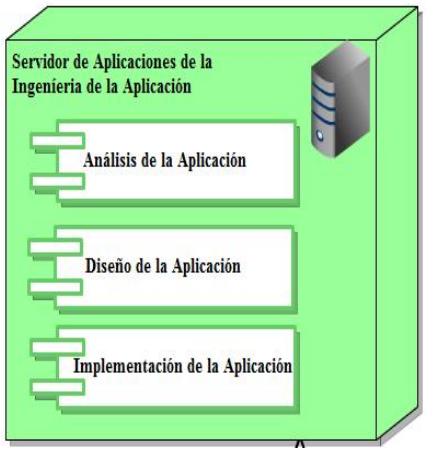
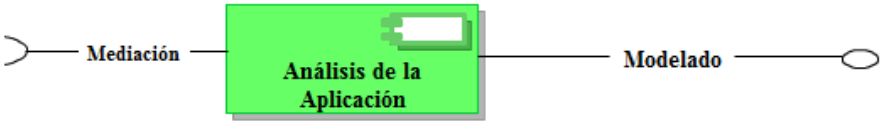
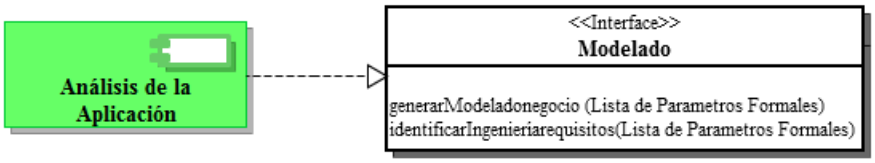
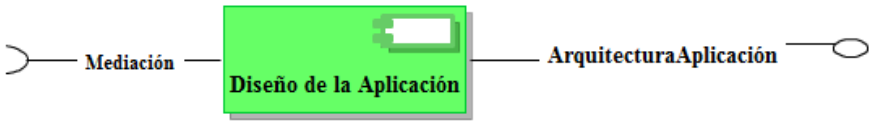
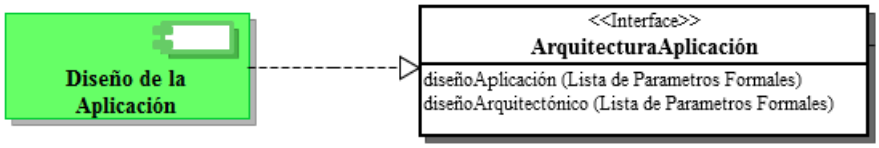


Figura 10. *Nodo "Servidor de las Aplicaciones de la Ingeniería de la Aplicación".*
Fuente: Autor de la investigación 2012

En el siguiente cuadro se describen los diversos componentes de la Ingeniería del Dominio:

Cuadro 5. Componentes de la Ingeniería de la Aplicación.

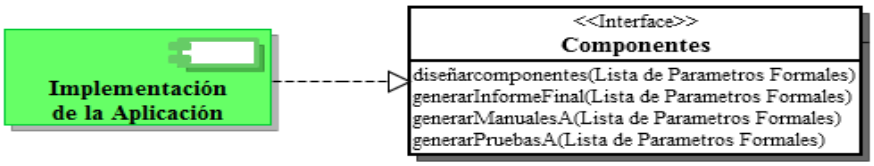
COMPONENTE	DESCRIPCIÓN
ANALISIS DE LA APLICACIÓN	Este componente proporciona la interfaz Modelado y requiere de la interfaz Mediación.  Este componente define requisitos mediante reutilización de componentes existentes, el conocimiento del dominio y las necesidades del cliente. Este componente solicita al componente "Middleware para una Línea de Producción de Software" los artefactos generados por el componente análisis del dominio (Listado de Requisitos Funcionales y No Funcionales, entre otros) que son utilizados en la ejecución de sus funcionalidades. Sus funcionalidades están basadas en el Modelado del negocio y en la Ingeniería de requisitos. 
DISEÑO DE LA APLICACIÓN	Este componente proporciona la interfaz ArquitecturaAplicación y requiere de la interfaz Mediación.  Este modulo genera la arquitectura de la aplicación; este componente solicita al componente "Middleware para una Línea de Producción de Software" los artefactos generados por el: componente de análisis de la aplicación (Listado de Requisitos de la Aplicación) y el componente de diseño del dominio (Arquitectura de Referencia (baseline) para la familia del dominio e Informe de decisión arquitectural) que son utilizados en la ejecución de sus funcionalidades. Sus funcionalidades están basadas en el Diseño de la Aplicación y en el Diseño Arquitectónico.



IMPLEMENTACIÓN DE LA APLICACIÓN

Este componente proporciona la interfaz Componentes y requiere de la interfaz Mediación.

Este componente selecciona y configura los componentes de software reutilizables; Este modulo solicita al componente "Middleware para una Línea de Producción de Software" los artefactos generados por: el componente de Diseño de la aplicación (Arquitectura de la Aplicación) y el componente de implementación del dominio (Componente seleccionado y Probado, Clasificación del componente y Documento de publicación del componente artefacto) que son utilizados en la ejecución de sus funcionalidades. Sus funcionalidades están basadas en: Diseñar componentes, Elaborar Informe Final, Generar manuales de la aplicación, Generar pruebas de la aplicación.



Fuente: Autor de la investigación 2012

2.2. Servidor Middleware.

Este nodo contiene el componente Middleware para una Línea de Producción de Software**,** permite la interoperabilidad entre las aplicaciones de las ingenierías del dominio y de la aplicación, que usan diversos formatos de datos o se ejecutan en plataformas heterogéneas. Ver Figura 11.

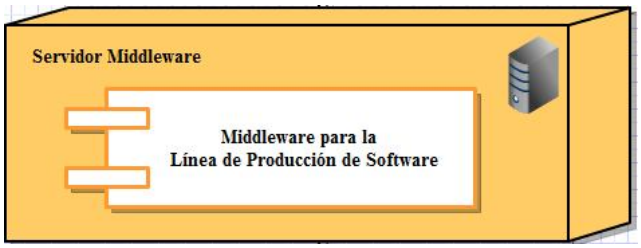


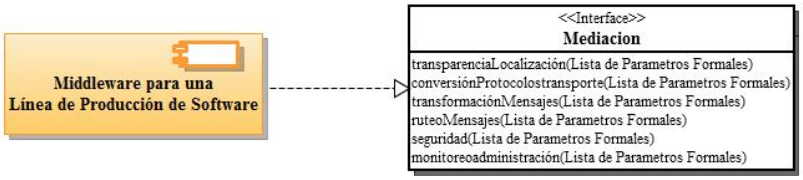
Figura 11. *Nodo "Servidor Middleware".*
Fuente: Autor de la investigación 2012

El Middleware para la Línea de Producción de Software (MLPS) está basado en las siguientes tecnologías de integración: Bus de Servicios Empresariales (Enterprise Service Bus, ESB) que implementa la Arquitectura Orientada a Servicios (Service Oriented Architecture, SOA) que se apoya en la tecnología de Servicios Web (Web Services, WS). De esta forma, las aplicaciones de las ingenierías del dominio y de la aplicación se comunican a través del middleware de forma independiente a la plataforma tecnológica en la que fueron implementadas. El ESB actúa como mediador entre las aplicaciones de las ingenierías que reciben y envían mensajes. Ver Cuadro 6.

Cuadro 6. El componente middleware para una línea de producción de software.

COMPONENTE	DESCRIPCIÓN
MIDDLEWARE PARA UNA LINEA DE PRODUCCIÓN DE SOFTWARE	Este componente proporciona la interfaz Mediación y requiere de las interfaces SWTecnicas, SWDominio, SWAplicacion, RArtefactos y RTecnicas.  Este componente tiene las siguientes funcionalidades: Transparencia de localización, Conversión de protocolos de transporte, Transformación de

mensajes, Ruteo de mensajes, Seguridad, Monitoreo y administración.

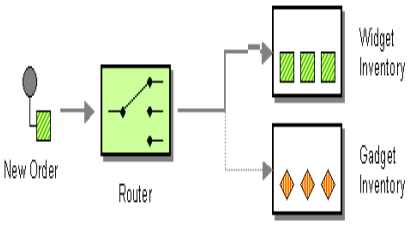


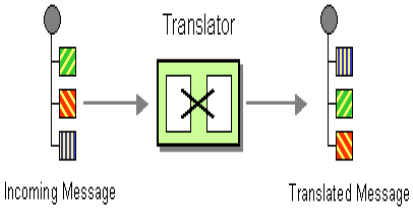
Fuente: Autor de la investigación 2012

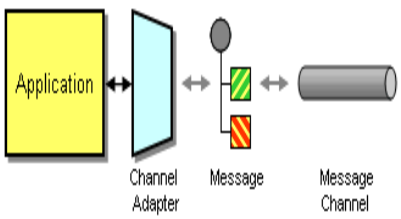
Este componente está basado en estándares abiertos (XML, XSL, WSDL, SOAP, entre otros) utilizadas para realizar las funciones básicas de mediación (servicios de enrutamiento, transformación de mensajes y soportes para los diversos protocolos). Asimismo, se utilizan patrones implementables usando funcionalidades estándares de ESB; estos patrones se enfocan en la resolución de problemas comunes en el uso de mensajería, y que se incorporan dentro de las funcionalidades de este tipo de mediadores, como por ejemplo el ruteo y transformación de mensajes.

En el siguiente cuadro se muestran las funciones del componente Middleware para la Línea de Producción de Software y los patrones que lo integran.

Cuadro 7. Funciones Básicas de Mediación

FUNCIONES DE MEDIACIÓN	DESCRIPCIÓN	PATRONES
RUTEO DE MENSAJES	Existen varios tipos de enrutamiento dentro de un ESB. Uno de los más usados es el enrutamiento basado en el contenido, introduce una serie de reglas o una lógica de negocio que se aplica al contenido del mensaje	**CONTENT-BASED ROUTER** 

	en la etapa del enrutamiento y que hacen posible que el middlware encamine los mensajes a proveedores de servicio específicos basándose en su contenido; dando prioridad, por ejemplo, a los solicitudes de determinadas aplicaciones o marcando las solicitudes de gran tamaño para darles un tratamiento especial. Para este componente existen los siguientes patrones: Message Router, Pipes and Filters, Message Dispatcher, Routing Slip, Dead Letter Channel, Dynamic Router, Content-Based Router, Recipient List, Message Filter, Message Broker, Channel Purger.	El enrutamiento basado en contenido es la capacidad de enrutar un mensaje en función de uno o más valores contenidos dentro del mensaje. El servicio de enrutamiento inspecciona cada mensaje y lo enruta al extremo de destino en función del contenido del mensaje y de la lógica de enrutamiento creada. El enrutamiento basado en contenido proporciona una base para la agregación de servicios, el control de versiones del servicio y el enrutamiento de prioridad.
TRANSFORMACIÓN DE MENSAJES.	Permite transformar mensajes de los servicios web de las ingenierías del dominio y la aplicación, a través del estándar XSLT. Para este componente existen los siguientes patrones: Message Translator, Splitter, Aggregator, Message History, Envelope Wrapper, Content Enricher, Content Filter, Claim Check, Messaging Mapper.	**MESSAGE TRANSLATOR**  Provee la capacidad de transformar el formato de un mensaje entrante al formato esperado por la aplicación destino a través de XSLT.
	Permite la conectividad a través de varios adaptadores que permite interactuar con múltiples protocolos de comunicación o	**CHANNEL ADAPTER**

SOPORTE DE PROTOCOLOS DE TRANSPORTE	transporte (HTTP, FTP, POP3/SMTP, sistemas de archivos, entre otros) para la integración de aplicaciones. Las ingenierías del dominio y de la aplicación pueden solicitar servicios de diversas maneras, por ejemplo, SOAP a través de JMS, SOAP a través de HTTP, entre otras. Para este componente existen los siguientes patrones: Channel Adapter, Messaging Bridge, Messaging Gateway, Service Activator.	 Este componente se encarga de la interacción con el sistema de mensajería en las primitivas que este brinde para intercambiar mensajes y los hará accesibles a la aplicación, invocando funcionalidades de la aplicación ante la llegada de mensajes y enviando mensajes ante eventos que ocurran en la aplicación. La implementación de este patrón se basa en el uso de las capacidades de conectividad multiprotocolo que provee un ESB. En general los ESB tienen incorporados adaptadores estándares para múltiples protocolos comunes (FTP, SMTP, HTTP, TCP, entre otros). Se pueden incorporar adaptadores particulares, para brindar conectividad al ESB (y a los servicios que estén disponibles dentro de este). El uso de estos adaptadores o conectores permite integrar aplicaciones que trabajan en otros protocolos diferentes a Web Services al bus de servicios del ESB, con lo que luego pueden interactuar con otros servicios expuestos en este de forma transparente.

Fuente: Autor de la investigación 2012

2.3. Servidor de Servicios Web.

Este nodo contiene los siguientes componentes: servicios web de ingeniería del dominio, servicios web de ingeniería de la aplicación y servicios web para obtener las técnicas. Estos servicios se encargan de exponer una funcionalidad bien definida a la aplicación que la requiera. Se utiliza los protocolos de transporte (SOAP, REST/HTTP(S), entre otros) para solicitar servicios que están publicados en los componentes de servicios web de las ingenierías del dominio, aplicación y las técnicas. Ver Figura 12.

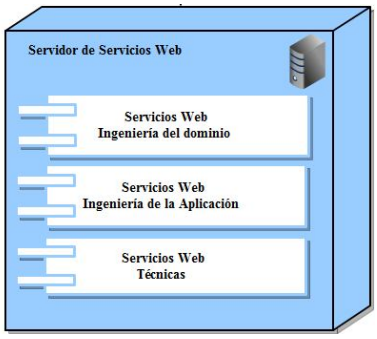


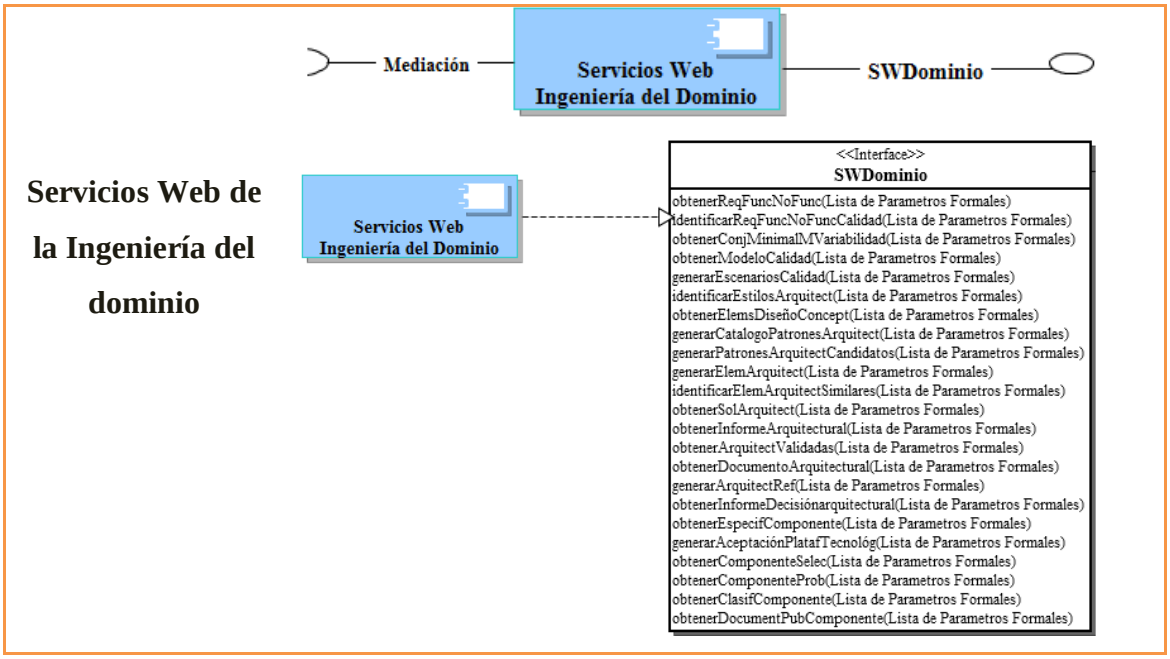
Figura 12. *Nodo "Servicios Web".*
Fuente: Autor de la investigación 2012

2.3.1. Servicios Web de la Ingeniería del dominio.

En este componente están publicados los servicios web de la ingeniería del dominio. Ver Cuadro 8.

Cuadro 8. Componente servicios web de la ingeniería del dominio.

COMPONENTE	DESCRIPCIÓN
	Este componente proporciona la interfaz SWDominio y requiere de las interfaces Mediación.



A continuación, en el cuadro 9, se presentan las funcionalidades Básicas del Componente Servicios Web de la Ingeniería del dominio utilizando la nomenclatura del método **InDoCaS**:

Cuadro 9. Funcionalidades Básicas del Componente Servicios Web de la Ingeniería del Dominio

INGENIERÍA DEL DOMINIO	N°.	FUNCIONALIDAD
Análisis del Dominio	F_01	Obtener de Requisitos Funcionales y Requisitos No Funcionales del Dominio.
	F_02	Identificar Requisitos Funcionales y No Funcionales y Propiedades de Calidad.
	F_03	Obtener Conjunto Minimal de Requisitos y Modelo de Variabilidad.
	F_04	Obtener el Modelo de Calidad del Dominio como instancia de ISO/IEC 25010.

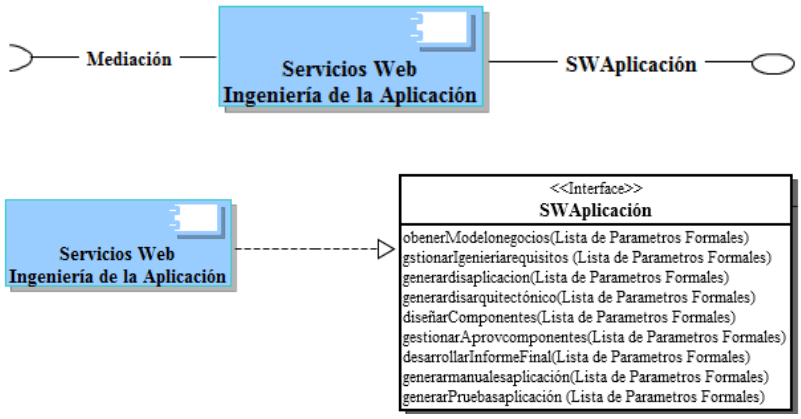
	F_05	Generar los Escenarios de Calidad.
	F_06	Identificar Estilos Arquitecturales para el Dominio.
Diseño del Dominio	F_07	Obtener Elementos del Diseño Conceptual.
	F_08	Generar Catalogo de Patrones Arquitecturales, Patrones Arquitecturales Candidatos.
	F_09	Generar los Elementos Arquitecturales (Componentes y Conectores).
	F_10	Identificar Elementos Arquitecturales Similares.
	F_11	Obtener la Solución Arquitectural y el Informe de Decisión Arquitectural.
	F_12	Obtener Arquitecturas validadas a la familia y el Documento de razonamiento arquitectural.
	F_13	Generar la Arquitectura de Referencia (baseline) para la familia del dominio y el Informe de decisión arquitectural.
Implementación del Dominio	F_14	Obtener la Especificación formal del Componente y Aceptación de la Plataforma Tecnológica.
	F_15	Obtener el Componente Seleccionado.
	F_16	Obtener Componente Probado
	F_17	Obtener Clasificación del Componente
	F_18	Obtener Documento de Publicación del Componente

Fuente: Autor de la investigación 2012

2.3.2. Servicios Web de la Ingeniería de la Aplicación.

En este componente están publicados los servicios web de la ingeniería de la aplicación. Ver Cuadro 10.

Cuadro 10. Componente servicios web de la ingeniería de la aplicación.

COMPONENTE	DESCRIPCIÓN
Servicios Web de la Ingeniería de la Aplicación	Este componente proporciona la interfaz SWAplicación y requiere de las interfaces Mediación. 

Fuente: Autor de la investigación 2012

A continuación, en el cuadro 11, se presentan las funcionalidades Básicas del Componente Servicios Web de la Ingeniería de la Aplicación, manteniendo la nomenclatura del proceso **InDoCaS**:

A continuación, manteniendo la nomenclatura del proceso **InDoCaS**, se presenta el resumen de la actividad:

Cuadro 11. Funcionalidades Básicas del Componente Servicios Web de la Ingeniería de la Aplicación

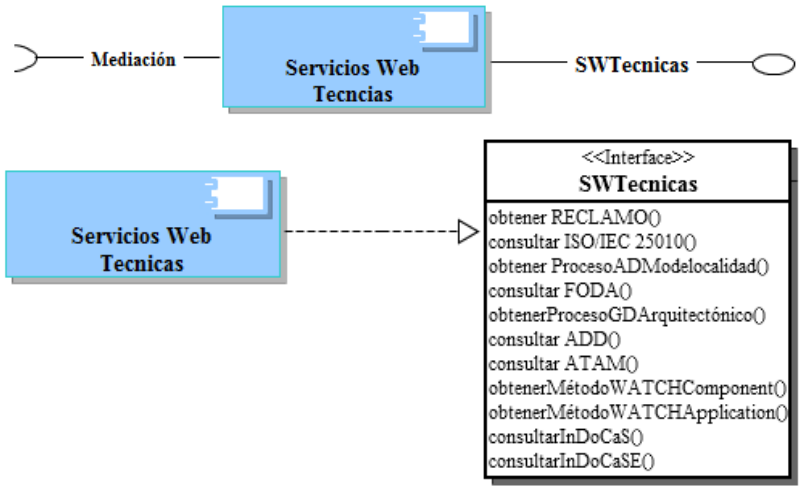
INGENIERÍA DE LA APLICACIÓN	N°.	FUNCIONALIDAD
Análisis de la Aplicación	F_19	Obtener el modelo de negocios
	F_20	Gestionar la ingeniería de requisitos
Diseño de la Aplicación	F_21	Generar el diseño de la aplicación
	F_22	Generar el diseño arquitectónico
Implementación de la Aplicación	F_23	Diseñar componentes
	F_24	Elaborar Informe Final
	F_25	Generar manuales de la aplicación
	F_26	Generar pruebas de la aplicación

Fuente: Autor de la investigación 2012

2.4. Servicios Web de Técnicas.

En este componente están publicados los servicios web de la ingeniería de la aplicación. Ver Cuadro 12.

Cuadro 12. Componente servicios web de técnicas.

COMPONENTE	DESCRIPCIÓN
Servicios Web de Técnicas	Este componente proporciona la interfaz SWTécnicas y requiere de las interfaces Mediación. 

Fuente: Autor de la investigación 2012

Cuadro 13. Funcionalidades Básicas del Componente Servicios Web de Técnicas

N°	FUNCIONALIDADES
F_01	Obtener RECLAMO - Requirement Clasification Model
F_02	Consultar ISO/IEC 25010
F_03	Obtener el proceso para el análisis del dominio para construir el modelo de calidad.
F_04	Consultar FODA - Feature-Oriented Domain Analysis
F_05	Obtener el proceso general para el diseño arquitectónico del dominio.

F_06	Consultar ADD - Attribute-Driven Design Method
F_07	Consultar ATAM - Architecture Tradeoff Analysis Method
F_08	Obtener el Método WATCH – Component.
F_09	Obtener el Método WATCH – Application.

Fuente: Autor de la investigación 2012

2.5. Servidor de Base de Datos.

Este nodo contiene los siguientes componentes: Repositorio de Artefactos y Repositorio de Técnicas. Ver Figura 13.

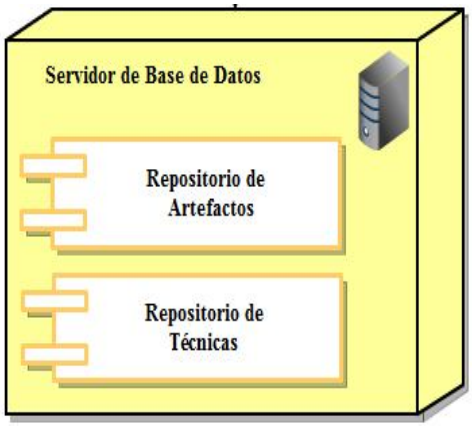


Figura 13. *Nodo "Servidor de Base de Datos".*
Fuente: Autor de la investigación 2012

La colección de artefactos y técnicas de una línea de producción de software necesitan ser almacenadas a través de repositorios de datos que faciliten la velocidad,

consistencia, soporte grandes cantidades de información, balanceo de carga y la replicación de datos. Se propone el uso de la base de datos NoSQL porque se adapta a los requisitos de un documento, ya que no imponen una estructura de datos en forma de tablas y relaciones, permiten almacenar información en otros formatos como clave-valor, mapeo de columnas, documentos o grafos, son más flexibles, las consultas son más rápidas.

2.5.1. Repositorio de Artefactos.

Este componente almacena artefactos generados de las ingenierías del dominio y aplicación. Este modulo tiene como objetivo asegurar la disponibilidad de activos para apoyar el desarrollo de productos de la línea de productos de software. Estos artefactos deben ser almacenados en una base de datos NoSQL orientado a documentos. Ver Cuadro 14.

Cuadro 14. Componente de Repositorio de Artefactos.

COMPONENTE DE REPOSITORIO DE ARTEFACTOS.	COMPONENTE REPOSITORIO DE ARTEFACTOS Y LA INTERFACE REQUERIDA.
	

Fuente: Autor de la investigación 2012

Cuadro 15. *Meta-data de los artefactos*

META-DATA DE ARTEFACTO	
Artefacto	**DESCRIPCIÓN**
Nro. De Identificación	
Nombre	
Nro. De Identificación de la Funcionalidad	
Nombre de la Funcionalidad	
Constructor	
Formatos Asociados	*Lista, Esquema, Modelo, Diagrama, Tabla*

Fuente: Autor de la investigación 2012

En los siguientes cuadros, se presentan los artefactos de salida de la ingeniería del domino en sus disciplinas análisis, diseño e implementación del dominio y los artefactos de salida de la ingeniería de la aplicación.

Cuadro 16. Colección de artefactos de la ingeniería del dominio.

		ARTEFACTOS	ESTRUCTURA Y FORMATOS
Ingeniería del Dominio	**Análisis del Dominio**	1. Lista de requisitos funcionales.	Documento que define requisitos funcionales a través de una tabla.
		2. Lista de requisitos no funcionales.	Documento que define requisitos no funcionales a través de una tabla.
		3. Conjunto de características.	Documento que contiene un diagrama FODA para presentar el conjunto de características.
		4. Conjunto minimal de requisitos funcionales y no funcionales.	Documento que define conjunto de requisitos mínimos y obligatorios de la familia a través de dos tablas.
		5. Conjunto de puntos de variación	Documento que muestra los puntos de variación a través de una tabla, los cuales describen dónde

			existen diferencias en las aplicaciones, expresan la variabilidad en las características.
		6. Lista de requisitos funcionales y propiedades de calidad asociadas. 7. Lista de requisitos no funcionales y propiedades de calidad asociadas.	Documento que define requisitos funcionales y propiedades de calidad asociadas a través de una tabla. Documento que define requisitos no funcionales y propiedades de calidad asociadas a través de una tabla.
		8. Modelo de calidad del dominio.	Este documento contiene un modelo estándar ISO/IEC 25010, el cual representa la calidad de un producto de software del dominio como una expresión acerca de la capacidad del software de ejecutar y mantener el nivel de servicio especificado.
		9. Escenarios de calidad.	En este documento se describe los requisitos arquitecturales, características del ISO/IEC 25010 y atributos a través de una Tabla.
		10. Estilos arquitecturales.	Este documento describe estilos arquitecturales que soporta el diseño arquitectural en el cual se representan los componentes y las relaciones entre ellos con las restricciones de su aplicación y las asociaciones y reglas del diseño para su construcción.
	Diseño del Dominio	11. Elementos del diseño del dominio	Este documento muestra a través de una tabla los requisitos funcionales, la importancia y dificultad.
		12. Patrones arquitecturales candidatos.	Este documento contiene un conjunto de patrones presentados a través una tabla para ser estudiados y revisados como posible solución arquitectural.

		13. Elementos Arquitecturales Similares (Componentes y conectores).	Este documento muestra la estructura de la arquitectura mediante un diagrama UML de componentes y sus relaciones.
		14. Conjunto de soluciones arquitecturales candidatas.	Este documento incorpora las soluciones alternativas o parciales que satisfacen el modelo de calidad del dominio. Y muestra los requisitos funcionales y el patrón escogido a través de una tabla.
		15. Soporte de decisión arquitectural.	El documento de soporte de decisión arquitectural está conformado por: comentarios de la escogencia, decisiones consideradas y rechazadas, trazabilidad de la decisión.
		16. Arquitecturas Validadas a la familia.	Este documento contiene los requisitos no funcionales, las variables de calidad asociadas y el patrón escogido; y se visualiza a través una tabla.
		17. Documento de razonamiento arquitectural.	El documento de razonamiento arquitectural está conformado por: comentarios, indicadores de razonamientos, una tabla que contiene los requisitos no funcionales, las variables de calidad asociadas, el patrón escogido y comentarios de decisiones consideradas y rechazadas.
		18. Arquitectura base para la línea del producto.	Este documento muestra la arquitectura mediante un diagrama de componentes y sus conectores.
		19. Informe sobre decisión arquitectural.	Es un documento que describe las ventajas o beneficios de la decisión arquitectural.

	Implementación del Dominio	20. Especificación formal del Componente	Es un documento que contiene funcionalidad esperada por el componente, sus interfaces y operaciones con sus parámetros de entrada, parámetros de salida, restricciones, pre y post condiciones, todo con ayuda del modelado UML 2.0. Se incluyen el contrato de uso y contrato de realización del componente a través de tablas.
		21. Aceptación de la Plataforma Tecnológica	Este documento muestra una tabla de evaluación para extraer la plataforma que más se ajusta a las funcionalidades de los componentes.
		22. Componente Seleccionado	Se muestra el tipo de aprovisionamiento seleccionado, la relación con la plataforma tecnológica seleccionada previamente y las funcionalidades del componente. Estas funcionalidades se muestran a través de un diagrama de componentes.
		23. Componente Probado	Es un documento muestra características y las funcionalidades del componente, el Diagrama de Componentes y el diseño del plan de prueba a través de tablas.
		24. Clasificación del componente	En este documento contiene los datos correspondientes a la clasificación del componente, el acceso que va a tener el componente, la especificación de cada una de sus formas. Esta información se muestra a través de tablas y diagrama de componente.
		25. Documento de publicación del componente	Es un documento que resume las especificaciones de las formas de componente, los resultados antes de la publicación y la información de los resultados obtenidos en la utilización que se obtenga después de su publicación.

Fuente: Autor de la investigación 2012

Cuadro 17. Colección de artefactos de la ingeniería de la Aplicación.

		ARTEFACTOS	ESTRUCTURA Y FORMATOS
Ingeniería de la Aplicación	**Análisis de la Aplicación**	1. Modelo del dominio de la Aplicación	Es un documento que contiene un conjunto de sub-modelos (Objetivos, Reglas del Negocio, Actores y Roles, Objetos del Negocio, Tecnologías), cada uno de los cuales representa un elemento organizacional del sistema de negocios, estos modelos son elaborados usando la notación UML
		2. Documentos de Requisitos	Este documento describe cada uno de los requisitos. Se estructura en dos partes: la primera parte está dirigida a los usuarios y consiste en describir la aplicación y sus requisitos en la terminología propia de los usuarios. La segunda parte está dirigida al Diseño de (casos de uso, modelo preliminar de clases, arquitectura preliminar de la aplicación, prototipo de la aplicación).
	Diseño de la Aplicación	3. Documento de Diseño de la Aplicación	El documento está constituido por el Diseño de interfaz usuario/sistema, Diseño de la base de datos, Diseño de componentes de software. Estos se muestran a través de los modelos en UML que representan la visión técnica de la arquitectura y las descripciones textuales que complementan y aclaran dicha especificación técnica.
		4. Documento de Diseño de la Arquitectura	El documento está constituido por el Diseño arquitectural (Casos de Uso, Interacción, Clases, Componentes, despliegue). Estos se muestran a través de los diagramas UML.
	Implementación de la	5. Documento de implementación	Este documento contiene la descripción y el código fuente de cada uno de los componentes arquitectónicos producidos, así como los detalles de su integración.

		6. Informe Final	Es un documento resume el desarrollo del proyecto, mediante la documentación de las principales decisiones que se tomaron, los logros alcanzados, las dificultades que se enfrentaron, los resultados de métricas usadas, entre otros.
		7. Aplicación	Es producto final del desarrollo de una aplicación, compuesto por sus tres elementos que la integran: programas, repositorios de datos (archivos y bases de datos) y documentos técnicos (manuales de uso y mantenimiento).
		8. Documento de Pruebas	Documento técnico que se produce durante el desarrollo de una aplicación. Su objetivo es documentar el diseño de las pruebas y los resultados de su ejecución.|

Fuente: Autor de la investigación 2012

En la tabla anterior, se puede observar que los artefactos de salida son documentos que contienen figuras, cuadros, texto, diagramas. Estos documentos deben ser almacenados en un repositorio descrito en la sección anterior.

2.5.2. Repositorio de Técnicas.

Este componente contiene las técnicas utilizadas en la construcción de las funcionalidades de las aplicaciones de las ingenierías del dominio y aplicación. Ver Cuadro 18.

Cuadro 18. Componente de Repositorio de Técnicas.

Componente de Repositorio de Técnicas.	Componente Repositorio de Técnicas y la interface requerida.
	

Fuente: Autor de la investigación 2012

Estas técnicas deben ser almacenadas en una base de datos NoSQL orientado a documentos utilizando un conjunto de meta-datas para las búsquedas.

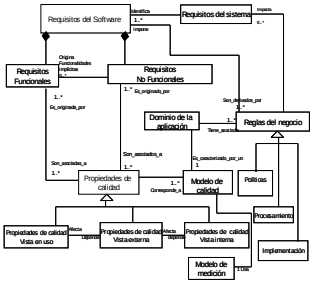
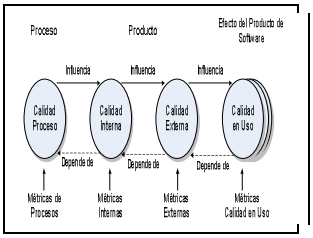
Cuadro 19. Meta-data de técnicas

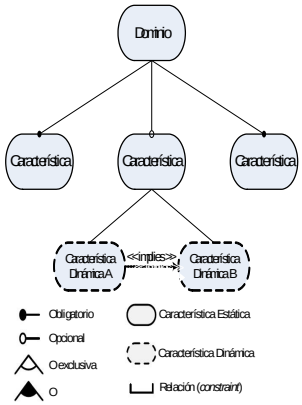
META-DATA DE TÉCNICAS	
Técnicas	**DESCRIPCIÓN**
Nro. De Identificación	*T01, T02, … Tn*
Nombre	*RECLAMO, ISO/IEC 25010, FODA, ADD, ATAM, Método WATCH – Component, InDoCaS, InDoCaSE, Método WATCH*
Formatos Asociados	*Listas, Esquemas, Modelos, Diagramas, Tablas*

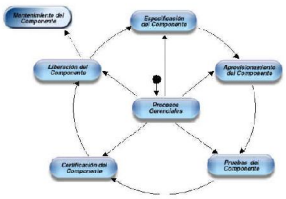
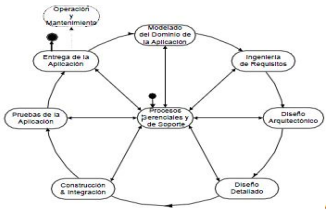
Fuente: Autor de la investigación 2012

En el siguiente cuadro, se muestra una colección de técnicas que son utilizadas en la construcción de las funcionalidades de las ingenierías del dominio y de la aplicación.

Cuadro 20. Colección de Técnicas.

NOMBRE Y DESCRIPCIÓN	APLICACIÓN	FORMATO ASOCIADO
RECLAMO (Requirement Clasification Model) Modelo de clasificación de requisitos, proporciona una taxonomía de requisitos que facilita la identificación de las diferentes clases de requisitos de un sistema de software. (Chirinos et al., 2004)	En la ingeniería del dominio específicamente: 1. Análisis del dominio en la Identificación de Requisitos y la identificación de de propiedades de calidad. 2. Diseño del dominio en la Determinación de la variabilidad de la familia y validación del modelo de calidad de la familia con arquitecturas candidatas.	 Es un documento que contiene un modelo para definir requisitos.
ISO/IEC 25010. El Estándar Internacional ISO/IEC 25010 describe un modelo bipartito para la calidad del producto de software, a) Calidad interna y externa b) Calidad de Uso. (ISO/IEC 25010, 2007) 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. Calidad externa, proporciona una visión de la "caja negra" del software y trata las características relacionadas	En la ingeniería del dominio específicamente: Análisis del dominio en la identificación de propiedades de calidad y formulación del modelo de calidad del dominio. Diseño del dominio en la determinación de la variabilidad de la familia y validación del modelo de calidad de la familia con arquitecturas candidatas	Se presenta como un documento que describe un modelo bipartito (Calidad interna y externa, Calidad de Uso) para la calidad del producto de software. Enfoque de Calidad  Características y sub-características

con la ejecución del software. Calidad de uso: es la medida de la calidad en su ambiente operacional para usuarios específicos que realizan tareas específicas.		
FODA (Feature-Oriented Domain Analisys) Análisis del dominio orientado a rasgos, desarrollado por el SEI (Software Engineering Institute) para el modelo de variabilidad. Es uno de los principales métodos de partida en el terreno del desarrollo de Líneas de Producción. (SEICMU, 1990)	Se utiliza en la ingeniería del dominio específicamente: Análisis del dominio en la Identificación de Requisitos y para obtener modelo de similitudes y variabilidad.	Es un documento que proporciona un modelo de variabilidad. 
ADD (Attribute-Driven Design Method) Método de diseño dirigido por atributos, usado para la formulación de escenarios de calidad; método para la definición de arquitecturas de software que basa el proceso de descomposición en los atributos de calidad que el software tiene que cumplir.	En la ingeniería del dominio específicamente: Análisis del dominio en la creación de escenarios de calidad.	Es un documento que proporciona un método para la definición una arquitectura de software.

Método WATCH – Component. Es un método de desarrollo especializado en producir componentes de software reutilizable derivado de una variación al método WATCH. (Hammar y Montilva, 2003).	Se utiliza en la ingeniería del dominio específicamente en la implementación del dominio. Las actividades instanciados a usar del método WATCH – COMPONENT son: Especificación del componente, Aprovisionamiento del componente, Pruebas del componente y Liberación del componente.	El modelo de proceso se representa con una figura que en su estructura emplea la metáfora de un reloj de pulsera para describir el orden de ejecución de los procesos técnicos de desarrollo de aplicaciones.  Las actividades se representan como fases y flujos de procesos que contiene un conjunto de tareas
El método WATCH – Application. Modelo de procesos para el desarrollo de aplicaciones empresariales (Montilva y Barrios, 2004)	En la ingeniería de la aplicación	Documento fundamentado en las mejores prácticas de la Ingeniería de Software y cubre todo el ciclo de vida de las aplicaciones; modelado, definición de los requisitos de los usuarios, puesta en operación de la aplicación. 

ATAM (Architecture Tradeoff Analysis Method) Método para elegir una arquitectura para un sistema. Las Fases del ATAM. Fase 0. Características. Fase 1. Características. Fase 2. Características. Fase 3. Características.	Se utiliza en la ingeniería del dominio específicamente en el diseño de dominio para la evaluación arquitectural. Las Etapas del ATAM: 1. Presentar el ATAM. 2. Presentar el objetivo de la Organización que utilizará el sistema. 3. Presentar la arquitectura. 4. Identificar los enfoque arquitectónicos. 5. Generación del Árbol de Servicios con los atributos de calidad. 6. Analizar las soluciones arquitectónicas. 7. Escenarios priorizados. 8. Análisis de los esquemas arquitectónicos presentados 9. Presentar los resultados.	Documento que define un método que consta de nueve etapas y cuatro fases para la evaluación de la arquitectura.

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

Después de realizar el modelo arquitectural para una línea de producción de software, se llegan a las siguientes conclusiones:

1. Se ha definido un modelo arquitectural para una línea de producción de software que contiene los siguientes componentes: análisis del dominio, diseño del dominio, implementación del dominio, análisis de la aplicación, diseño de la aplicación, implementación de la aplicación, servicios web de la ingeniería del dominio, servicios Web de la ingeniería de la aplicación, repositorio de técnicas, repositorio de artefactos, middleware para una línea de producción de software.
2. Mediante la instanciación del método WATCH – Application se incorporo un conjunto de funcionalidades que guían al proceso de la ingeniería de la aplicación.
3. El Middleware para la Línea de Producción de Software está basado en las siguientes tecnologías de integración: Bus de Servicios Empresariales que implementa la Arquitectura Orientada a Servicios que se apoya en la tecnología de Servicios Web.
4. El middleware para la Línea de Producción de Software actúa como un mediador entre las aplicaciones de las ingenierías del dominio y de la

aplicación, está basado en estándares abiertos utilizados para la mediación que provee servicios de enrutamiento, transformación de mensajes y soportes para los diversos protocolos.

5. Gracias a la integración de las ingenierías del dominio y la aplicación, usando al proceso InDoCaS como guía en el enfoque de calidad, es posible obtener una familia de productos con calidad para una línea de producción de software.
6. El modelo arquitectural propuesto permite que las arquitecturas resultantes de su instanciación desarrollen productos de software con características como la integrabilidad e interoperabilidad.

Recomendaciones

Con base en las conclusiones elaboradas como resultado de la investigación es necesario presentar algunas recomendaciones:

1. Desarrollar las aplicaciones de la ingeniería del dominio representada por InDoCaSE y de la aplicación para permitir la trazabilidad de sus actividades.
2. Incorporar al diseño del middleware para una línea de producción de software, patrones que solucionen problemas en las funciones de seguridad, monitoreo, administración y control de transacciones para completar el modelo arquitectural.
3. Implementar los objetos de aprendizaje en los repositorios de artefactos y técnicas, que permitan realizar una trazabilidad de los activos de software de una línea de producción y organizar la reutilización.
4. Incorporar artefactos y técnicas a las colecciones respectivamente, haciendo un estudio exhaustivo de los formatos utilizados.

GLOSARIO DE TÉRMINOS

Actividad: Conjunto estructurado de acciones o tareas realizadas por uno o más actores con la finalidad de alcanzar un objetivo predefinido. Una actividad utiliza recursos (insumos) para generar productos o prestar servicios.

Atributo de Calidad: característica de un producto por el cual se juzga la calidad de algunos actores o partes interesadas. Los requisitos de calidad tales como los de interoperabilidad, rendimiento, seguridad, confiabilidad y facilidad de uso tienen una influencia significativa en la arquitectura de software de un sistema.

Aplicaciones: programas de computación que provee un conjunto de funciones automatizadas para apoyar las actividades que realizan sus usuarios.

Dominio: es un área de aplicación de productos de software.

Componente: es una parte encapsulada del sistema de software; son entidades (clientes, servidores, bases de datos, filtros y capas de un sistema jerárquico) que tiene una interfaz.

Conector: itinerario de tiempo de ejecución de la interacción entre dos o más componentes.

Modelo: representación de un proceso, producto u otro elemento que interviene en el desarrollo de un sistema o aplicación.

UML *(Unified Modeling Language), Lenguaje de modelado de sistemas y software* que unifica un conjunto de notaciones diferentes que permiten modelar distintos aspectos de un sistema, tales como su estructura, funcionalidad, comportamiento e implementación.

WS *(Web Services), Servicios Web:* es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones.

REFERENCIAS BIBLIOGRÁFICAS

Cabello, M. (2008). *Baseline-Oriented Modeling: Una Aproximacion MDA basada en líneas de productos de software para el desarrollo de aplicaciones.* Universidad Politécnica de Valencia.

Canelón, R. (2010). *Un proceso para la ingeniería de dominio basado en calidad de software. Una aplicación al dominio del aprendizaje móvil sensible al contexto.* Universidad Central de Venezuela (UCV).

Caponi, M., Rodríguez, P., Zamudio, P. (2008). *Mensajería SI.* Universidad de la República de Montevideo, Uruguay.

Díaz, R. (2012). *A desamblar, Telecomunicaciones para la revolución*. Las teclas del conocimiento libertad y amor.

Garimella, K., Lees, M., Williams, B. (2008). *Introducción a BPM para Dummies*®. P 41 Wiley Publishing, Inc., Indianápolis, Indiana

Gómez E. (2011). *Modelo arquitectural para aplicaciones móviles usando el enfoque de líneas de producción dinámica de software.* Universidad Centroccidental Lisandro Alvarado (UCLA).

Hernández, R. Fernández, C. y Baptista, P. (2007). *Metodología de la investigación*. 4a. ed. México: McGraw-Hill Interamericana.

Hurtado, J. (2008). *Metodología de la Investigación Holística*. Caracas: Fundación Sypal.

Hurtado, J. (2010). *Metodología de la Investigación Holística.* (4ta ed.). Caracas-Bogotá. Ediciones Quirón.

López, A. (2007). *Guía para la puesta en marcha de un repositorio institucional*. Madrid: SEDIC.

Mendoza, L., Pérez, M., Titaeva, I. (2009).*Modelo basado en características para identificar modelos de integración de aplicaciones empresariales. Estudios de caso en Venezuela* 6(3): 52- 64.

Mondejar, R. (2010). *Distributed AOP Middleware For Large Scale Scenarios*. Tesis Doctoral. Universidad Rovira I Virgili. Tarragona

Montilva J., Barrios J. y Rivero M. (2008). *Gray Watch Método de Desarrollo de Software para Aplicaciones Empresariales.* Proyecto METHODIUS. FONACIT 2005000165. Mérida.

Montilva, J., Barrios, J. (2007). *Desarrollo de Software Empresarial*.

Poter, M. (2007). *Integración de aplicaciones*. TecnologyDay (26): 24-37

Pernalete, D., López, M., Montaño, N., Miguel, V. (2010). *IMS – Learning Design y el Modelo Arquitectural de AMBAR.* Universidad Nacional Experimental Francisco de Miranda.

Quintero, R. (2008). *Desarrollo Dirigido por Modelos de Aplicaciones Web que integran Datos y Funcionalidad a partir de Servicios Web.* Tesis Doctoral. Universidad Politécnica de Valencia, España.

Ramírez, T. (2007). *Cómo hacer un proyecto de investigación.* PANAPO. Caracas-Venezuela.

Rivero, (2011). *Una línea de producción de software para sistemas transaccionales con una aplicación al proceso de desarrollo de software en la Coordinación Nacional de Tecnología de Información de la Universidad Nacional Experimental Politécnica "Antonio José de Sucre" (UNEXPO)*. Universidad Centroccidental Lisandro Alvarado (UCLA).

Rodríguez, C., Pelegrina, A., Garrillo, J., Rodríguez, M., Bermúdez, M. (2010). *Diseño e implementación de software distribuido de soporte a la integración e interoperabilidad groupware.* Disponible en URL: http://www2.unalmed.edu.co/~pruebasminas/index.php?option=com_docman&task=doc_view&gid=1510&tmpl=component&format=raw&Itemid=285 (Consulta: noviembre 02, 2011)

Silveira, R., Pastor, J. (2009). *Servicios de Integración de SI Empresariales: Rol e Importancia de los Procesos de Negocio*. Universidad Politécnica de Catalunya, España.

Universidad Pedagógica Experimental Libertador (2007). *Manual de trabajos de grado de especialización y maestría y tesis doctorales. Fondo de la editorial Pedagógica Experimental Libertador*. Cuarta Edición. Barquisimeto, Venezuela.

A. CURRÍCULO VITAE DEL AUTOR

Deyanireth Coromoto Duarte Marín

Cursante del Postgrado en Ciencias de la Computación Mención Ingeniería de software de la Universidad Centroccidental "Lisandro Alvarado". Nació en Barquisimeto, Estado Lara, el 03 de Diciembre de 1981. Realizó estudios de Educación primaria en la Escuela Básica "Vicente Salías", Barquisimeto-Venezuela. Seguidamente cursó sus estudios de Educación Secundaria en el Liceo "Dr. Pastor Oropeza" y la Educación Diversificada en Escuela Técnica "Ambrosio Perera", Barquisimeto-Venezuela, donde obtiene el título de Bachiller en Ciencias mención Técnico Medio en Informática. Posteriormente inicia su carrera universitaria en la Universidad Centroccidental "Lisandro Alvarado" obtiene el título de "Ingeniero en Informática" en el año 2006. Entre tanto, obtiene los certificados de: **Manipulación de Alimentos** avalado por el Ambulatorio Don Felipe Ponte H. Tipo III Nro.: 000-235 en febrero 2012, **Actualización Profesional PHP Nivel Intermedio** avalado por Universidad Centroccidental "Lisandro Alvarado" Decanto de Ciencias y Tecnologías en Julio 2008, **Técnico en Sistemas de Telecomunicaciones I, Mención: Planta Externa** avalado por Centro de Tecnología y Sistemas de Telecomunicaciones Cesar Utches en Julio 2007, **Actualización Profesional JAVA nivel básico** avalado por Centro de Capacitación Profesional Micronet en Mayo 2007, **Asistencia al Curso Combo de Lenguaje C** avalado por Centro de Asesoramiento y Desarrollo Informático C.A. en Junio 2005, **Programador en Visual FoxPro 6.0 nivel básico** avalado por Centro de Computación G&T SISTEMAS, C.A. Febrero 2003, **Inglés Básico Level: Juvenile I y II, Adolescente Elementar I, II y III, Adultos I, II** avalado por Picadillo Enlisa Center Periodo (1996-2000), **Asistencia a la conferencia "Formación de Auditores en Gestión de la Calidad (ISO 9000-2000)"** avalado por FUNDAMETAL en Julio 2002, **Contabilidad Computarizada** avalado por Centro de Especialistas en contabilidad en Abril 2000, **Oficinista en Computación** avalado por Escuela de programación y sistemas de Gente Acción Solidaria en Septiembre 1999, **Elaboración De Proyectos Comunitarios** avalado por Universidad Central de Venezuela en Marzo 1998. Ha obtenido reconocimientos por haber obtenido la calificación sobresaliente de 19 y 20 puntos en las siguientes asignaturas: ***Contabilidad I, Desarrollo de habilidades y pensamiento III*** de la carrera Ingeniería en Informática. Posee experiencia laboral como analista de Sistemas en el departamento de Ventas y Manufactura de UNIPARTS, C.A. desde Octubre 2010 a Agosto 2011, como Especialista en Desarrollo en el departamento de proyectos en Grupo Corporativo MARNA, S.A. desde octubre 2007 hasta Julio 2010, como Programador en el departamento de informática en OPENLABS, C.A. desde Junio hasta Septiembre 2007, como Sectorizador en el departamento de COPRA de CANTV desde Enero hasta Abril 2007.