UNIVERSIDAD CENTROCCIDENTRAL
"LISANDRO ALVARADO"

MODELO DE INTEGRACIÓN BASADO EN LÍNEAS DE PRODUCCIÓN DE SOFTWARE PARA APLICACIONES RESTFUL

DESIREÉ LEILING MARTÍNEZ

Barquisimeto, 2012

UNIVERSIDAD CENTROCCIDENTRAL “LISANDRO ALVARADO”
DECANATO DE CIENCIAS Y TECNOLOGÍA
MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

MODELO DE INTEGRACIÓN BASADO EN LÍNEAS DE PRODUCCIÓN DE SOFTWARE PARA APLICACIONES RESTFUL

Trabajo de Grado presentado para optar al grado
Magíster Scientiarum en Ciencias de la Computación
Mención Ingeniería de Software

Por: DESIREÉ LEILING MARTÍNEZ

Barquisimeto, 2012

MODELO DE INTEGRACIÓN BASADO EN LÍNEAS DE PRODUCCIÓN DE SOFTWARE PARA APLICACIONES RESTFUL

Por: DESIREÉ LEILING MARTÍNEZ

Trabajo de Grado Aprobado

_____ _____

Jurado 1 Jurado 2

_____

Jurado 3

Barquisimeto, 27 de Noviembre de 2012

DEDICATORIA

"En quien he puesto toda mi confianza A ti Jehová, Dios Misericordioso que me llenas de Paz, Amor, Inteligencia y que me has bendecido con tu maravillosa presencia en mi vida..."

"A mi Madre Rosa Aura Martínez y mi Abuela Rafaela de Martínez aunque no estén cerca su presencia siempre me acompaña, por enseñarme a luchar para alcanzar mis metas, por sus oraciones, comprensión y amor..."

"A mi tía Marlen Martínez porque ha sido una fuente de inspiración en mi vida tía gracias por el ejemplo que has dado a mi vida."

"A mi Esposo José Meza, quién me ha motivado a alcanzar esta meta como parte de mi crecimiento profesional..."

"A mis sobrinos Jazmín, Onaimar, Andrés, Onailiz, Rubí y José Andrés quienes son mi alegría al despertar ustedes llenan mi vida de bendiciones, espero que esta meta les sirva de motivación y ejemplo de vida…"

AGRADECIMIENTO

A Dios por darme la capacidad, inteligencia y voluntad para lograr esta nueva meta alcanzada en mi vida.

A mi madre Rosa Aura, muchas gracias mami por hacer de mi lo que soy hoy en día ¡Te Amo!

A mi negro bello José Meza, por su paciencia, confianza y apoyo incondicional en todo momento. ¡Te amo cielo!

A mi amiga Gerana Espinoza por ser fuente de apoyo, motivación y excelente compañera de estudio. ¡Un abrazo!

A mis profesores Edison Sira, Edgar González y Leonardo Ponte por el apoyo brindado durante esta ruta llamada maestría.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

UNIVERSIDAD CENTROCCIDENTRAL “LISANDRO ALVARADO”
DECANATO DE CIENCIAS Y TECNOLOGÍA
MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

MODELO DE INTEGRACIÓN BASADO EN LÍNEAS DE PRODUCCIÓN DE SOFTWARE PARA APLICACIONES RESTFUL

Autor: Desireé Leiling Martínez
Tutor: Msc. Edgar R. González M

RESUMEN

La Integración de Aplicaciones Empresariales (IAE), representa el siguiente paso lógico en la evolución de la tecnología que comenzó con la automatización de procesos de negocio aislados y ha llegado afectar la manera cómo se lleva a cabo los negocios dentro de una empresa. Al mismo tiempo, IAE y las tecnologías que produce, ayudan a definirlas posibilidades futuras para el desarrollo de nuevos proyectos de software. En este sentido, la presente investigación propone un modelo de integración basado en líneas de producción de software para aplicaciones RESTful, que permita durante el proceso de desarrollo de software identificar las características de integración comunes y los componentes de software reutilizables, para lograr una creación eficiente de nuevas aplicaciones RESTful que comparten similitudes pero que varían en conocimientos y gestión de procesos; logrando de esta manera alcanzar mayor escalabilidad y mantenibilidad del software. Esta investigación está sustentada bajo la modalidad de proyecto especial, para la cual se definieron una serie procedimientos para llegar a la propuesta, entre estos los más significativos: caracterización de modelos de integración IAE, creación de líneas de productos de software para aplicaciones RESTful utilizando el proceso para la ingeniería de Dominio basado en calidad de software InDoCaS y por último un caso de estudio para el modelo propuesto.

Palabras Clave: Integración de Aplicaciones Empresariales, Modelos de integración IAE, Líneas de Producción de Software, Aplicaciones RESTful, InDoCaS.

INTRODUCCIÓN

Hoy en día, cada organización tiene su propia infraestructura, que consiste en sistemas, aplicaciones, personas y procesos. Los sistemas de Tecnologías de la información están diseñados para apoyarse entre ellos dentro de la misma organización. En este sentido, las organizaciones carecen de escalabilidad y de la capacidad de compartir información; es por ello que deben establecer estrategias que apunten hacia la conformación de sistemas integrados para aprovechar al máximo la infraestructura tecnológica que poseen en la actualidad.

Así mismo, (Ross, Weill y Robertson, citado en Pardo y Valero, 2011) señalan que las organizaciones actuales buscan continuamente estrategias en infraestructuras que le permitan evolucionar y mantenerse competitivamente en mercado. Una de esas estrategias es lograr la integración entre los sistemas existentes dentro de la empresa como también con otros sistemas externos. En atención a esto, surge el concepto de Integración de Aplicaciones Empresariales (IAE).

Según Linthicum (1999), IAE es el intercambio sin restricciones de datos y procesos de negocio entre las aplicaciones conectadas y fuentes de datos de la empresa. Por Otro lado, el Club BPM (2011), entiende por IAE el conjunto de tecnologías que permite el movimiento e intercambio de información entre diferentes aplicaciones y procesos de negocio dentro y entre organizaciones.

Es importante señalar que IAE, está representada por medio de modelos de integración; estos modelos expresan en un lenguaje común, las soluciones para la integración, capturan las múltiples posibilidades que representan para la organización y muestran como se combinan para lograr la solución deseada (Brown, citado en Mendoza et al., 2009).

En este orden de ideas, la integración o interoperabilidad entre aplicaciones, se logra cuando se establece una relación entre sus componentes; utilizando interfaces flexibles para proveer servicios o reutilizar funcionalidades, Pardo y Valero (2011). La interoperabilidad entre sistemas viene acompañada de una adecuada arquitectura,

por lo que resulta oportuno mencionar uno de los estilos arquitecturales que recientemente ha tenido un gran auge en el área de integración de aplicaciones.

En relación a esto, REST (Representational State Transfer) es un estilo arquitectónico propuesto por Fielding (2000) en su tesis doctoral. Según Wirdemann y Baustert (2007), REST describe todo un paradigma de arquitectura para aplicaciones que solicitan y manipulan recursos en la web utilizando los métodos estándar de HTTP. GET, POST, PUT y DELETE.

En tal sentido, REST se ubica dentro del marco de Servicios Web con arquitecturas comunes (aplicaciones RESTful). Ahora bien, de lo anteriormente expuesto se podría decir que las aplicaciones RESTful podrían considerarse como una línea de producto por lo siguiente, Barbosa (2010) se refiere a las líneas de productos aquellas que posibilitan la creación de un conjunto de sistemas en un mismo dominio que comparten una serie de características comunes.

Sumado a esto, las líneas de producción de Software (LPS) consisten en un ciclo de vida dual, en la que la primera fase (Ingeniería del Dominio) se construye una arquitectura común a la línea de productos y un conjunto de componentes reutilizables denominados "core-assets". En la segunda fase (Ingeniería de Aplicaciones) se construye los productos de software a partir de la personalización de la arquitectura, mediante técnicas de variabilidad de software y los componentes core, que constituyen gran parte de la funcionalidad del producto final, Barbosa (2010).

De esta manera, la investigación en curso aborda el estudio en el área de modelado de software, a fin de proponer un modelo construido a partir de características identificadas en los modelos de integración de IAE propuestos por Brown (2000), basado en el desarrollo de Líneas de Producción con arquitectura REST, con el propósito de mejorar el proceso de desarrollo de software en función del tiempo y calidad de la entrega.

Ante la situación planteada, el presente trabajo de investigación, está organizado en cinco (5) capítulos y secciones anexas:

Capítulo I El Problema: inicialmente se expone el contexto que genera el tema de la investigación u objeto de estudio, seguidamente se definen las metas y/o objetivos

que se quieren alcanzar con la investigación, posterior a esto se describen los argumentos por los cuales se debería realizar la investigación, llámese esto la justificación e importancia de la investigación. Y por último se definen los alcances de la misma.

Seguidamente el Capítulo II Marco Teórico. En donde, en primer lugar se hace referencia a los antecedentes relacionados con el tema de investigación, respecto a Modelos de Integración, Líneas de Producción de software (Ingeniería de Dominio y Aplicación) y REST, y por último se presentan las bases teóricas que soportan el objeto de estudio.

Luego en el Capítulo III Marco Metodológico. Se define la naturaleza del estudio comprendido por el tipo de investigación y diseño de la investigación. Además, se presentan las fases o tareas a realizar para diseñar la propuesta.

A continuación el Capítulo IV, contiene la propuesta del estudio, detallando su justificación, objetivos y descripción de la misma. Adicionalmente, se presenta un caso de estudio.

Finalmente, en el Capitulo V se encuentran las conclusiones o hallazgos y posibles recomendaciones generadas de la presente investigación.

CAPÍTULO I

EL PROBLEMA

Planteamiento del Problema

Uno de los retos que enfrentan las organizaciones actuales es el problema de la interoperabilidad entre los sistemas de información, puesto que los servicios se ofrecen a los usuarios a través de aplicaciones separadas; restringiendo la realización de consultas integrales, además de reducir la visión y proyección organizacional, Pardo y Valero (2011).

Según un estudio realizado por Frantz y Otros (2011), un desafío recurrente es hacer que estas aplicaciones interoperen entre sí para mantener sus datos sincronizados o para crear una nueva funcionalidad. No es raro que la integración se lleve a cabo por un usuario que copia los datos desde una aplicación y lo pega en otra. Esto obviamente no es escalable, lo que motiva a muchas empresas a invertir en soluciones automatizadas de IAE. Por desgracia, las aplicaciones no suelen ser fáciles de integrar debido a muchas razones, por ejemplo, las tecnologías en las que se basan son diferentes, sus APIs no son compatibles desde el punto de vista semántico, o no puede proporcionar una API en absoluto, que es el caso de muchas aplicaciones web.

Por otro lado, se observa que en las organizaciones destinadas a ofrecer servicios de integración entre sistemas se encuentran con la tarea de identificar el modelo de integración a utilizar para el desarrollo de la aplicación de integración, dependiendo del propósito de las organizaciones y de los niveles de integración que se requiera, bien sea a nivel de procesos ó a nivel de datos, Mendoza et al. (2009). Adicionalmente, se debe definir que componentes de los que ya tienen desarrollados sirven para la integración requerida, de manera que se garantice calidad del producto.

En este sentido, Brown (2000) propone varios modelos de integración entre ellos: Ciclo, Semilla, Web, Flujo, Onda, Anillo, célula y árbol. Estos modelos demuestran un amplio panorama de la aplicación de los mismos en diversos procesos organizacionales, donde cada proceso está relacionado con el propósito de la organización. Por otro lado, en Venezuela, Mendoza y Otros (2009) investigaron en el área de Integración de Aplicaciones Empresariales (IAE) y mencionan que esta integración constituye un proceso mediante el cual hardware, software y procedimientos de negocio se combinan haciendo posible una utilización más fácil de la información y de los sistemas.

De acuerdo a la investigación que ellos realizaron, no se han encontrado otras investigaciones que se hayan apoyando en este trabajo, por esto se hace necesario extender esta investigación ya tiene gran aporte en el área de IAE.

En otro orden de ideas, Fielding (2000), propone REST como un estilo arquitectónico que pretende atender los requerimientos de la web, entre los componentes de este estilo arquitectural, tenemos: funciona sobre una arquitectura cliente-servidor, los servidores no almacenan el estado de los clientes, la comunicación puede ser repetida cuando no hay cambios, clientes y servidores se comunican utilizando una interfaz uniforme; todas estas condiciones tienen como finalidad aumentar la interoperabilidad, desempeño, escalabilidad y fiabilidad de los sistemas.

En un estudio realizado por Pautasso y Otros (2008), donde compara REST con los servicios WS-* basado en decisiones arquitecturales, incluyendo las fortalezas y debilidades de ambas variantes. Ellos mencionan la confusión sobre las mejores prácticas y además identifican la falta de métodos estandarizados para el diseño.

Por su parte, Li et al. (c.p Pardo y Valero, 2011) indican que la interoperabilidad entre sistemas se logra básicamente utilizando interfaces flexibles para proveer servicios o reutilizar funcionalidades. Según Sommerville (2006), el proceso de diseño de software en la mayoría de las disciplinas de la ingeniería se basa en la reutilización de sistemas o componentes. Asimismo, Montilva et al. (2003) señala que bajo el modelo de desarrollo de software basado en componentes, las nuevas

aplicaciones se construyen mediante la integración o composición de componentes. Sametinger (c.p Montilva et. al, 2003) define la composición de software como "el proceso de construir aplicaciones mediante la interconexión de componentes de software a través de sus interfaces.

De lo anteriormente expuesto, Sommerville (2006) destaca que las aproximaciones más efectivas para la reutilización es la creación de líneas de productos o familias de aplicaciones. Aunado a lo anterior, es necesario destacar que una línea de productos de software es un conjunto de aplicaciones con una arquitectura común. En relación a esto, Díaz y Trujillo (2010), mencionan que el objetivo principal de una línea de productos de software es sacar partido de los elementos comunes y manejar de una manera eficaz las variaciones. El reto es delimitar el ámbito de este dominio, identificar las variaciones que se van a soportar y dotarse de la infraestructura que permita producir el producto a bajo coste y manteniendo los estándares de calidad.

Es por esto que las líneas de Producción de Software pueden incrementar significativamente la productividad de los ingenieros de software, de esta manera se genera una reducción en el esfuerzo y el coste necesario para desarrollar, y por consiguiente, colocar en marcha y mantener un conjunto de productos similares.

De acuerdo a las investigaciones realizadas, se han encontrado otras investigaciones que están relacionadas con LPS, como por ejemplo, el proceso para la ingeniería de dominio basado en Calidad de Software **InDoCaS** presentado por Canelón (2010), el cual tiene un gran aporte para el área de LPS abarcando las fases de análisis y diseño de la ingeniería de dominio, por esta razón se considera necesario extender esta investigación.

Asimismo, se hace énfasis en las aplicaciones RESTful, porque están definidas bajo un estilo arquitectural llamado REST, lo que quiere decir que estas aplicaciones poseen una arquitectura común a todas, de allí se estima como una Línea de producción de Software para aplicaciones RESTful. Cabe señalar, que antes de trabajar con esta nueva línea de productos, se debe tomar en cuenta los modelos de

integración de IAE porque son estos los que nos van indicar la ruta de integración de acuerdo los propósitos de las organizaciones y sus sistemas.

Por tanto, se considera necesario el estudio de un modelo de integración enfocado en Líneas de producción de Software para Aplicaciones RESTful, que permita durante el proceso de desarrollo de software identificar las características de integración comunes y los componentes de software reutilizables, para lograr una creación eficiente de nuevas aplicaciones RESTful que comparte similitudes pero que varían en conocimientos y gestión de procesos; logrando de esta manera alcanzar mayor escalabilidad y mantenibilidad del software.

De lo anteriormente expuesto, se plantean como interrogantes de esta investigación las siguientes:

- ¿Cuáles serán las características que definen los Modelos de Integración de IAE?
- ¿Cuáles serán los modelos de integración de IAE que se adecuan al dominio de aplicaciones RESTful?
- ¿En qué forma el enfoque de LPS beneficia el desarrollo de aplicaciones RESTful?
- ¿Cómo se deben combinar líneas de producción de software y las aplicaciones RESTful para conformar un modelo de integración?

Partiendo de estas interrogantes, es necesario señalar que las mismas obtendrán sus respuestas por medio de los objetivos de la presente investigación.

Objetivos de la Investigación

Objetivo General

Proponer un Modelo de Integración basado en Líneas de Producción de Software para Aplicaciones RESTful.

Objetivos Específicos

1. Describir las características de los Modelos de Integración de IAE.

2. Identificar los Modelos de Integración que se adecuan al dominio de Aplicaciones RESTful.
3. Comprender los beneficios que genera el uso de líneas de producción de software en aplicaciones RESTful.
4. Construir el Modelo de Integración basado en líneas de producción de software para aplicaciones RESTful.

Justificación e Importancia de la Investigación

La presente investigación fundamenta su justificación en proveer un modelo de Integración que permita identificar Modelos de Integración en Aplicaciones RESTful, y apoye el proceso de construcción de líneas de producción de software (LPS), en función de lograr el objetivo principal de las mismas, que es reducir el tiempo, costo y complejidad de crear y mantener los productos de la línea. Adicionalmente, el enfoque de LPS para Aplicaciones RESTful nos permite crear no solamente un producto de software sino conjunto de productos para el mismo dominio.

Por otro lado, el objeto de estudio se enfoca en Aplicaciones RESTful puesto que el estilo arquitectónico que representan, facilitan el desarrollo de los clientes, ofrecen mayor estabilidad frente a futuros cambios. Es necesario señalar que las aplicaciones RESTful están basadas en servicios web que ofrecen mayor simplicidad al momento de publicar el servicio y al momento de consumirlo. Por consiguiente, se busca desarrollar una línea de producción de software para este tipo de aplicaciones buscando eliminar vulnerabilidades como seguridad de los datos y calidad de la mismas, todo esto gracias al proceso de ingeniería de dominio que estará representado por ***InDoCas***, (Canelón (2010).

Ahora bien, se considera que los resultados de esta investigación son importantes en el área de aplicaciones para integración RESTful, líneas de producción de Software en su contexto de ingeniería de dominio, logrando ser un modelo guía para impulsar o crear nuevos proyectos en Empresas, Instituciones. Además, beneficia a la comunidad científica en el campo de investigación para estudios de Pregrado y

Postgrado tanto nivel Nacional como Internacional, teniendo en cuanta que los objetivos de la investigación generan nuevos conocimientos.

Desde el punto de vista de la ingeniería del software, se puede incorporar el modelo de Integración, como parte de las herramientas de análisis de problema en los ambientes de organizacionales y la Línea de Producción de Aplicaciones RESTful, para garantizar reducción tiempos de entrega y costos asociados al producto de integración.

Para finalizar, los usuarios de aplicaciones RESTful, podrán disfrutar de mejores tiempos de respuesta, facilidad desarrollo de las mismas, calidad en el servicio.

Alcance y Limitaciones de la Investigación

El alcance del presente estudio se ajusta a la presentación de un modelo de integración basado en líneas de producción de software para aplicaciones RESTful, que puede ser aplicado en empresas de cualquier sector y en cualquier parte del mundo, debido a que garantiza interoperabilidad y escalabilidad durante los procesos de cambios que ocurren en las organizaciones. En lo que se refiere a los aspectos asociados a la variabilidad se gestionara por medio del enfoque de líneas de producción de software.

Asimismo, se utilizará el proceso para la ingeniería de dominio basado en calidad de software **I**n**D**o**C**a**S**, para capturar las similitudes del dominio, y así lograr el desarrollo de software bajo el enfoque de líneas de producción de software.

Adicionalmente, las características de integración obtenidas se realizaron en base a los Modelos de integración de IAE propuestos por Brown (2000), para aclarar el contexto de la dinámica de los procesos de negocios relacionados con aplicaciones RESTful.

En lo que respecta a limitaciones, se tiene que el proceso para la ingeniería de dominio basado en calidad de software **I**n**D**o**C**a**S**, utilizado en la creación de la línea de producción para aplicaciones RESTful se aplicó en su etapa de análisis del dominio. Por otro lado, se tomó como antecedente la investigación de Mendoza y

Otros (2009), la cual tiene como limitante que su propuesta fue aplicada solo a Empresas Venezolanas. Así mismo es necesario mencionar que la caracterización de los modelos integración está limitada solamente a caracterizar cuáles modelos se adecuan a las aplicaciones RESTful, más no a la validación de los mismos.

CAPÍTULO II

MARCO TEÓRICO

Con el propósito de sustentar desde una perspectiva teórica el problema, se ha necesario presentar en el marco teórico del trabajo de investigación: en primer lugar, otras investigaciones que se han realizado, inherentes al problema en estudio (Antecedente de la investigación). En segundo lugar, aquellos enfoques teóricos derivados del paradigma que ha sido definido, vinculados con algunas dimensiones de análisis del problema (Bases Teóricas).

Antecedentes de la Investigación

Antecedente de modelos basado en características de Integración de Aplicaciones Empresariales

El proceso de software debe incluir un conjunto de actividades y resultados que generan un producto de software, este conjunto de actividades están enmarcadas en un modelo del proceso de desarrollo de software y son utilizadas para explicar diferentes enfoques en el desarrollo de software. A continuación se exponen los antecedentes de modelos basados en características de Integración de aplicaciones empresariales (IAE).

En lo tocante, Mendoza y Otros (2009), realizaron una investigación acerca de modelos de integración en aplicaciones empresariales, la cual tuvo como objetivo proponer un modelo basado en características que permita identificar los modelos de IAE propuestos por Brown (Ciclo, Semilla, Web, Flujo, Onda, Anillo, Célula y Árbol), para estudiar la factibilidad de su implantación en las organizaciones venezolanas. En consecuencia al modelo propuesto, se buscó determinar que modelos

o combinación de ellos se están aplicando en Venezuela, para ello se realizaron ochos (8) casos de estudio en organizaciones venezolanas.

Como resultado, del proceso de identificación de los modelos basado en el modelo de características propuesto, los autores encontraron que todos los modelos estaban presentes en las organizaciones y también se demostró que las organizaciones venezolanas pueden ampliar su conocimiento sobre Sistemas de Información y Tecnologías de Información a través del uso de tales modelos, mejorando la integración de sus sistemas. Esta investigación sirve de apoyo al presente proyecto, debido a que presenta características similares con el objeto de estudio, puesto descubre un procedimiento para identificar características de los modelos de integración de IAE presentes en las organizaciones.

Sin embargo, se limita a proponer un modelo de especificación basado en características de modelos de integración de aplicaciones en general, sin profundizar en la integración de aplicaciones más específicas como es el caso de aplicaciones RESTful sumada al enfoque de líneas de producción de software.

Antecedentes de líneas de producción y Aplicaciones RESTful

Se inicia con, Gómez (2011), quien en su trabajo de grado de maestría, investigó sobre líneas de producción de software en aplicaciones móviles, con el propósito de proponer un modelo arquitectural para aplicaciones móviles. Debido a esto, especializo el proceso de la ingeniería de dominio en su disciplina análisis del dominio presentado en ***InDoCas***, aplicado al dominio de aprendizaje de idiomas asistido por móviles.

Como resultado de su investigación, indica que los sistemas de software están sujetos a la evolución, a las actualizaciones dinámicas debido a cambios por el cliente, contexto de uso o tecnología, además encontró que apoyado en el proceso para la ingeniería del dominio basado en calidad de software ***InDoCas***, es posible lograr una arquitectura base con calidad para una familia de productos.

El aporte de esta investigación viene dado por el hallazgo encontrado por la investigadora garantizando que el uso de ***InDocas*** es útil para la definición de la familia de productos. De allí, que se utilizará ***InDoCas*** como proceso desarrollo para la ingeniería de dominio.

No obstante, esta investigación se limita a proponer un modelo para aplicaciones móviles basado en líneas de productos de software dinámicas, mientras que el objeto de estudio de esta investigación trata sobre ese el enfoque de líneas de producción de software pero en aplicaciones RESTful.

Por otro lado, Schreier (2011) en su investigación Modelando Aplicaciones RESTful, propone un metamodelo para la construcción de aplicaciones RESTful, con el objetivo apoyar el proceso de desarrollo en las fases de análisis y diseño de aplicaciones RESTful.

El metamodelo para REST está divido en modelos estructurales y de comportamiento. Primero describe los tipos de recursos posibles (ver tabla 1), sus atributos y relaciones, así como su interfaz y las representaciones, este último ofrece la posibilidad de describir el comportamiento con maquinas de estado. Asimismo, utilizó el metamodelo, tomando como caso de estudio una Web Álbum, como ejemplo de aplicación RESTful.

Tabla 1. Tipo de Recursos

Tipo de Recurso	Descripción
Recurso principal	Conceptos básicos del dominio de modelado, por ejemplo, fotos y álbumes.
Subrecursos	Parte de otro recurso, que también deben ser direccionados
Lista de recursos	Lista de todos los recursos concretos de un recurso principal
Filtros de los recursos	Lista de recursos concretos con las propiedades deseadas
Proyección de recursos	Contiene sólo un subconjunto de atributos de otro recurso
Agregación de los recursos	Atributos agregados de diferentes recursos para reducir la cantidad interacción
Paginación de los recursos	Divide grandes recursos en diferentes páginas
Actividad de los recursos	Representa un solo paso de un flujo de trabajo

Fuente: Adaptado de Schreier(2011).

Como resultado, en primer lugar, se obtiene una primera versión del metamodelo el cual ofrece un vocabulario para REST en la práctica y sirve de base para el desarrollo dirigido por modelo. En segundo lugar, se detectan sus futuras mejoras, entre ellas: tomar en cuenta aspectos de autenticación, aplicar diferentes casos de estudios para obtener una mayor validación del metamodelo.

Esta investigación, se considera que tiene un aporte para el objeto de estudio, puesto que nos proporciona una herramienta (metamodelo) para modelar aplicaciones RESTful, comprendiendo las etapas de análisis y diseño del proceso de desarrollo de software.

Sin embargo, se observa que el metamodelo está limitado a aplicaciones RESTful simple sin tomar en cuenta la creación de Líneas de Producción para este tipo de software.

Bases Teóricas

El basamento teórico de la presente investigación, consiste en definir claramente los elementos que soportara la investigación. Se aborda la Integración de Aplicaciones Empresariales (IAE), Modelos de Integración de IAE, Reutilización del Software, Ingeniería de Dominio, Ingeniería de Aplicación, Líneas de Producción de Software, REST y Aplicaciones RESTful.

Integración de Aplicaciones Empresariales

La Integración de Aplicaciones Empresariales (IAE),representa el siguiente paso en la evolución de la tecnología que comenzó con la automatización de procesos aislados y que ha llegado afectar la manera como se llevan a cabo los negocios dentro una empresa.

Linthicum (1999), define IAE como el intercambio sin restricciones de datos y procesos de negocio entre las aplicaciones conectadas y fuentes de datos de la empresa. Por otro lado, Ruh et al. (2001) describe IAE como la creación de nuevas soluciones estratégicas de negocios mediante la combinación de la funcionalidad de

las aplicaciones existentes de una empresa, paquetes de aplicaciones comerciales y código nuevo utilizado como un middleware común.

Según Sharma, Stearns y Ng (2001), define la integración de aplicaciones empresariales como la integración de aplicaciones y fuentes de datos empresariales de manera que pueden compartir fácilmente los procesos empresariales y los datos. La integración de las aplicaciones y fuentes de datos debe llevarse a cabo sin necesidad de cambios importantes en las aplicaciones existentes y los datos.

Más tarde, Brown (2000), conceptualiza a EAI como el proceso de colocar hardware, software y procesos de negocios en el contexto de manera que cuando se combinan las interfaces entre los componentes es transparente, la información puede ser fácilmente compartida y los sistemas que trabajan juntos pueden lograr sinergias.

Modelos de Integración de IAE

El Modelado de Integración (MI) ofrece un núcleo de modelos flexibles los cuales fueron desarrollados y descubiertos en proyectos relacionados con la integración. Cada uno de estos modelos actúa como una plantilla o patrón, que representa una dinámica particular que se puede aprovechar para ofrecer una mejor integración. Las técnicas de MI introducen un núcleo de modelos flexibles que son aplicables en todas las industrias. Cada uno de estos modelos actúa como una plantilla, que representa una dinámica particular, que se puede aprovechar para ofrecer una mejor integración, Brown (2000).

Los Modelos de integración se producen en la intersección donde se superponen el modelado de negocios y técnicas, y se desarrolla durante todos los niveles y puntos de vista, siempre que el pegamento que une las piezas haya sido separado a través del análisis. Mientras que los modelos técnicos son analíticos (es decir, separación de los componentes), y los modelos de negocio tienden a ser orientado a la tarea (hacer el trabajo), los modelos de integración se orientan a aclarar el contexto. Capturan múltiples perspectivas y puede funcionar en más de un nivel a la vez. Esta explicación se observa con mayor detalle en la figura 1.

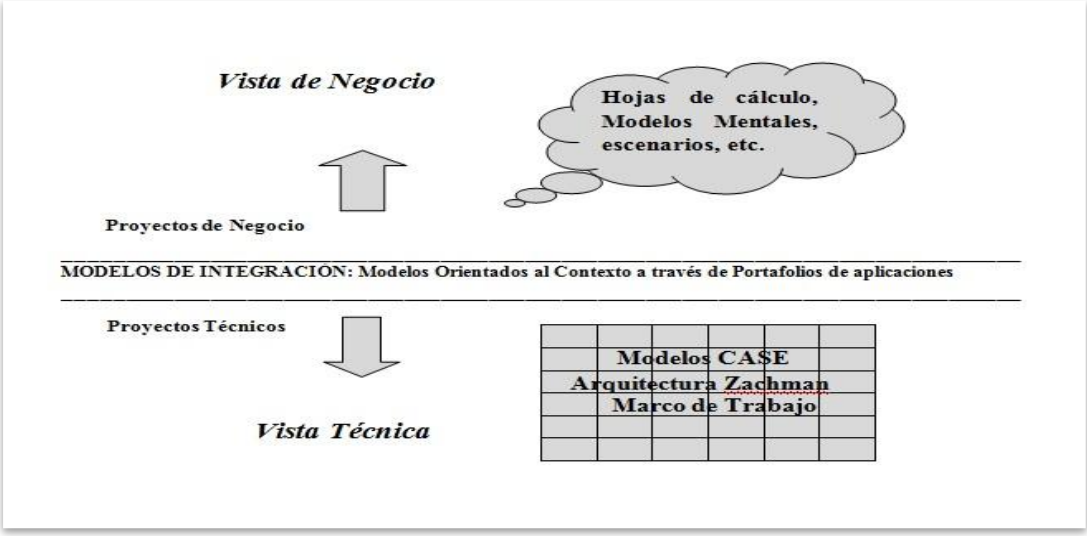


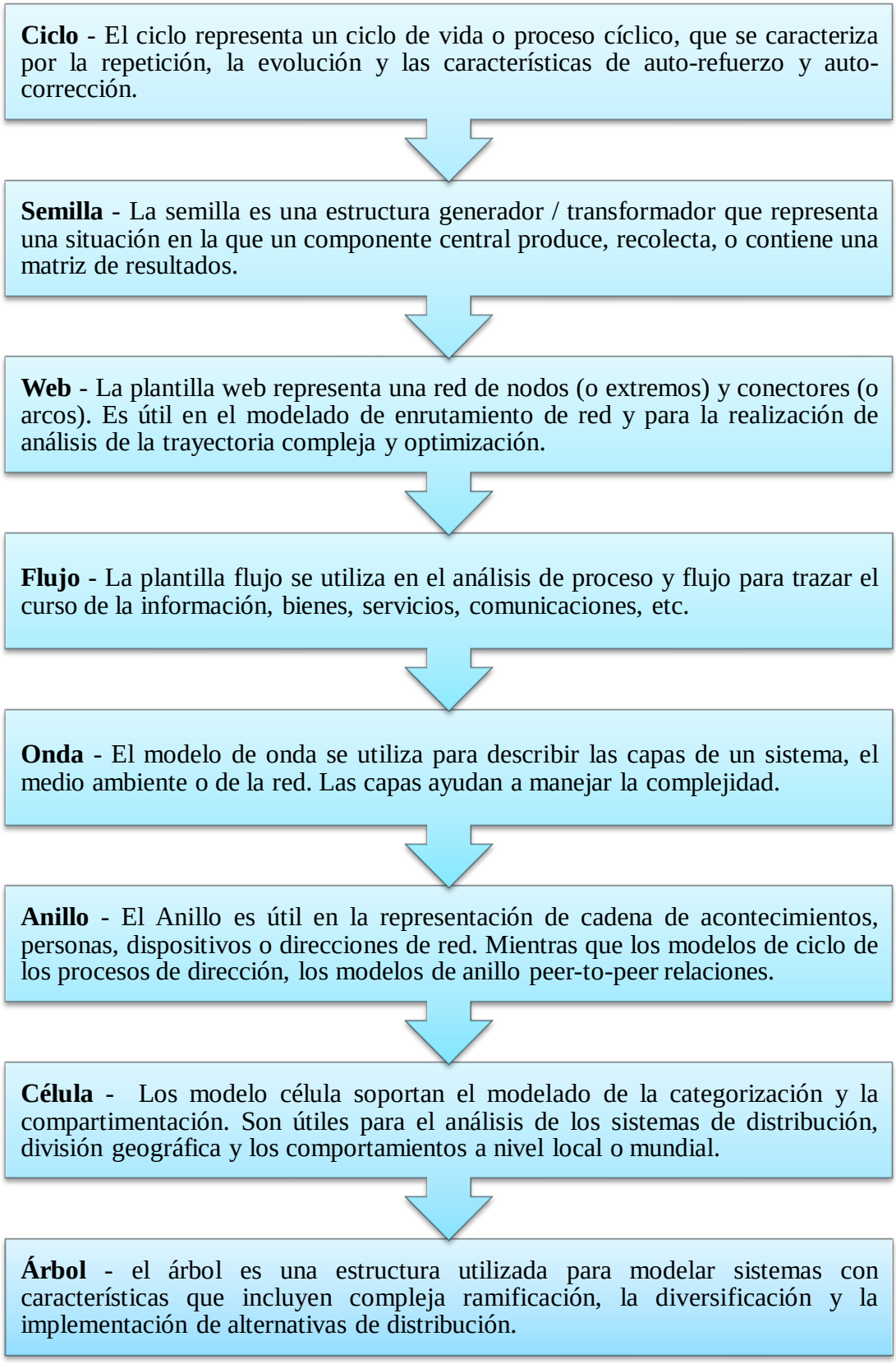
Figura 1. Modelado de Integración. Fuente: Adaptado de Brown (2000).

Entre las características principales de los modelos de integración tenemos:

- **Aclaran el Contexto**
  Los Modelos de Integración ayudan a colocar la tecnología en el fondo y llevan al negocio a un primer plano. Logran esto mediante el tratamiento de los sistemas como contexto, y se centra en la dinámica de las interacciones de los sistemas. En general, ¿Una actividad de negocio específica tiene que seguir un ciclo? ¿Es necesaria una red de conexiones? O ¿Contiene capas de complejidad? ¿Cómo funciona? Preguntas como éstas ayudarán a definir cuales modelos de integración seleccionar.
- **Capturan múltiples perspectivas**
  Debido a que son un conjunto flexible de plantillas de modelado, se pueden aplicar en una variedad de entornos diferentes. Se puede modelar la dinámica de un escenario de un cliente, mostrando lo que es importante acerca de cómo nuestros clientes pueden interactuar con nuestro sitio web, por ejemplo. O pueden aplicarse al entorno de operaciones informático, que representa una serie de sistemas informáticos y dispositivos que admiten la interacción del cliente.

- **Funcionan en más de un nivel**
  Los modelos de integración pueden ser utilizados para centrarse en los detalles significativos de un proceso o sistema, o para tender un puente entre los sistemas a un nivel muy alto. De nuevo, esto es porque se concentran en la dinámica de una situación, no en la implementación de tecnología. El nivel que se captura depende de la perspectiva que se describe.
- **Cruce de Portafolios de Aplicación**
  Algunas compañías colocan todos los sistemas de aplicación que se utilizan en un área de negocio (por ejemplo, mercadeo, finanzas, operaciones, ventas, etc.) en un "portafolio", o conjunto de sistemas de aplicación, para cada área de negocio. Las aplicaciones dentro de un portafolio son más propensas a estar bien integradas. Las que están en portafolios por separado rara vez son así. Una de las funciones de los modelos de integración es proporcionar una manera de centrarse en los aspectos de la actividad empresarial que tienen lugar a través de los portafolios de aplicaciones.
- **Proporcionan espacio para la infraestructura**
  Cuando se cruzan las líneas entre los portafolios de aplicaciones, los modelos de integración pueden destacar las oportunidades para las economías de escala o la reutilización de una variedad de medios técnicos. También pueden proporcionar marcadores de posición para reunir una infraestructura.

Con base en las experiencias de muchos proyectos de software, Brown (2000), propone un catálogo de modelos de integración el cual ofrece una síntesis de las mejores prácticas en el campo de la integración. Los modelos representan los patrones encontrados útil en proyectos de diferentes industrias. A continuación se presenta el catalogo de Modelos de Integración:


Ciclo - El ciclo representa un ciclo de vida o proceso cíclico, que se caracteriza por la repetición, la evolución y las características de auto-refuerzo y auto-corrección.
Semilla - La semilla es una estructura generador / transformador que representa una situación en la que un componente central produce, recolecta, o contiene una matriz de resultados.
Web - La plantilla web representa una red de nodos (o extremos) y conectores (o arcos). Es útil en el modelado de enrutamiento de red y para la realización de análisis de la trayectoria compleja y optimización.
Flujo - La plantilla flujo se utiliza en el análisis de proceso y flujo para trazar el curso de la información, bienes, servicios, comunicaciones, etc.
Onda - El modelo de onda se utiliza para describir las capas de un sistema, el medio ambiente o de la red. Las capas ayudan a manejar la complejidad.
Anillo - El Anillo es útil en la representación de cadena de acontecimientos, personas, dispositivos o direcciones de red. Mientras que los modelos de ciclo de los procesos de dirección, los modelos de anillo peer-to-peer relaciones.
Célula - Los modelo célula soportan el modelado de la categorización y la compartimentación. Son útiles para el análisis de los sistemas de distribución, división geográfica y los comportamientos a nivel local o mundial.
Árbol - el árbol es una estructura utilizada para modelar sistemas con características que incluyen compleja ramificación, la diversificación y la implementación de alternativas de distribución.

Reutilización de Software

El proceso de Diseño en la mayoría de las disciplinas de ingeniería se basa en la reutilización de sistemas o componentes existentes. La ingeniería de software basada en reutilización es una estrategia comparable en la que el proceso de desarrollo es adaptado a la reutilización de software existente. Las unidades de software que se utilizan pueden ser de tamaños diferentes por ejemplo: Reutilización de sistemas de aplicaciones, Reutilización de Componentes y Reutilización de Objetos y funciones, Sommerville (2006).

La ventaja obvia de la reutilización de software es que los costes totales del desarrollo deberían reducirse. Se necesita especificar, diseñar, implementar y validar menos componentes de software. Sin embargo, la reducción de los costes es solo una ventaja de la reutilización. En la tabla 2, se muestras otras ventajas de la reutilización de activos de software.

Tabla 2. Beneficios de la reutilización de software.

Beneficio	Explicación
Incremento de la confiabilidad	El software reutilizado, que ha sido probado en sistemas en funcionamiento, debería ser más confiable que el software nuevo debido a que sus fallos en la implementación y el diseño ya han sido encontrados y reparados.
Reducción del riesgo del proceso	El costo de software existente ya es conocido, mientras que el coste de desarrollo es siempre cuestionable. Éste es un factor importante para la gestión de proyectos debido a que reduce el margen de error en la estimación de costes del proyecto. Esto es particularmente cierto cuando se reutilizan componentes de software relativamente grandes tales como subsistemas.
Uso efectivo de especialistas	En lugar de hacer el mismo trabajo una y otra vez, estos especialistas de aplicaciones pueden desarrollar software reutilizable que encapsule su conocimiento.
Cumplimiento de estándares	Algunos estándares, tales como los estándares de interfaz de usuario, pueden implementarse como un conjunto de componentes reutilizables

	estándar. Por ejemplo, si los menús en una interfaz de usuario se implementan usando componentes reutilizables, todas las aplicaciones presentan el mismo formato de menú a los usuarios. El uso de interfaces de usuario estándares mejora la confiabilidad debido a que es menos probable que los usuarios cometan errores cuando se encuentran con una interfaz similar.
Desarrollo acelerado	Sacar al mercado un sistema tan pronto como sea posible es a menudo más importante que los costes totales de desarrollo. La reutilización del software puede acelerar la producción del sistema debido a que se reducen los tiempos de desarrollo y validación.

Fuente: Sommerville (2006).

Durante los veinte últimos años, se han desarrollado muchas técnicas para soportar la reutilización del software. Estas explotan el hecho de que los sistemas del mismo dominio de aplicación son similares y tienen potencial para la reutilización. La tabla 3, muestra varias formas de soportar la reutilización del software.

Tabla 3. Aproximaciones que soportan la reutilización del software.

Aproximación	Descripción
Patrones de diseño	Las abstracciones genéricas similares entre aplicaciones se representan como patrones de diseño que muestran los objetos abstractos y concretos y sus interacciones.
Desarrollo basado en Componentes	Los sistemas se desarrollan integrando componentes (colecciones de objetos) que cumplen los estándares de modelado de componentes.
Marcos de Aplicaciones	Las colecciones de clases concretas y abstractas pueden adaptarse y extenderse para crear sistemas de aplicaciones.
Envoltura de sistemas heredados	Sistemas heredado que pueden ser envueltos definiendo un conjunto de interfaces y proporcionando acceso a estos sistemas heredados a través de estas interfaces.
Sistemas orientas a Servicios	Los sistemas se desarrollan enlazando

	servicios compartidos, que pueden ser proporcionados de forma externa.
Líneas de Productos de Aplicaciones	Un tipo de aplicación se generaliza alrededor de una arquitectura común para que pueda ser adaptada para diferentes clientes.
Integración COTS	Los sistemas se desarrollan integrando sistemas de aplicaciones existentes.
Aplicaciones Verticales Configurables	Un sistema genérico se diseña para que pueda configurarse para las necesidades de los clientes de sistemas particulares.
Librerías de Programas	Están disponibles para reutilización de librerías de funciones y de clases que implementan abstracciones comúnmente usadas.
Generadores de Programas	Un sistema generador incluye conocimiento de un tipo de aplicación particular y puede generar sistemas o fragmentos de un sistema en ese dominio.
Desarrollo del software orientado a aspectos	Componentes compartidos entrelazados en una aplicación en diferentes lugares cuando se compila el programa.

Fuente: Sommerville (2006).

Líneas de Producción de Software

Las líneas de producción de Software nace de la necesidad de reutilización del software proceso del cual se habló anteriormente. La reutilización está sustentada en el desarrollo basado en componentes. Northrop (2008) señala que, una línea de producto de software es un conjunto de sistemas que comparten un software común, logrando un conjunto de características que satisfacen las necesidades específicas de un determinado segmento del mercado o de la misión y que está desarrollado a partir de un conjunto común de componentes de una manera prescrita. En la figura 2, podemos observar el funcionamiento de las líneas de Productos de Software, estas permiten tomar ventaja económica de lo común y de la variación ligada.

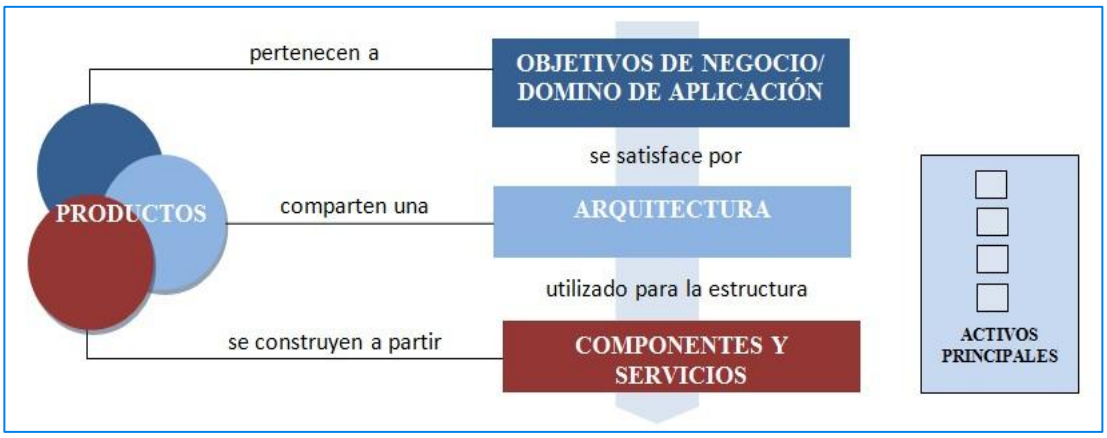


Figura 2. Funcionamiento de las Líneas de Producción de Software. Fuente: Adaptado de Northrop (2008).

En este sentido, las líneas de productos ayudan a amortizar la inversión en lo que se refiere a activos principales como:

- Requisitos y Análisis de Requisitos.
- Modelo del Dominio.
- Diseño y Arquitectura de Software.
- Ingeniería de Rendimiento.
- Documentación.
- Planes de Prueba, Casos de Prueba y Datos de Prueba.
- Personas: sus Conocimientos y Habilidades.
- Procesos, Métodos y Herramientas.
- Presupuestos, Cronogramas y Planes de Trabajo.
- Componentes y Servicios.

En otras palabras, una línea de producto es igual a la reutilización estratégica, lo que se traduce en más beneficios. Northrop (2008), menciona algunos contextos donde las líneas de producción de software han sido exitosas, entre ellos: teléfonos móviles, sistemas de aviónica, sistemas de conocimiento, buscapersonas, sistemas de control de motores, dispositivos de almacenamiento masivo, sistemas de facturación, sistemas pequeños basados en la web, impresoras, productos electrónicos, adquisición

de sistemas de gestión de la empresa, sistemas financieros y fiscales, dispositivos médicos, piscifactoría de software de gestión, etc.

Por otro lado Krueger (2006), señala que una línea de producción de software se refiere a técnicas de ingeniería para crear un portafolio de sistemas de software similares, a partir de un conjunto compartido de activos de software, usando un medio común de producción.

En este sentido, una línea de producción de software consta de cuatro (4) conceptos básicos (véase figura 3), Montilva (2006):

1. ***Entrada***: Activos de Software

   Es la colección de partes de software (requisitos, diseños, componentes, casos de prueba, etc.) que se configuran y componen de una manera prescrita para producir los productos de la línea.

2. ***Control***: Modelos de Decisión y Decisiones de producto

   Los Modelos de Decisión describen los aspectos variables y opcionales de los productos de la línea. Cada producto de la línea es definido por un conjunto de decisiones (decisiones del producto).

3. ***Proceso***: Producción del Producto

   Se establecen los mecanismos o pasos para componer y configurar productos a partir de los activos de entrada. Las decisiones del producto se usan para determinar que activos de entrada utilizar y como configurar los puntos de variación de esos activos.

4. ***Salida***: Productos de Software

   Es el conjunto de todos los productos que pueden o son producidos por la línea de productos.

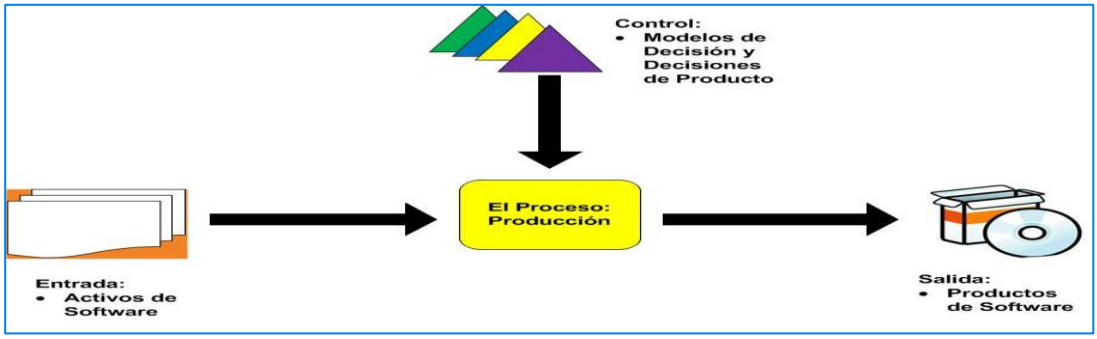


Figura 3. Modelo Básico de una Línea de Producción de Software. Fuente: Adaptado de Montilva (2006).

Estos conceptos ilustran los principales objetivos de las líneas de producción de software que es capitalizar lo común y gestionar las variaciones a fin de reducir el tiempo, esfuerzo, costo y la complejidad de crear y mantener una línea de producción de los sistemas de software similares.

A continuación los beneficios de las líneas de producción de software:

- Beneficios tácticos de ingeniería:
  - Reducción en el tiempo promedio de creación y entrega de nuevos productos, número promedio de defectos por producto, esfuerzo promedio requerido para desarrollar y mantener los productos y costo promedio de producción de los productos.
  - Incremento en el número total de productos que pueden ser efectivamente desplegados y mantenidos.
- Beneficios tácticos de negocios:
  - Reducción en el tiempo de entrega (time-to-market) y el tiempo de retorno (time-to-revenue) de nuevos productos y riesgos en la entrega de productos.
  - Mejoras en el valor competitivo del producto y en la reputación de la empresa.
  - Mayores márgenes de ganancia.
  - Mejor calidad de los productos.

- Mayor escalabilidad del modelo de negocios en términos de productos, mercados y agilidad para expandir el negocio a nuevos mercados.

En este orden de ideas, Díaz y Trujillo (2010), señalan que las líneas de producción de software constan de aspectos metodológicos, como: las estrategias y los procesos.

- ***Estrategias***: el proceso de desarrollo de la LPS depende, entre otros muchos factores, ámbito de la LPS. Es fundamental saber acotar la familia de productos que serán objeto línea. En general, existe una tendencia a generalizar en exceso cuando se está desarrollando software re-usable, considerando casos poco probables. Es la filosofía "si acaso". Sin embargo, esta excesiva generalización, si se repite con distintas "características" compatibles entre sí, puede dar lugar a una explosión combinatoria. Esto favorece la variabilidad, pero incurre en costes de pruebas, documentación y desarrollo adicionales, que pueden finalmente no rentabilizarse para casos poco probables. La pregunta es entonces cuánta variabilidad soportar, qué grado de re-uso van soportar los elementos comunes (activos principales), es decir, delimitar el ámbito de la línea producto.
- ***Procesos:*** Un aspecto central compartido por las distintas metodologías de desarrollo de LPS división de los procesos de ingeniería en dos equipos de trabajo El primer equipo se encarga de la Ingeniería de Dominio, el cual es definido por Clements (2001) como core asset development. Este equipo es responsable de desarrollar los elementos comunes al dominio: estudiar el dominio, definir su alcance (requisitos) dentro mercado objetivo de la LPS, definir las features, implementar los core assets reutilizables y su mecanismo de variabilidad, y establecer cómo es el plan de producción. Y el segundo equipo se encarga de la Ingeniería de Producto definido por Clements (2001) como product development. Sus cometidos incluyen desarrollar los productos para clientes concretos, a partir de los recursos basados no en los requisitos del dominio, sino requisitos concretos de clientes. Para ello, este segundo

equipo utiliza los recursos creados por el equipo anterior. La pregunta es entonces cuánta variabilidad soportar, qué grado de re-uso van soportar los elementos comunes (activos principales), es decir, delimitar el ámbito de la línea producto.

Ingeniería de Dominio

Czarnecki yEisenecker (2000) define la ingeniería de dominio como la actividad de recolectar, organizar y almacenar la experiencia adquirida en la construcción de sistemas o partes de los sistemas en un dominio particular en forma de activos reutilizables (es decir, workproducts reutilizables), así como proporcionar un medio adecuado para la reutilización de estos activos (es decir, recuperación, clasificación, difusión, adaptación, montaje, etc.) en la construcción de nuevos sistemas.

Por otro lado, Pressman (2005), señala que la ingeniería de dominio tiene como finalidad identificar, construir, catalogar y diseminar un conjunto de componentes de software que sean aplicables para el software existente y futuro en un dominio de aplicación particular.

Asimismo, Pohl, Böckle y Van der Linden (2005) resaltan que en la ingeniería de dominio el interés común y la variabilidad de un conjunto de aplicaciones de línea de productos previstos son identificados, documentados, y producidos. La variabilidad está explícitamente documentada en el modelo de variabilidad ortogonal a fin de facilitar la reutilización de los activos de la línea de productos durante la ingeniería de aplicación.

Según pohl et al. (2005), la ingeniería de Dominio, está comprendida en varios sub- procesos los cuales se destacan en la figura 4.

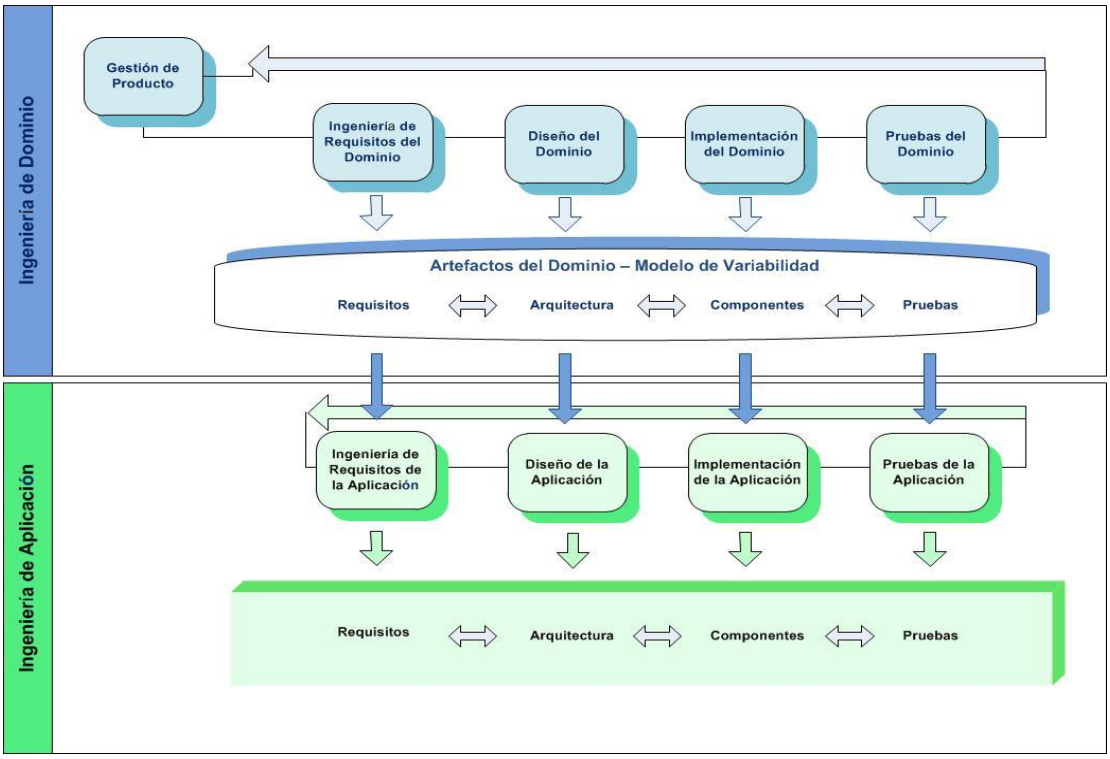


Figura 4. Ingeniería de Dominio y Aplicación. Fuente: Adaptado de Pohl et al. (2005).

A continuación se ofrece una breve descripción de cada uno de los sub-procesos de la ingeniería de dominio:

- **Gestión del Producto**: El objetivo de la gestión de productos es hacer una importante contribución al éxito empresarial mediante la integración del desarrollo, producción y comercialización de productos que satisfagan las necesidades de los clientes. En base a los objetivos globales y muy abstractos de la compañía, las decisiones estratégicas tienen que ser hechas. La ingeniería de requerimientos del Dominio y aplicación tienen que cumplir con las características especificadas en la hoja de ruta. Mientas La ingeniería de requerimientos del Dominio proporciona artefactos reutilizables de los requerimientos, la ingeniería de la aplicación crea los artefactos de los requerimientos para aplicaciones específicas, previstas en la hoja de ruta del producto. Observe en la figura 5, los flujos de información entre la gestión de productos básicos y sus subprocesos relacionados.

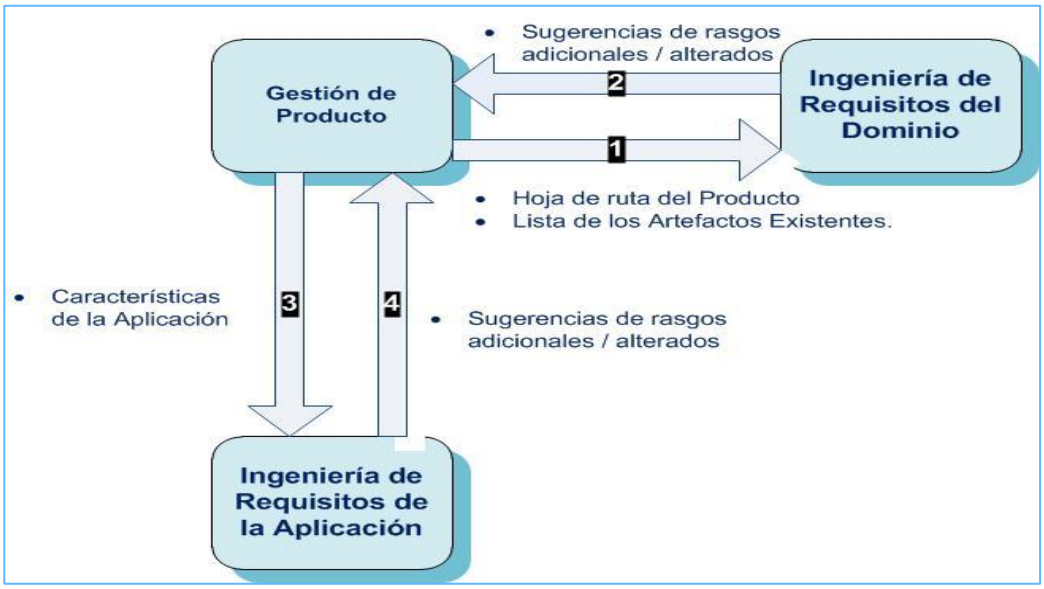


Figura 5. Los flujos de información entre la Gestión del producto y otros sub-procesos. Fuente: Adaptado de Pohl et al. (2005).

- **Ingeniería de Requisitos del Dominio**: abarca todas las actividades para obtener y documentar los requisitos comunes y variables de la línea de productos.La ingeniería de requisitos del dominio tiene que cumplir con la especificación de las características principales de la línea de productos proporcionados por la gestión del producto. En base a estas características, se crea detallados requisitos comunes y variables suficientes para orientar el diseño de dominio. Además, los requisitos de ingeniería de dominio proporciona la entrada para el sub-proceso ingeniería de requisitos de la aplicación, que se ocupa de crear objetos específicos de los requisitos de la aplicación (Ver en la figura 6, los flujos de información entre la ingeniería de requisitos del Dominio y otros subprocesos).

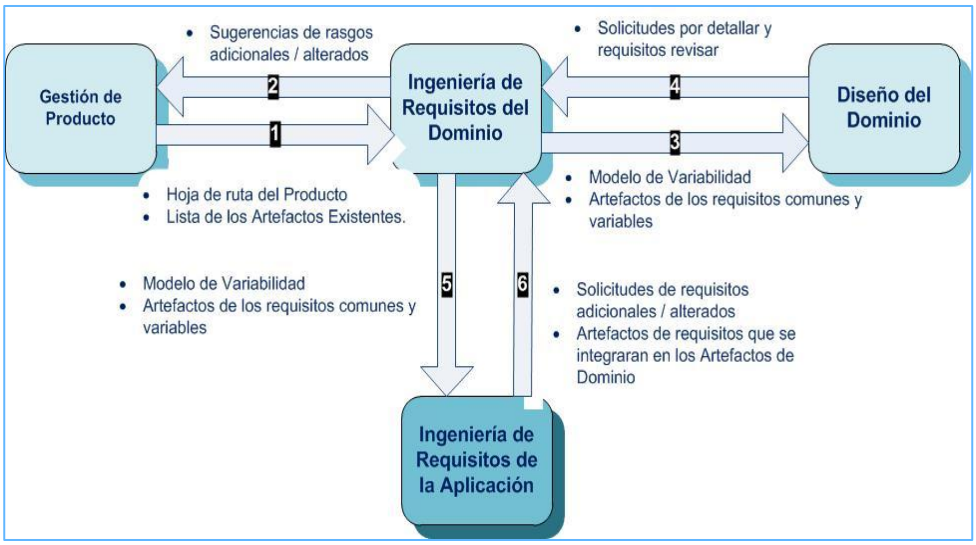


Figura 6. Flujos de información entre la ingeniería de requisitos del Dominio y otros subprocesos. Fuente: Adaptado de Pohl et al. (2005).

- **Diseño del Dominio**: abarca todas las actividades para la definición de la arquitectura de referencia de la línea de productos. La arquitectura de referencia proporciona una estructura común de alto nivel para todas las aplicaciones de la línea de productos. La arquitectura de referencia tiene que ser flexibles, capaces de evolucionar y mantener. Su diseño incorpora el alojamiento de las necesidades futuras y la tecnología. En particular, la arquitectura de referencia cambia con el tiempo. La arquitectura de referencia incluye una estructura variable que es la base de las estructuras de todas las aplicaciones. Además, la arquitectura de referencia proporciona la textura de los componentes reutilizables y las interfaces. Junto con la arquitectura de referencia, se transmite una selección de objetos reutilizables de dominio que la realización de dominio debe construir. La selección de objetos abarca los componentes reutilizables y las interfaces, así como su trazabilidad a componentes específicos de la aplicación y las interfaces (Véase en la figura 7, el flujo de información entre el diseño del dominio y otros subprocesos).

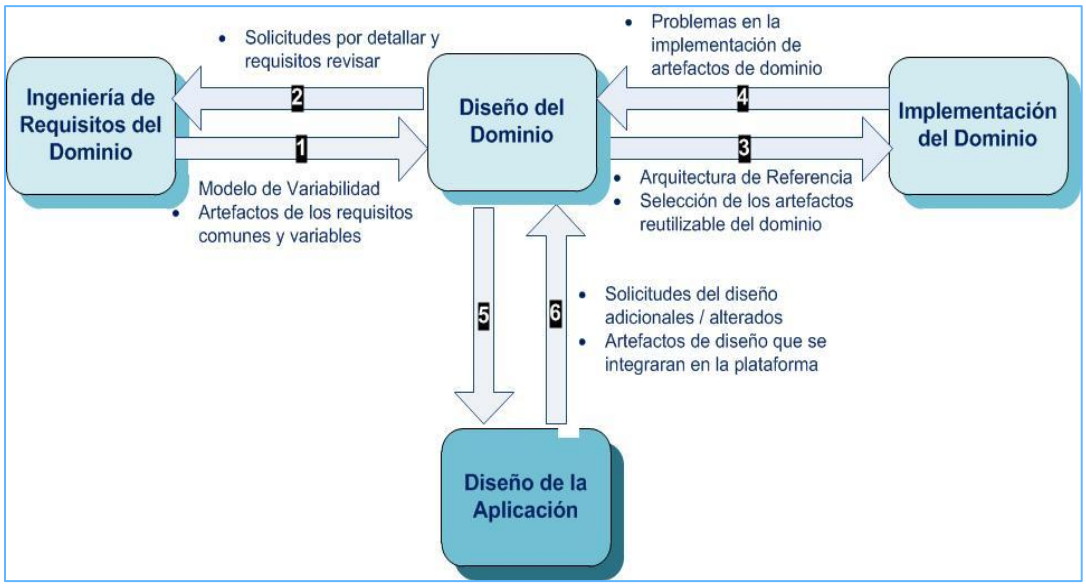


Figura 7. Flujo de información entre el diseño del dominio y otros subprocesos. Fuente: Adaptado de Pohl et al. (2005).

- **Implementación del Dominio**: se refiere al diseño detallado y la implementación de componentes de software reutilizables. En particular, el diseño de interfaces es crucial, ya que son la base para la variabilidad de arquitectura basada en las variantes de configuración. Como componentes diferentes que proporcionan o requieren una única interfaz, puede haber muchos interesados en el diseño de componentes, que tienen su propio interés en la mudanza del nivel de abstracción. El desarrollador del componente tiene diferentes mecanismos disponibles para aplicar la variabilidad de los componentes. La elección se rige por la textura arquitectónica, permitiendo así un alto grado de uniformidad a ser logrados en la aplicación. La variabilidad tiene que ser presentado al desarrollador de la aplicación con el fin de permitir la selección de las variantes adecuadas. Por ejemplo, esto puede hacerse mediante la relación de parámetros de componentes para el modelo de variabilidad. A continuación se muestra en la figura 8, el flujo de información entre la implementación del dominio y otros subprocesos.

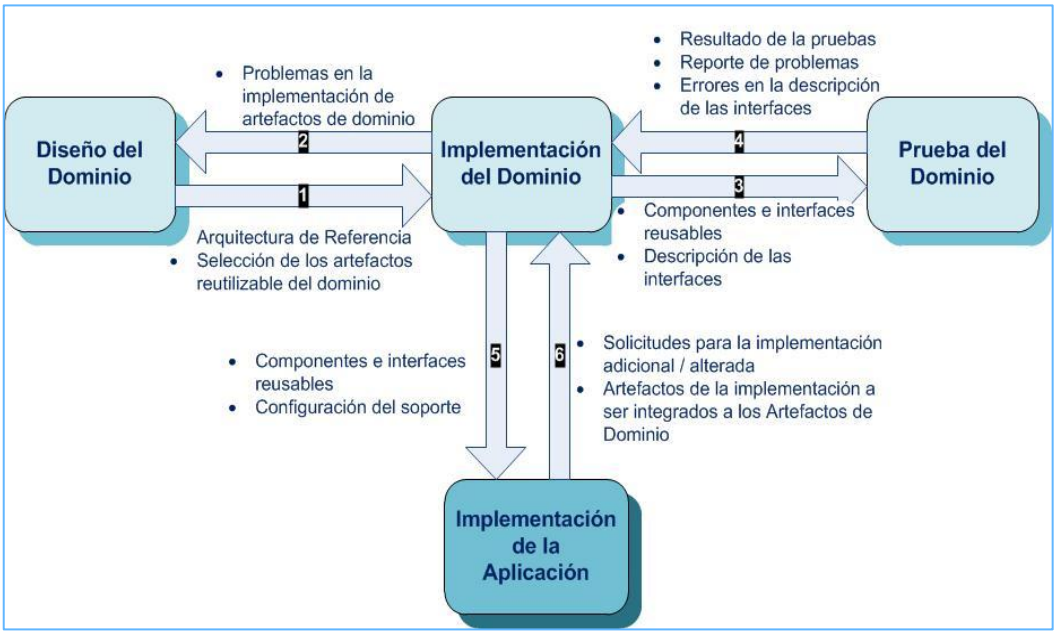


Figura 8. Flujo de información entre la implementación del dominio y otros subprocesos. Fuente: Adaptado de Pohl et al. (2005).

- **Pruebas del Dominio**: este proceso es el responsable de la validación y verificación de componentes reutilizables. Aquí se pone a prueba los componentes en contra las especificaciones, es decir, requisitos, la arquitectura y los artefactos de diseño. Además, en las pruebas de dominio se desarrolla artefactos de pruebas reutilizables para reducir el esfuerzo de pruebas de aplicaciones. Se recomienda dos estrategias que se puede aplicar en combinación. La primera estrategia, la estrategia de aplicación simple (SAS), implica la construcción de una o más aplicaciones de ejemplo. Esto permite una validación inicial de la línea de productos de software. Además, la prueba se puede realizar de la misma manera como en la ingeniería de un solo sistema. La segunda estrategia, la estrategia de reusabilidad y comunalidad (CRS), realiza pruebas de los artefactos comunes en la ingeniería de dominio y proporciona artefactos variables de prueba para su reutilización en las pruebas de aplicaciones. La reutilización de estos artefactos reduce el esfuerzo durante las pruebas de aplicaciones.Las interrelaciones esenciales entre las pruebas de dominio y los otros sub-procesos de la ingeniería de dominio se muestran en la figura 9. Los

resultados producidos durante la ingeniería de requerimientos del dominio se utilizan como entrada para la prueba del dominio del sistema. La arquitectura resultante del diseño de dominio es necesaria para la prueba de integración de dominio, y los componentes producidos durante la implementación de dominio son validados en la prueba de la unidad de dominio.

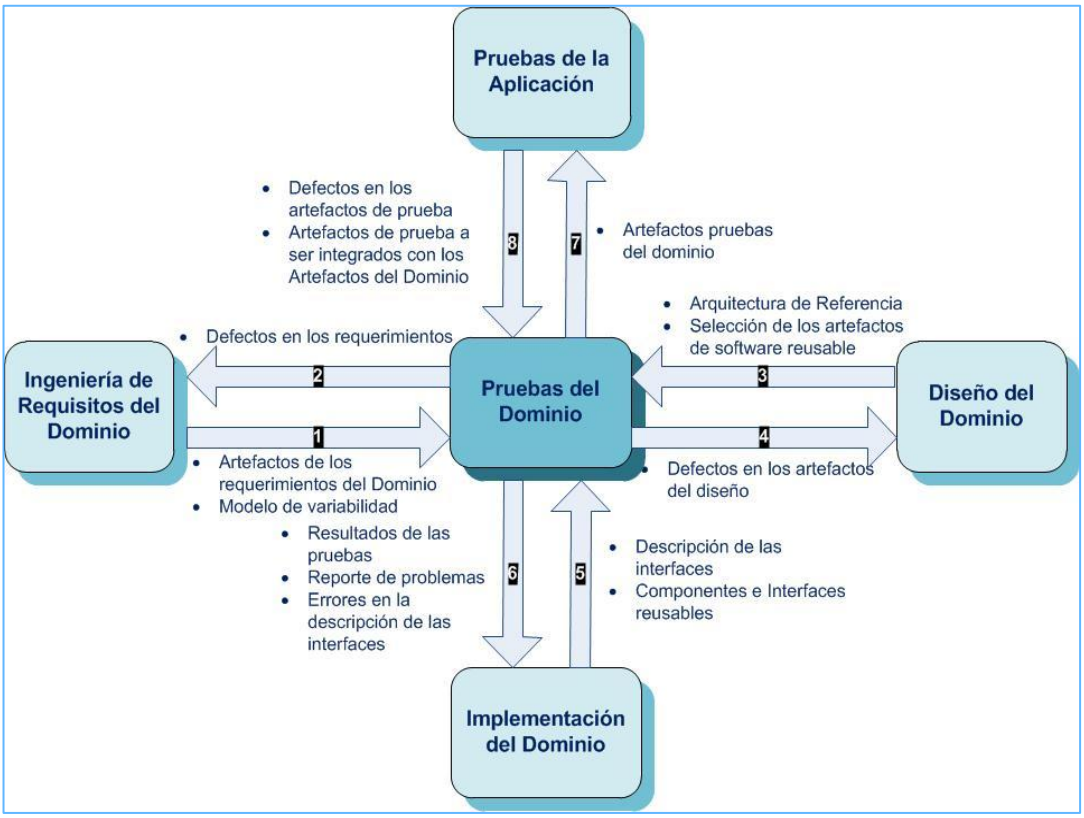


Figura 9. Flujo de información entre las Pruebas del dominio y otros subprocesos. Fuente: Adaptado de Pohl et al. (2005).

Ingeniería de Aplicación

La ingeniería de aplicación tiene como propósito el desarrollo de los productos de LPS a través de la reutilización de activos de software producidos durante la Ingeniería de Dominio, Montilva (2006).

En otras palabras, se enfoca en la construcción o desarrollo de productos individuales, que provienen de una línea de producción y que satisfacen un conjunto de requisitos y limitaciones expresado por un usuario en específico, Pohl et al. (2005).

Por otro lado, Díaz y Trujillo (2010), denominan a la ingeniería de aplicación como ingeniería de producto la cual utiliza la infraestructura creada por el equipo de dominio para construir productos concretos para clientes que lo demanden. En consecuencia, se parte de unos requisitos concretos que deben de estar alineados de alguna forma con los requisitos de la línea de producto, aunque puede haber una pequeña parte nueva que hay que desarrollar para este producto.

Continuando con Pohl y Otros (2005), la ingeniería de la aplicación está compuesta por cuatro (4) sub-procesos, que se encuentran en la figura anterior, estos son:

- **Ingeniería de Requisitos de Aplicación**: Consiste básicamente en la detección de diferencias entre los requisitos de la aplicación y las capacidades disponibles de la plataforma. Esta tiene como entrada los requisitos de la Ingeniería de Dominio y tiene como salida los requisitos para la Ingeniería de Aplicación. Esta fase comprende todas las actividades para el desarrollo de la captura de los requisitos de la aplicación, entre ellas: obtención los requisitos que quieren que cumpla la aplicación, especificación completa los requisitos de la aplicación de una línea de producción en particular, factibilidad, reusabilidad, conocimiento de las capacidades de la línea de producción y la estimación del esfuerzo de la implementación. A continuación, en la figura 10, se observa los flujos de información entre la ingeniería de requisitos de la aplicación y otros subprocesos.

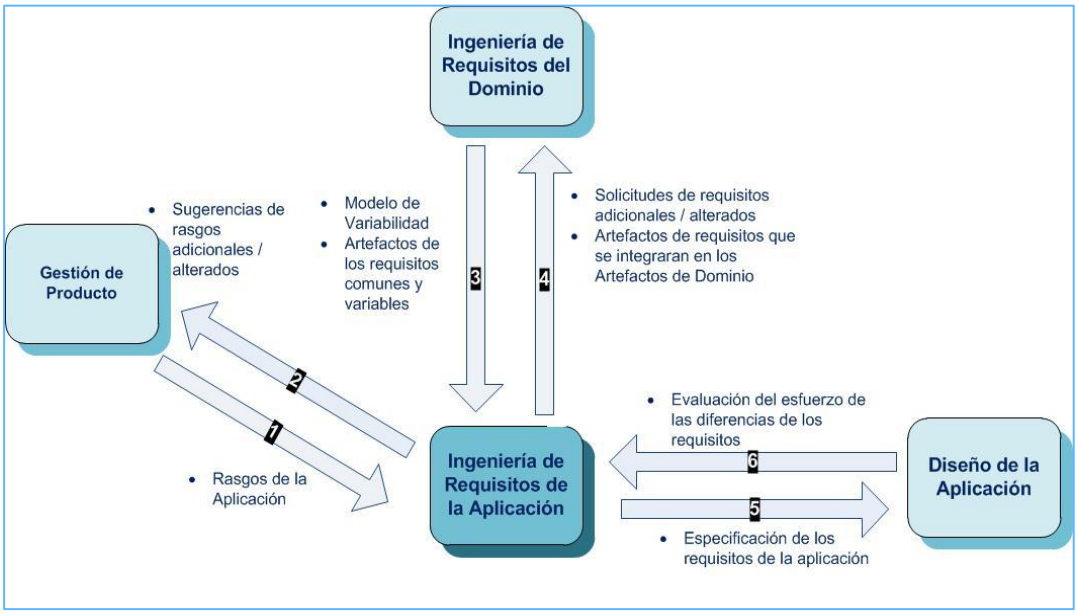


Figura 10. Flujos de información entre la ingeniería de requisitos de la aplicación y otros subprocesos. Fuente: Adaptado de Pohl et al. (2005).

- **Diseño de la Aplicación**: usa como referencia la arquitectura generada en la Ingeniería de Dominio incorporando nuevas adaptaciones específicas en la aplicación. Tiene como objetivo generar la arquitectura de la aplicación. Cabe destacar que, la variabilidad en el sub-proceso de diseño de la aplicación, representa la estructura general del sistema considerado (por ejemplo, los dispositivos de hardware específicos disponibles en el sistema). Adicionalmente, en el diseño de la aplicación se debe evaluar el esfuerzo de realización de cada adaptación necesaria y se pueden rechazar los cambios estructurales que requieren un esfuerzo similar para el desarrollo de la aplicación. Seguidamente, la figura 11 presenta el flujo de información generado en el sub-proceso de diseño de la aplicación.

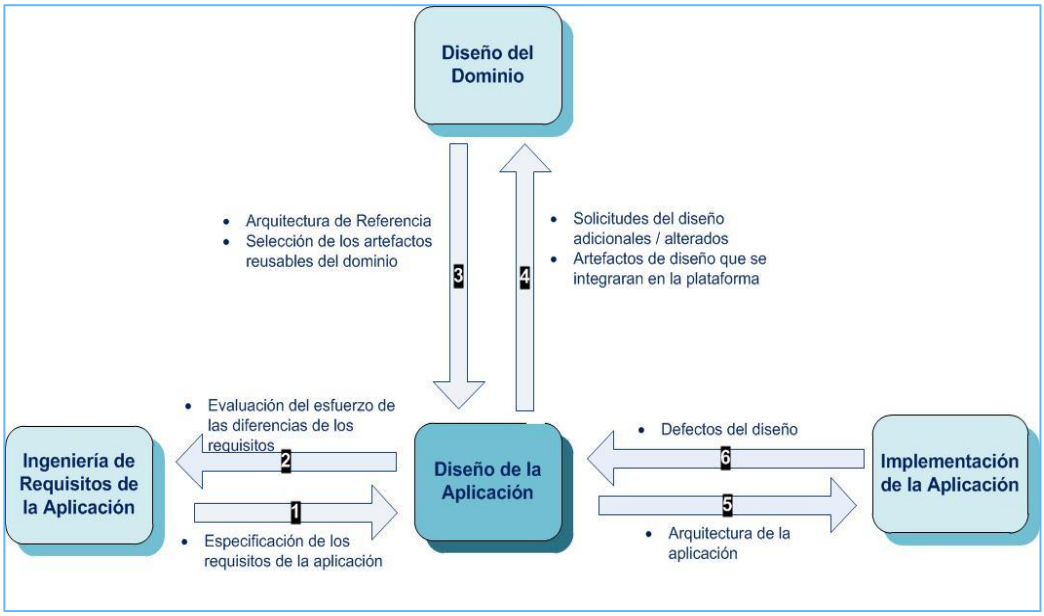


Figura 11. Flujos de información generada en el sub-proceso de diseño de la aplicación. Fuente: Adaptado de Pohl et al. (2005).

- **Implementación de la Aplicación**: la principal tarea en este proceso es la selección y configuración de los componentes de software reutilizables, así como la realización de los activos específicos de la aplicación; ambas actividades son agrupadas para crear la aplicación. En esta fase, la entrada consiste en la arquitectura de la aplicación y en los artefactos reutilizables de la plataforma. La salida consiste en una aplicación ejecutándose en conjunto con los artefactos del diseño especificados. A continuación, en la figura 12 se observa el flujo de información generado en el sub-proceso de implementación de la aplicación.

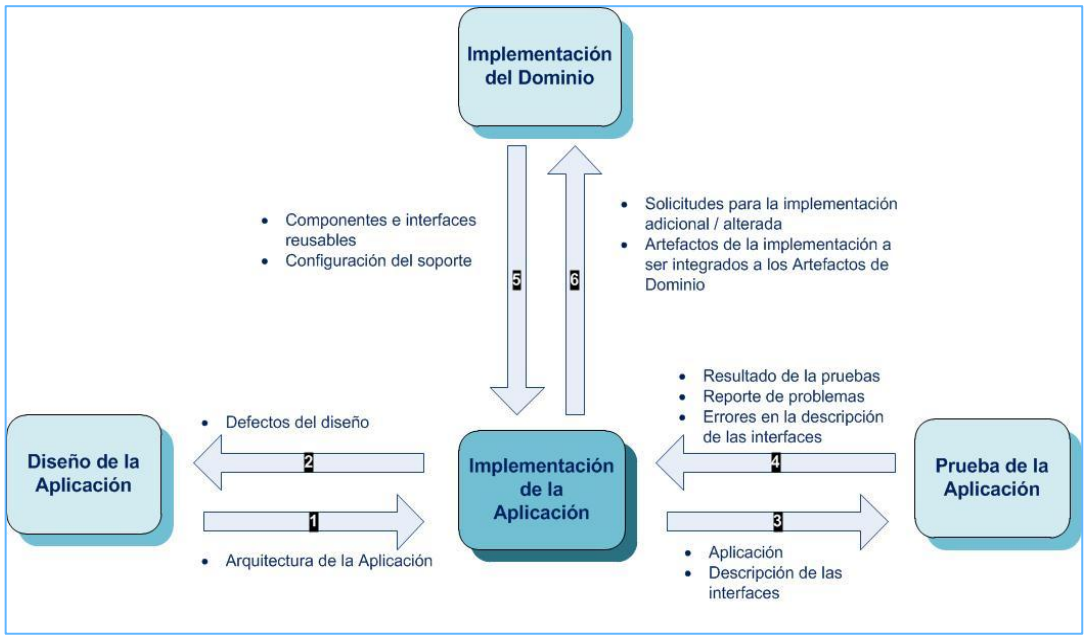


Figura 12. Flujo de información generado en el sub-proceso de implementación de la aplicación. Fuente: Adaptado de Pohl et al. (2005).

- **Pruebas de la Aplicación**: este proceso se basa en alcanzar suficiente calidad en el proceso de validación y verificación de la aplicación. Las pruebas de aplicación complementan las fases de prueba de la Ingeniería de Dominio, reutilizando los artefactos que se utilizaron en las pruebas de dominio. Las pruebas de aplicación ejecutan pruebas adicionales a fin de detectar configuraciones erradas y para asegurarse que exactamente las variaciones especificadas han sido realizadas. Seguidamente, la figura 13 muestra el flujo de información del sub-proceso pruebas de la aplicación.

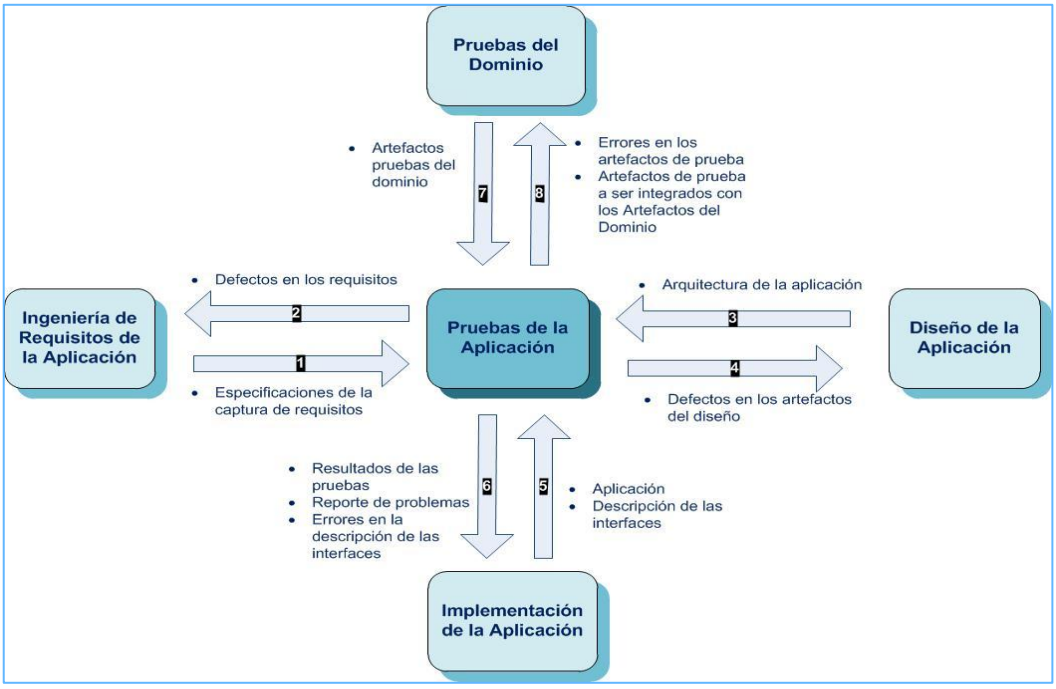


Figura 13. Flujo de información del sub-proceso pruebas de la aplicación. Fuente: Adaptado de Pohl et al. (2005).

SPEM 2

SPEM 2 es un estándar de metamodelado que sirve para representar procesos de ingeniería de softwareentiéndase un Proceso Software (PS) como "Un conjunto coherente de políticas, estructuras organizacionales, tecnologías, procedimientos y artefactos que son necesarios para concebir, desarrollar, instalar y mantener un producto software". Además se enmarca dentro de la Ingeniería de Procesos de Software, en inglés *Software Processes Engineering* (SEP), que es un área nueva de la Ingeniería de Software dedicada a "la definición, implementación, medición y mejora de los procesos de Ingeniería de Software".

SPEM 2 está basado en MOF (Meta Object Facility), un estándar previo de OMG que define una arquitectura de modelado de cuatro niveles conceptuales los cuales se visualiza en la tabla 4, Ruiz y Verdugo (2008). Así, SPEM es a los procesos software

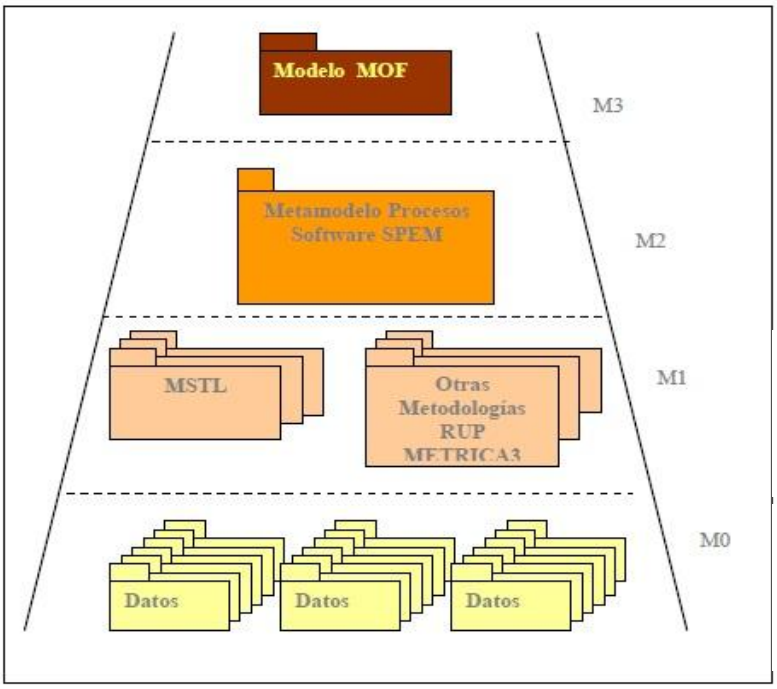
lo mismo que UML es a los sistemas software. UML es un metamodelo que sirve para representar modelos de sistemas software y SPEM es un metamodelo que sirve para representar modelos de procesos software.

Tabla 4. Niveles de modelado de MOF

Nivel	MOF	Ejemplo
M3	Modelo MOF (Meta-metamodelo)	
M2	Meta-modelo	UML, SPEM Entidad-Interrelación
M1	Modelo	Modelo UML
M0	Datos	Datos de un proyecto concreto

Fuente: Ruiz y Verdugo (2008).

Entre los distintos niveles consecutivos existen relaciones de instanciación, de manera que un elemento de nivel N_1 es una instancia de un elemento del nivel superior a N. La figura 14 muestra un ejemplo para el dominio de procesos de software.



Nivel M2: Metamodelo general para procesos de Software

Nivel M1: Proceso de desarrollo según metodología MSTL

Nivel M0: Datos de un proyecto completo que aplica MSTL

Figura 14. Aplicación de MOF a procesos de software. Fuente: Ruiz y Verdugo (2008).

Según Ruiz y Verdugo (2008), un metamodelo describe un conjunto de de conceptos genéricos y sus interrelaciones, sirven de base para la definición de modelos de un cierto dominio. Por tanto, un metamodelo es un modelo de modelos. Mediante el uso de un metamodelo se pueden representar modelos del correspondiente dominio.

El gran potencial de esta nueva manera de trabajar con procesos ha hecho que surja la Ingeniería de Procesos (también llamada Ingeniería de Métodos), que trata de aplicar a los procesos maneras y técnicas que antes han demostrado su utilidad en los productos (software). La Figura 15 muestra un resumen de las nuevas actividades y roles en esta nueva faceta.

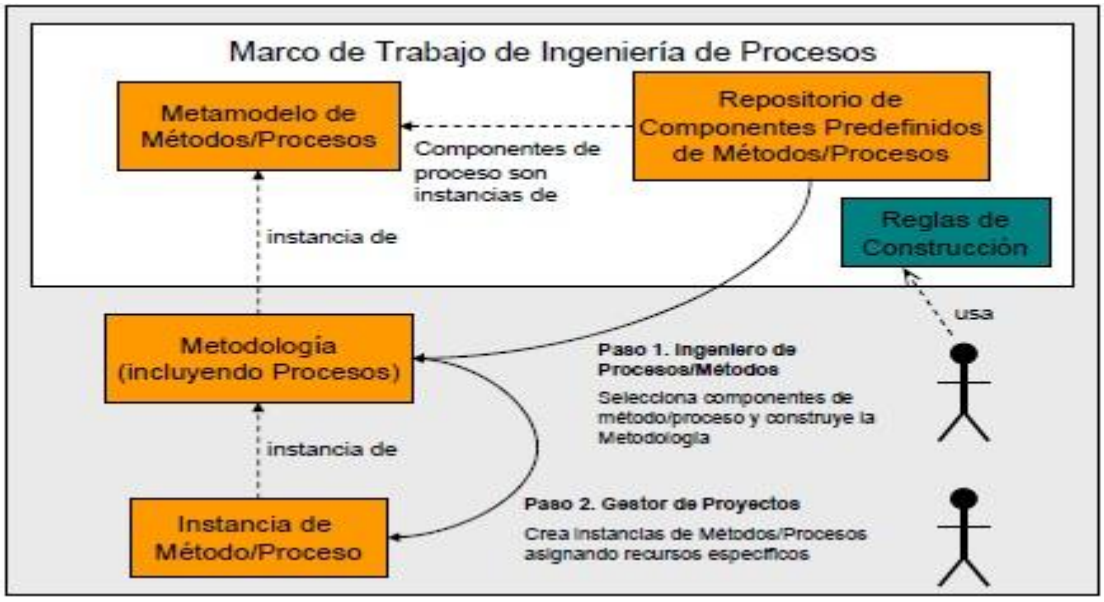


Figura 15. Ingeniería de Procesos. Fuente: Ruiz y Verdugo (2008).

En este sentido, SPEM 2 (Software & Systems Process Engineering Specification v2.0) es un metamodelo de procesos de ingeniería de software y de ingeniería de sistemas. SPEM 2, sirve para definir procesos de desarrollo de software, sistemas y sus componentes. Su alcance se limita a los elementos mínimos necesarios para definir los procesos sin añadir características específicas de un dominio o disciplina particular; pero sirve para métodos y procesos de diferentes estilos, culturas, niveles de formalismo o modelos de ciclo de vida.

Adicionalmente, SPEM 2 no es un lenguaje de modelado de procesos en general, sino que está orientado a los procesos de software, aunado a esto no posee conceptos propios para el modelado de comportamiento, pero si incluye mecanismos para encajar en el método externo seleccionado (Diagrama de actividad de UML 2, BPMN/BPDM).

Ruiz y Verdugo (2008), indican que la idea principal de SPEM 2 para representar procesos está basada en tres elementos básicos: rol, producto de trabajo y tarea. Las tareas representan el esfuerzo a realizar, los roles indican quién lo hace y los productos de trabajo representan las entradas que se utilizan en las tareas y las salidas que se producen. En la figura 16, muestra a continuación la idea principal del proceso con SPEM 2.

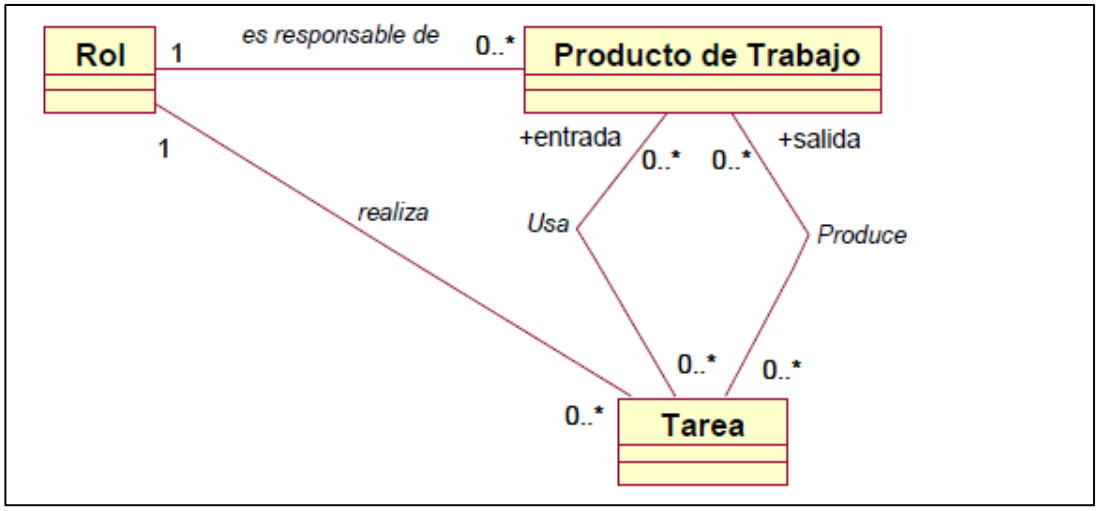


Figura 16. Idea Básica de proceso con SPEM 2. Ruíz y Verdugo (2008).

SPEM 2 puede ser una importante ayuda para que las empresas que llevan a cabo proyectos software debido a que pueden enfrentar mejor los problemas relacionados con los procesos. Entre dichos problemas, cabe destacar los siguientes:

- Miembros de los equipos no tienen acceso fácil y centralizado a la información de procesos que necesitan.
- Diferentes desarrolladores manejan diferentes fuentes o versiones de la misma información.

- Es difícil combinar e integrar informaciones y procesos que están en formatos propietarios diferentes.
- Cada libro, manual, herramienta utiliza un lenguaje y estilo diferente.
- Es duro definir una aproximación de desarrollo organizada y sistemática que se adapte las necesidades.
- Cultura, prácticas establecidas, requisitos de certificación, legales, etc.

Además de un metamodelo para la ingeniería de procesos, SPEM 2 es un marco de trabajo conceptual que provee los conceptos necesarios para modelar, documentar, presentar, publicar, gestionar, intercambiar y realizar métodos y procesos de software.

Por ello está destinado a ingenieros de procesos, jefes de proyectos, gestores de proyectos y programas; que son los responsables de mantener e implementar procesos para sus organizaciones o para proyectos específicos. En la figura 17, se muestra el resumen del marco de trabajo general de SPEM 2.

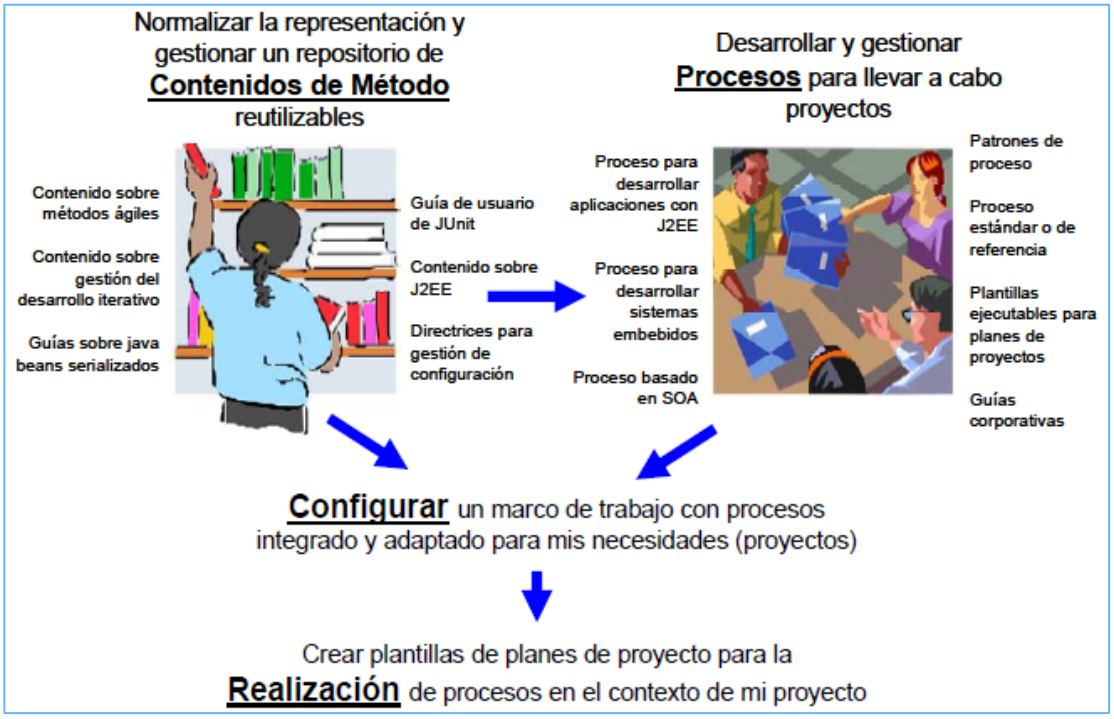


Figura 17. Marco de trabajo general de SPEM 2. Fuente: Ruíz y Verdugo (2008).

Al trabajar con SPEM 2 existen 4 escenarios fundamentales:

a) ***Crear un repositorio*** de "contenidos de método" reutilizables, es decir, una colección organizada de roles, tareas, productos de trabajo, guías, fragmentos de método y procesos, etc. Esto en sí solo ya es un valor para cualquier organización de software, ya que supone disponer de un repositorio de conocimiento sobre procesos en un formato estandarizado. Además, es de gran ayuda a los desarrolladores de software porque en su trabajo necesitan conocer cómo hacer las tareas de desarrollo y mantenimiento de software, cómo gestionar el proyecto, cómo comprender los productos de trabajo que se deben crear en cada tarea, cuales son las habilidades requeridas en cada rol, y disponer de las guías, directrices, plantillas, etc. adecuadas en cada momento. Además, el repositorio basado en SPEM es una base de conocimiento ideal para la formación en procesos.

b) Dar soporte al ***desarrollo, gestión y crecimiento de procesos software***. Esto implica combinar, reutilizar y extender los elementos de método anteriores para configurar los procesos que sirven para guiar los proyectos. Desde fragmentos de proceso elementales se puede llegar a generar todo un proceso completo o toda una metodología, incluyendo varios procesos. SPEM 2 ayuda a los equipos de desarrollo a definir o seleccionar un proceso, incluyendo opciones para que los mismos métodos puedan ser aplicados de forma diferente en momentos distintos o en proyectos distintos, para comprender claramente como se relacionan unas tareas con otras, o para que los procesos puedan ser vistos como flujos de trabajo o como estructuras de desglose de trabajo (WBS), según interese en cada momento. Para lo anterior, SPEM soporta la creación sistemática de procesos basada en la reutilización de contenidos de método. También provee el fundamento conceptual para que los ingenieros de procesos y gestores de proyectos seleccionen, adapten y rápidamente ensamblen procesos para sus proyectos concretos. El ensamblado rápido de procesos es posible gracias a la implementación de

catálogos de procesos predefinidos, "trozos" de proceso o patrones de proceso.

c) Establecer ***un marco de trabajo general de la organización*** a partir de los procesos y los elementos definidos anteriormente. Para ello, SPEM permite dar soporte al despliegue del contenido de método y proceso que justo se necesita en cada caso, teniendo en cuenta que ningún proyecto es exactamente como el anterior y nunca exactamente el mismo proceso software se ejecuta dos veces. En este punto es importante recordar que el nivel 3 de CMMI (proceso definido) necesita disponer de procesos estándares en la organización y de mecanismos de particularización (tailoring) y SPEM 2 provee de ambas cosas. Entre otras capacidades adicionales, SPEM 2 incorpora conceptos para:
   - Reutilización de procesos o patrones de procesos,
   - Variabilidad (procesos que incluyen partes alternativas configurables), y
   - Particularización (los usuarios definen sus propias extensiones, omisiones y puntos de variabilidad sobre procesos estándares reutilizados).

d) Generar ***plantillas para planes de proyecto*** concretos. Esto supone que a partir de ahora los jefes de proyecto pueden contar con mucha más información y disponible de manera automática a la hora de definir los planes de los proyectos. Para darles auténtico valor, las definiciones de los procesos deben ser desplegadas en formatos que permitan su realización automática (sistemas de gestión de proyectos y recursos, motores de flujos de trabajo). Para ello, SPEM incluye estructuras de definición de procesos que permiten expresar cómo un proceso será realizado de forma automática con estos sistemas. Ejemplos de ello son las iteraciones (una o varias definiciones de trabajo serán repetidas varias veces en un proyecto) y las ocurrencias múltiples (varias instancias de una definición de trabajo pueden llevarse a cabo a la vez de forma paralela).

Por otro lado, SPEM 2 posee un mecanismo de trabajo, el cual se describe de la siguiente manera (ver figura 18, Niveles de Modelado e instanciaciones):

- Como un ***Metamodelo MOF-compliant***, es decir, un metamodelo MOF del nivel M2 de forma que todos sus elementos están definidos mediante instanciación de elementos del meta-metamodelo universal (del nivel M3). Este metamodelo define todas las estructuras y reglas de estructuración para representar contenidos de métodos y procesos. Es completo en sí mismo, es decir, no necesita de otro metamodelo, aunque en la práctica reutiliza algunas clases de UML 2 por razones de ahorro.
- Como ***Perfil de UML 2***, es decir, como un conjunto de estereotipos UML 2 que permiten representar métodos y procesos usando UML 2. En este caso, la definición sólo abarca la presentación, mientras que las definiciones semánticas y restricciones están en el metamodelo anterior.

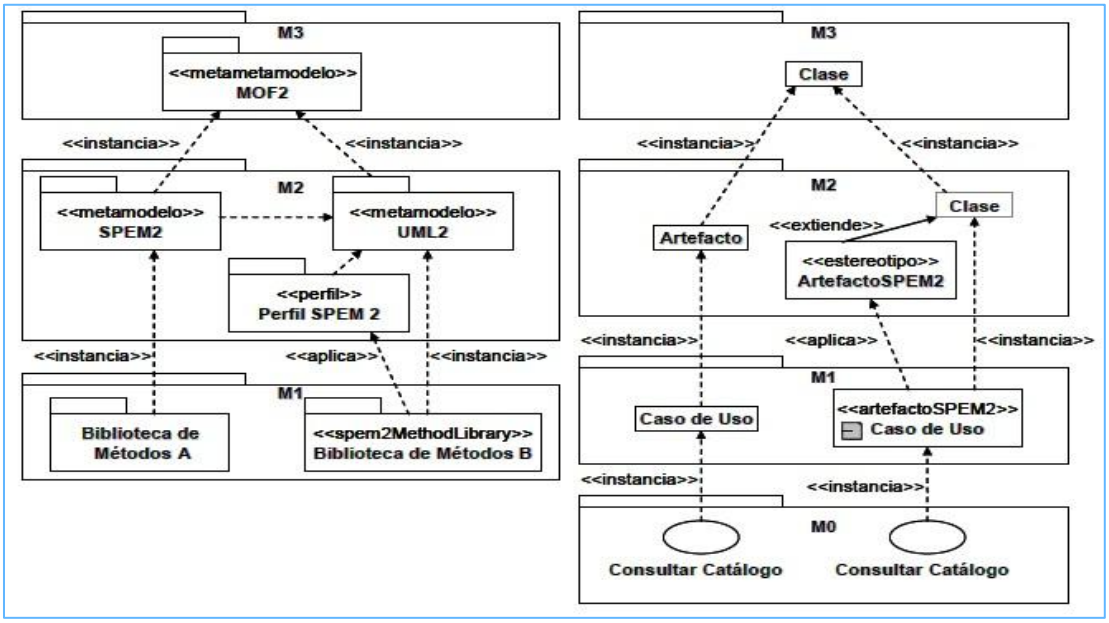


Figura 18. Niveles de Modelado e Instanciaciones. Fuente: Ruíz y Verdugo (2008).

Metamodelado de SPEM

Tal como se deduce del marco de trabajo (véase Figura 17), en SPEM 2 se distinguen dos grupos de conceptos a la hora de implementar una metodología (véase Figura 19 y Figura 20):

a) Primero, se puebla el Method Content (contenido del método) con Content Elements (elementos de contenido), es decir, los elementos primarios o constructores básicos.
b) Después, se combinan y reutilizan dichos elementos para obtener Processes (procesos).

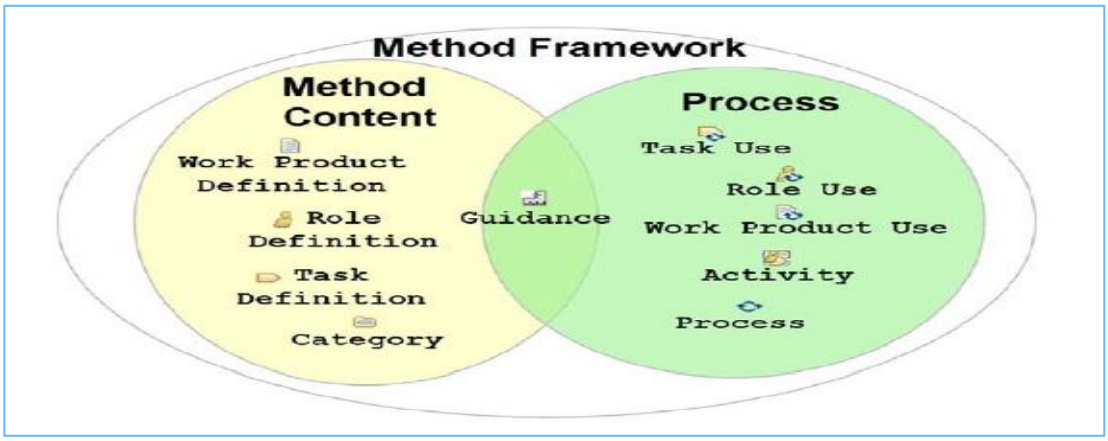


Figura 19. Aspectos principales para modelar con SPEM. Fuente: Ruíz y Verdugo (2008).

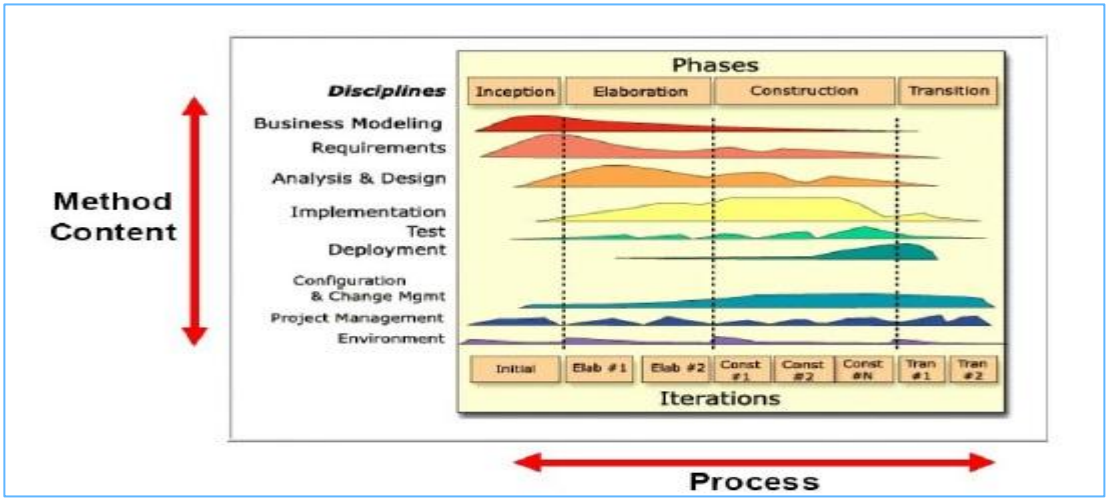


Figura 20. Ortogonalidad del Method Content y de los Process (igual que en RUP). Fuente: Ruíz y Verdugo (2008).

Otras características avanzadas del metamodelado de SPEM 2 son:

- Mantenimiento consistente de muchos ***procesos alternativos***. Para ello, SPEM incluye: i) un conjunto extendido de interrelaciones de reutilización y

variabilidad con semántica de herencia y orientación a aspectos; ii) conceptos de patrones de proceso, y iii) plugins de métodos. Estas opciones permiten tener diferentes variantes de procesos específicos, basados en los mismos contenidos de método y estructuras de procesos, pero aplicados con diferente detalle y escala.

- Muchos ***ciclos de vida diferentes***. SPEM permite trabajar con distintos tipos de ciclos de vida del software: Cascada, Iterativo, Incremental, Evolutivo, etc. Para ello incluye un conjunto de atributos que permiten especificar aspectos temporales para los elementos de proceso que luego pueden ser asociados a los planes de proyectos. Un ejemplo de atributo para clases de ciclos de vida es la propiedad Iteración, que permite representar que la ejecución de una o varias descripciones de trabajo se pueden repetir más de una vez.
- ***Variabilidad y extensibilidad***. Para lo cual SPEM incluye un mecanismo de plugins de dos tipos (ver en la Figura 21), estos son: a) ***Method plug-ins*** para particularizar y adaptar contenidos de método sin modificar el original; y b) ***Process plug-ins*** para procesos, pudiendo añadir o sustituir elementos de trabajo en el WBS (estructura de desglose de trabajo del proceso) sin afectar al original.

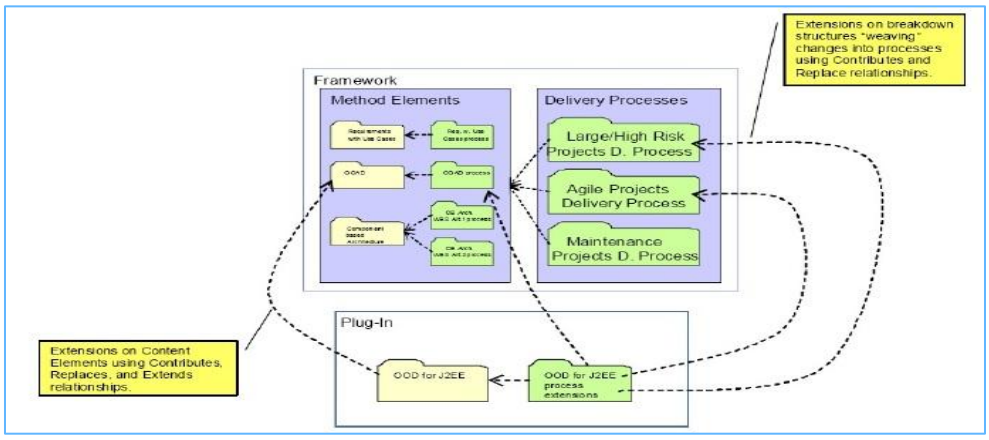


Figura 21. Ejemplo del mecanismo de variabilidad de SPEM para manejar la variabilidad y extensibilidad. Fuente: Ruíz y verdugo (2008).

- ***Patrones de proceso***. Son bloques (trozos de proceso) reutilizables para crear nuevos procesos. La selección y aplicación de un patrón de proceso puede ser hecha de dos formas: a) puede ser copiado y modificado, permitiendo individualizar el contenido del patrón según las necesidades de cada momento; o b) puede ser aplicado por medio del mecanismo de Actividad en Uso (Use Activity), que es una forma avanzada de reutilizar estructuras de proceso. Una Actividad en Uso define tipos de interrelaciones para que cuando el patrón esté siendo revisado o modificado, todos los cambios se reflejen automáticamente en todos los procesos en que se aplica el patrón.
- ***Componentes de proceso***. Son piezas de proceso sustituibles y reutilizables basadas en los principios de encapsulación y caja negra. No se especifica la descripción de trabajo interna del componente, sino que sólo se especifican los productos de trabajo de entrada y salida que habrá mediante los llamados puertos de productos de trabajo (ver Figura 22). Esta opción de SPEM permite manejar las situaciones en que un proyecto requiere que partes del proceso no sean decididas hasta la ejecución o nunca (caso típico: outsourcing).

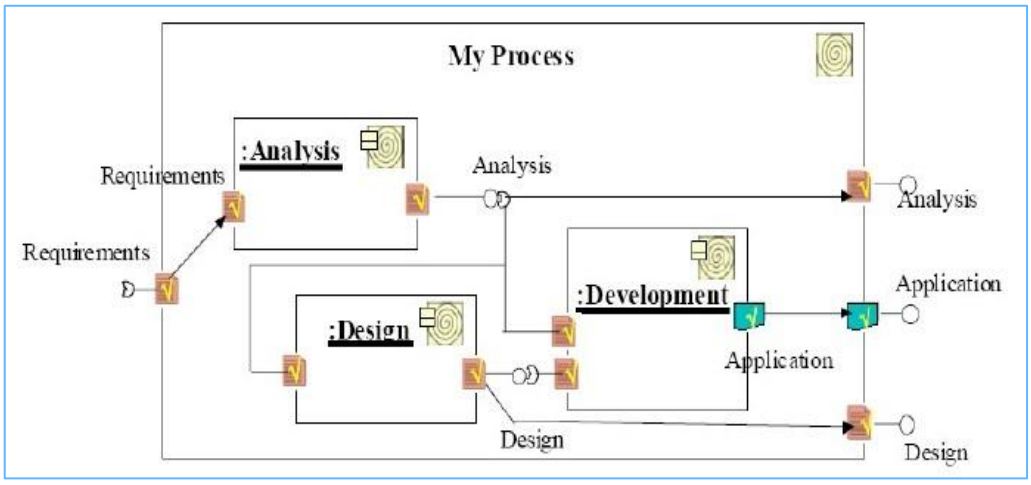


Figura 22. Ejemplo de Componente de proceso en SPEM 2. Fuente: Ruíz y Verdugo (2008).

Proceso para la ingeniería de Dominio Basado en Calidad de Software (InDoCaS)

Según Canelón (2010), InDoCaS (Proceso para la ingeniería de Dominio basado en Calidad de Software) puede ser utilizado en el enfoque de desarrollo de líneas de producción de software siendo instanciado para un dominio específico y los activos de software producidos pueden ser reutilizados para generar un producto de una familia particular del dominio.

InDoCaS tiene como propósito obtener una arquitectura base para una familia, aplicando las actividades correspondientes a la disciplinas de análisis y diseño. Este proceso se encuentra estructurado de la siguiente manera:

- RECLAMO (Requeriment Clasification Model): Modelo de Clasificación de requisitos propuesto por Chirinos y otros (2004).
- ISO/IEC 25010: El Estándar Internacional que describe un modelo bipartito para la calidad del producto de software, ISO/IEC (2009).
- Un proceso para el análisis del dominio para construir el modelo de calidad propuesto por Losavio y Otros (2008).
- FODA (Feature-Oriented Domain Analisys): análisis del dominio orientado a rasgos, desarrollado por el SEI (Software Engineering Institute) para el modelo de variabilidad, Kang et al. (1990).
- Un proceso general para el diseño arquitectónico del dominio propuesto por Hofmeister y otros (2007).
- ADD (Attribute-Driven Design Method): Método de diseño dirigido por atributos, usado para la formulación de escenarios de calidad, Bass et al (2003).
- ATAM (Architecture Tradeoff Analisys Method): Método para elegir una arquitectura para un sistema, utilizado en InDoCaS para la evaluación arquitectural.

Para modelar InDoCaS se utiliza la notación SPEM 2, en la figura 23 se puede observar la disciplina de análisis y diseño que componen este proceso.

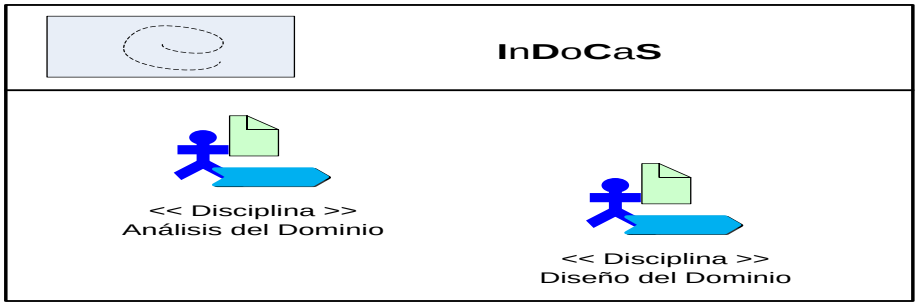


Figura 23. Disciplinas del Proceso InDoCaS. Fuente: Canelón (2010).

A continuación, se especifica una de estas disciplinas del Proceso **InDoCaS**.

InDoCaS: Análisis del Dominio

La fase de análisis del dominio del proceso InDoCaS es fundamental para el desarrollo de las líneas de producción de software, ya que allí se definen los aspectos comunes a todas las familias del dominio y los aspectos 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 mismo que brinda soporte al resto de las fases" Canelón (2010).

Se considera en general, que los requisitos no funcionales, expresados en términos de requisitos de calidad, son cruciales para el diseño arquitectural específicamente cuando las aplicaciones deben responder a situaciones críticas y a cambios en el entorno.

Cabe destacar que InDoCaS, define su propia nomenclatura para describir las actividades y artefactos que forman parte de las disciplinas del proceso (Análisis y Diseño), que se muestran a continuación en la tabla 5 y 6:

Tabla 5. Esquema de Actividad InDoCaS

InDoCaS	
ACTIVIDAD	**DESCRIPCIÓN**
Nombre de la Actividad:	
Responsable:	
Objetivo:	
Nro. De Identificación	

Tipo documento generado	
Técnica(s) utilizada(s)	
Artefactos de Entrada	
Artefactos de Salida	

Fuente: Canelón (2010).

Como se observa en la tabla 5, cada actividad lleva: el Nombre de la Actividad, Responsable, Objetivo, un número ordinal para el Nro. De Identificación, que sirve para futuras referencias, tipo de documento generado (lista, esquema, modelo, diagrama, tabla entre otros), la Técnica(s) Utilizada(s) para la construcción de la actividad, los Artefactos de Entrada que se necesitan para la actividad y los Artefactos de salida producidos en la actividad.

Tabla 6. Esquema de Artefacto InDoCaS

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	
Constructor:	
Objetivo:	
Nro. De Identificación ACTIVIDAD	
Nro. De Identificación ARTEFACTO	
Formato Asociado	

Fuente: Canelón (2010).

Adicionalmente en la tabla 6, se muestran los elementos para definir un artefacto InDoCaS entre ellos: Nombre del artefacto, Constructor responsable, un número ordinal para el Nro. De Identificación ACTIVIDAD para asociarlo a la actividad en referencia, un número ordinal para el Nro. De Identificación ARTEFACTO que hace referencia al artefacto y el formato asociado al tipo de documento generado (lista, esquema, modelo, diagrama entre otros).

Seguidamente se observa en la figura 24, el análisis del dominio utilizando la notación SPEM 2, está fase está conformada por seis actividades: 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.

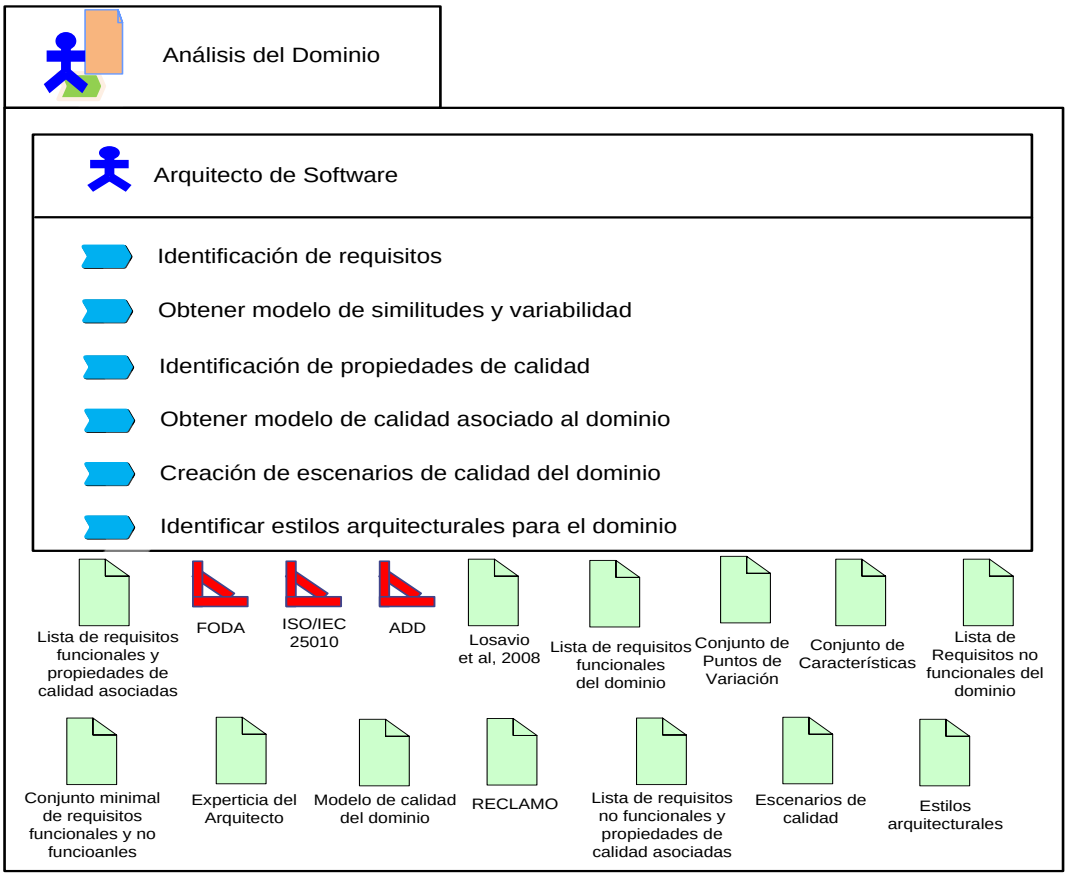


Figura 24. Análisis del Dominio InDoCaS. Fuente: Canelón (2010).

A continuación, en la figura 25, se presenta el diagrama de actividades de la disciplina análisis del dominio, donde se muestran las entradas y las salidas de cada una de las actividades y la secuencia de ejecución, así mismo se indica qué técnicas intervienen en cada una de las actividades.

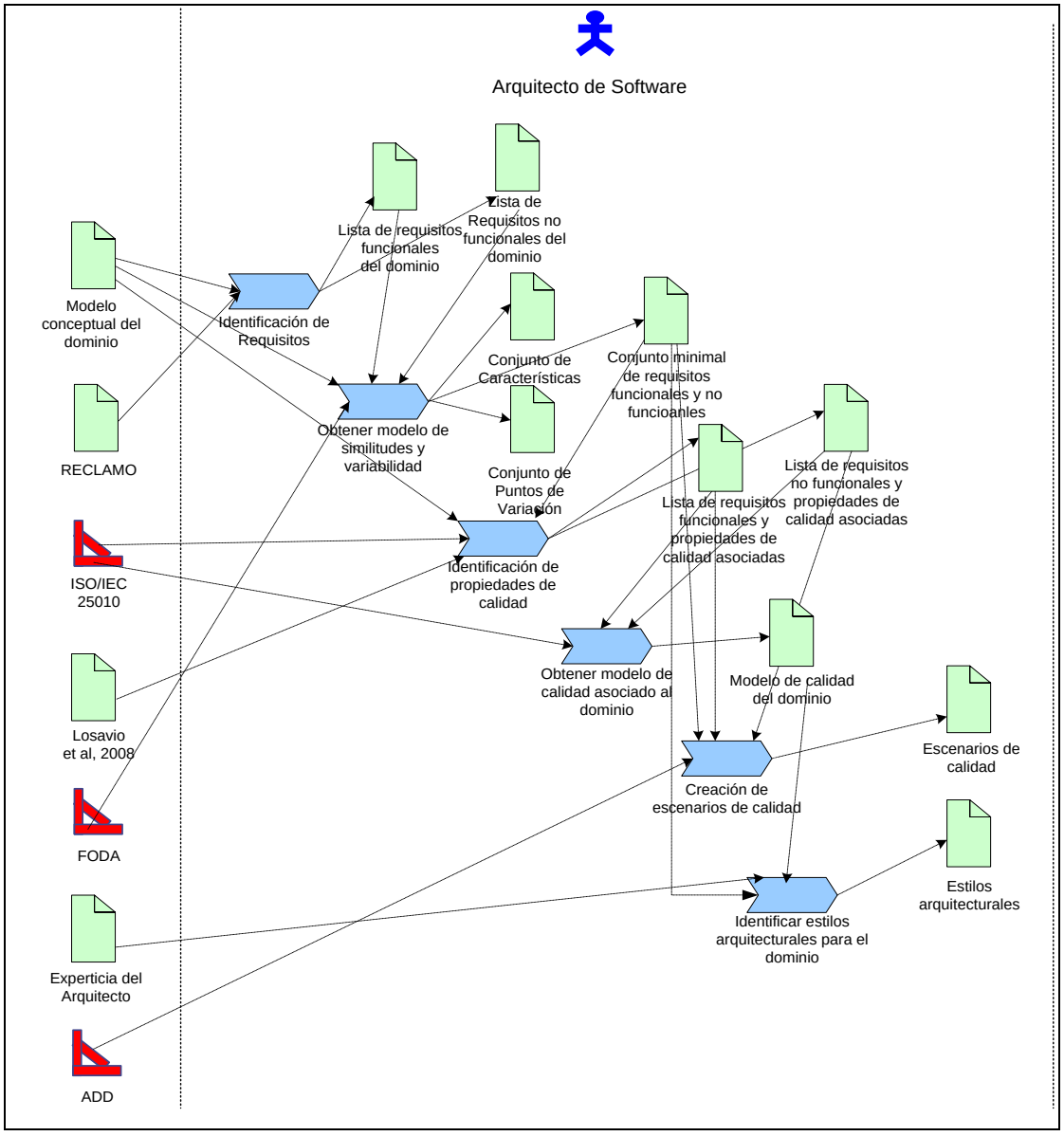


Figura 25. Diagrama de Actividad del Análisis del Dominio InDoCaS. Fuente: Canelón (2010).

A continuación en la tabla 7, se resumen las actividades de la disciplina análisis del dominio, los artefactos de entrada a cada actividad y los artefactos que produce la misma, así como la(s) técnica(s) asociada(s).

Tabla 7. Lista de actividades para el análisis del dominio InDoCaS

Nro.	Actividad	Artefactos de Entrada	Artefactos de Salida	Técnica
A_01	Identificación de Requisitos	- Modelo conceptual del dominio. - Reclamo.	- Lista de requisitos funcionales. - Lista de requisitos no funcionales.	RECLAMO
A_02	Obtener el modelo de similitudes y variabilidad	- Modelo conceptual del dominio. - Lista de requisitos funcionales. - Lista de requisitos no funcionales.	- Conjunto de características. - Conjunto minimal de requisitos funcionales y no funcionales. - Conjunto de puntos de variación.	FODA
A_03	Identificación de propiedades de calidad	- Modelo conceptual del dominio. - Conjunto minimal de requisitos funcionales y no funcionales. - Losavio et al., 2008.	- Lista de requisitos funcionales y propiedades de calidad asociadas. - Lista de requisitos no funcionales y propiedades de calidad asociadas.	ISO/IEC 25010
A_04	Obtener modelo de calidad del dominio	- Lista de requisitos funcionales y propiedades de calidad asociadas. - Lista de requisitos no funcionales y propiedades de calidad asociadas.	- Modelo de calidad del dominio.	ISO/IEC 25010
A_05	Creación de escenarios de calidad del dominio	- Modelo de calidad del dominio. - Lista de requisitos funcionales y propiedades de calidad asociadas. - Lista de requisitos no funcionales y propiedades de calidad asociadas. - Conjunto minimal de requisitos funcionales y no funcionales.	- Escenarios de calidad.	ADD
A_06	Identificar los estilos arquitecturales para el dominio	- Conjunto minimal de requisitos funcionales y no funcionales. - Modelo de calidad del dominio.	-Estilos arquitecturales.	Experticia del Arquitecto

Fuente: Canelón (2010).

Calidad de Software: Estándar ISO 25010

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, desde donde se extraen una serie de parámetros básicos que deben tomarse en cuenta, al desarrollar software de calidad.

En este sentido, la Organización Internacional para la Estandarización ISO publicó el estándar ISO-IEC 9126-1, en el año 2001. De acuerdo a las debilidades encontradas en este estándar, se hizo una revisión y se generó el ISO/IEC 25010 publicado en octubre del 2009. El mismo, es parte de la serie de estándares SQuaRE y fue preparado por el Comité Técnico Conjunto ISO/IEC JTC 1, Subcomité SC 7 de Ingeniería de Software y Sistemas.

El Estándar Internacional ISO/IEC 25010 describe un modelo bipartito para la calidad del producto de software el cual se presenta en la figura 26.

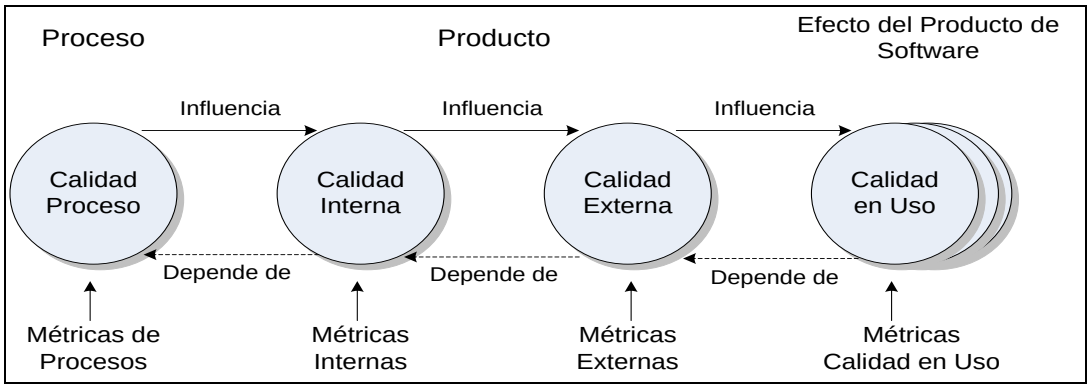


Figura 26. Enfoques de calidad. Fuente: ISO/IEC 25010 (2009).

Según ISO/IEC 25010, el enfoque de calidad está compuesto por:

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

b) ***Calidad Externa***: proporciona una visión de la "caja negra" del software y trata las características relacionadas con la ejecución del software.

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

La primera parte del modelo especifica ocho (8) características para la calidad interna y externa como se muestra en la figura 27, que se desglosan en sub-características, que se manifiestan externamente cuando el software se utiliza como parte de un sistema informático, y por consiguiente, es resultado de las cualidades internas del software.

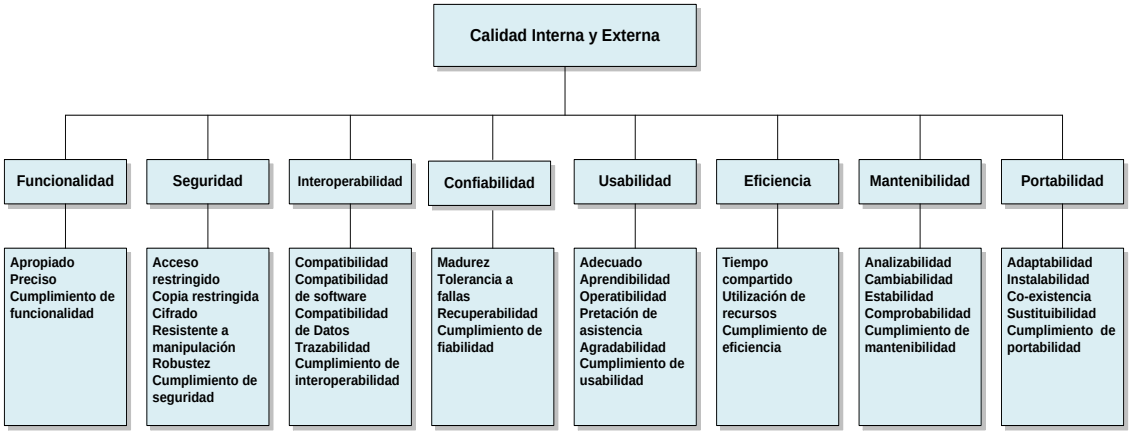


Figura 27. Características y sub-características de calidad interna y externa. Fuente: ISO/IEC (2009)

La segunda parte del modelo especifica cinco (5) características de la calidad en uso, como se muestra en la figura 28, que es el efecto combinado para el usuario de las ocho (8) características de la calidad del producto de software. Las características definidas son aplicables a todo tipo de software.

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

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

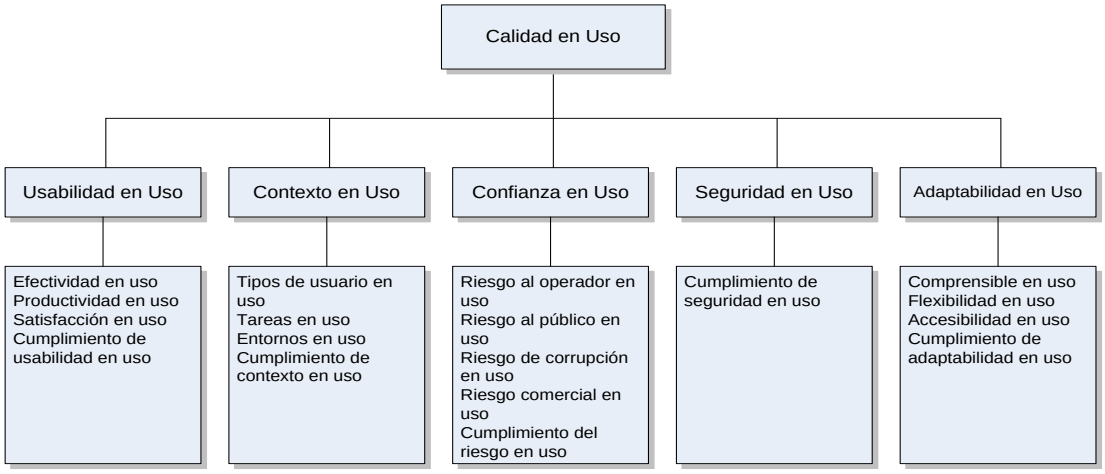


Figura 28. Características y sub-características de calidad en uso. Fuente: ISO/IEC (2009).

REST: Representational State Transfer

REST es un estilo de arquitectura de software para sistemas hipermedias distribuidos, tales como la Web. El término fue introducido por Roy Fielding en el año 2000 en su tesis doctoral, quien es uno de los principales autores de la especificación de HTTP. Básicamente REST, se refiere a una colección de principios para el diseño de arquitecturas en red, Navarro (2007). Está basado en estándares como HTTP, URL, Representación de los recursos: XML/HTML/GIF/JPEG y tipos MIME: text/xml, text/html.

A continuación, el detalle de los estándares en los cuales basa REST:

- ***HTTP - Protocolo de Transferencia de HiperTexto***: es un protocolo de la capa aplicación, considerado un estándar típico de petición respuesta de la computación cliente-servidor. Además, los recursos que se accede a través de HTTP se identifican mediante: Identificadores de recursos uniformes (URI) o

más específicamente, Localizadores de Recursos Uniformes (URLs), Oransa (2010). En la figura 29, se observa el proceso petición/ respuesta de HTTP.

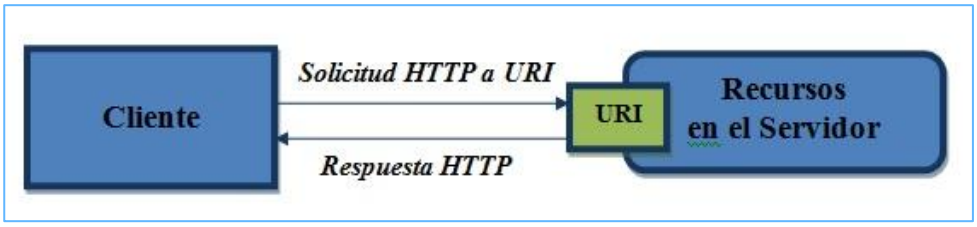


Figura 29. Estándar HTTP. Fuente: Adaptado de Oransa (2010).

HTTP es un protocolo sin estado. La ventaja de un protocolo sin estado, es que los host no necesitan conservar la información entre las solicitudes y los usuarios. Un método común para resolver este problema consiste en el envío y la recepción de cookies. Otros métodos incluyen sesiones del lado del servidor, las variables ocultas (si la página actual es un formulario) y parámetros codificados en la URL.

- ***URI – Interfaz de Recursos Uniforme***: es una cadena de caracteres utilizados para identificar un nombre o un recurso. Una URI se pueden clasificar en nombre (URN) y localizador (URL), o ambos. Un nombre de recursos uniforme (URN) funciona como el nombre de una persona, mientras que un localizador de recursos uniforme (URL) se asemeja a la dirección de esa persona. La URN define la identidad de un elemento y la URL proporciona un método para encontrarlo, Oransa (2010). A continuación la figura 30, muestra las categorías del esquema URI.

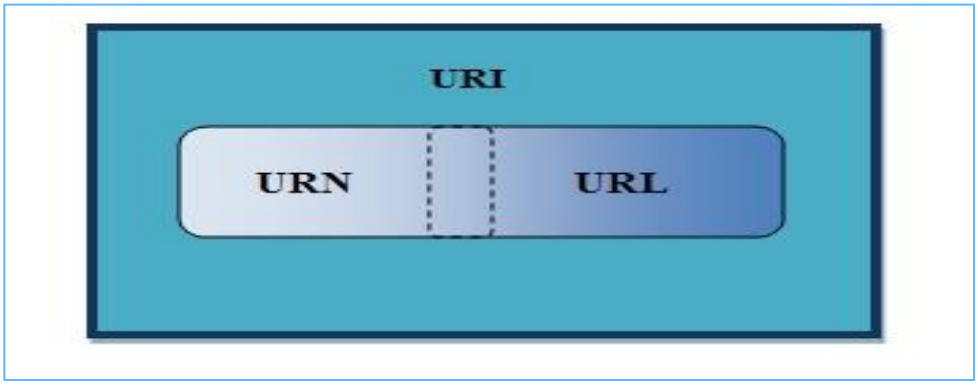


Figura 30. Diagrama de Venn de las Categorías del esquema URI. Fuente: Adaptado de Oransa (2010).

Por su lado, Fielding (2000), señala que REST utiliza un identificador de recursos para identificar el recurso particular que participa en una interacción entre componentes. Los conectores REST proporcionan una interfaz genérica para acceder y manipular el conjunto de valores de un recurso, independientemente de cómo la función de pertenencia se define o del tipo de software que está manejando la solicitud. La autoridad de asignación de nombres latiene el identificador de recursos, por lo que es posible hacer referencia al recurso, responsable de mantener la validez semántica de la asignación en el tiempo (es decir, garantizar que la función de pertenencia no cambie).

- ***Recursos***: es cualquier cosa que sea lo suficientemente importante como para ser referenciado como una cosa en sí misma, Richardson y Ruby (2007). Según Fielding (2010), es cualquier información que puede ser nombrado puede ser un recurso: un documento o una imagen, un servicio temporal (por ejemplo, "el tiempo de hoy en Los Ángeles"), una colección de otros recursos, un objeto no virtual (por ejemplo, una persona), y así sucesivamente. Define Recurso como: un mapeo conceptual para un conjunto de entidades, y no de la entidad que se corresponde con la asignación en cualquier punto particular en el tiempo.
- ***Representaciones***: Fielding (2000), indica que los componentes REST realizan acciones en un recurso mediante el uso de una representación para capturar el estado actual o previsto de ese recurso y la transferencia de esa representación entre los componentes. Una representación es una secuencia de bytes, más metadatos representación para describir estos bytes. Otros nombres de uso común, pero menos precisos para la representación son: documento, archivo, y la entidad mensaje HTTP, instancia o variante. Por su parte, Oransa (2010), menciona que diferentes clientes son capaces de consumir representaciones diferentes del mismo recurso. Por lo tanto, una representación puede adoptar diversas formas, pero tiene que estar disponible a través de la misma URI. La representación de los datos se ofrecen en una

variedad de formatos, para diferentes necesidades XML, JSON, HTML, Texto, entre otros.

En este mismo orden de ideas, Paramio (2007), señala los principios que define una arquitectura REST, entre ellos:

- El estado de la aplicación y la funcionalidad están divididos en lo que llamaremos recursos.
- Todos los recursos son accesibles de forma única mediante una sintaxis común en forma de enlaces hipermedios.
- Todos los recursos comparten una interfaz común para la transferencia de estado entre el cliente y el recurso, consistente en un conjunto estricto de operaciones bien definidas.
- Un protocolo cliente/servidor que no guarde el estado, cuyos resultados puedan almacenarse en un memoria caché, y que esté claramente separado en capas.

Asimismo, Pautasso et al. (2008), se refiere a principios tecnológicos e indica que el estilo arquitectónico REST se basa en cuatro (4) principios:

1. ***Identificación de recursos a través de URI***

   Un servicio web RESTful expone un conjunto de recursos que permitan identificar a los objetivos de la interacción con sus clientes. Los recursos son identificados por URIs, que proporcionan un espacio de direccionamiento global para el descubrimiento de servicios y recursos.

2. ***Interfaz Uniforme***

   Los recursos son manipulados usando un conjunto fijo de cuatro operaciones create, read, update, delete: PUT, GET, POST y DELETE. El PUT crea un nuevo recurso, que puede ser entonces eliminado usando DELETE. El GET recupera el estado actual de un recurso en alguna representación. Por último, el POST transfiere un nuevo estado dentro un recurso.

3. ***Mensaje Auto-Descriptivos***

   Los recursos están desacoplados de su representación, para que su contenido pueda ser accedido en una variedad de formatos (por ejemplo, HTML, XML,

texto plano, PDF, JPEG, etc.). Los metadatos sobre el recurso están disponibles y se utilizan, por ejemplo, para controlar el almacenamiento en caché, detectar errores de transmisión, negociar el formato de representación adecuado, y realizar la autenticación o control de acceso.

4. ***Interacciones con estado a través de hiperenlaces***

   Cada interacción con un recurso es sin estado, es decir, los mensajes de solicitud son auto-contenidos. Las interacciones con estado se basan en el concepto de transferencia de estado explícito. Existen varias técnicas para el intercambio de Estado, por ejemplo, reescritura de URI, las cookies, y los campos de formulario ocultos.

En la figura 31, se observa la comunicación o paso de mensajes utilizando REST.

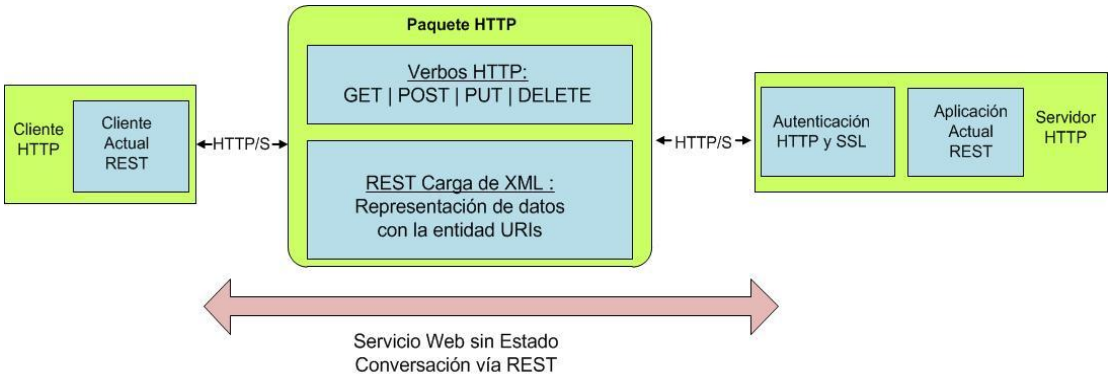


Figura 31. Paso de Mensajes utilizando REST. Fuente: Adaptado de Hinchcliffe (2005).

Estas condiciones finalmente tienen como objetivo aumentar la interoperabilidad, el desempeño, la escalabilidad y fiabilidad. Entre los beneficios de REST, Oransa (2010), menciona los siguientes:

Del lado del Cliente

- bookmarkable
- Fácil de experimento en el navegador
- Amplia compatibilidad con lenguaje de programación
- Elección de los formatos de datos

Del lado del Servidor

- Funciona bien con la infraestructura web existente
- Reducción de acoplamiento
- La expansión horizontal
- El servidor no mantiene el estado de aplicación
- Cacheable

Aplicaciones RESTful

Anteriormente, se menciono que REST nace del funcionamiento de la web. En este orden de ideas, las aplicaciones que funcionan bajo la web y que trabajan con el estilo arquitectural REST, son denominadas RESTful.

Richardson y Ruby (2007), establecen un enlace práctico entre las restricciones REST y el desarrollo de servicios web.

Por otro lado, Paramio (2007), indica que el aspecto de los recursos definidos en mediante una interfaz REST es el de una estructura jerárquica de directorios y archivos, donde el identificador de recurso colocado en la última parte de la URL hace el papel de archivo que contiene la información deseada, y el resto de la dirección parece un grupo de directorios anidados perfectamente lógicos en cuyo interior está almacenado el recurso.

Pautasso (2008), define varias pautas a seguir para diseñar servicios web RESTful, entre ellas:

1. Identificar los recursos que se van exponer como servicios (por ejemplo, orden de compra, catálogo de libros, etc.)
2. Definir una "buena" URL para cada recurso. Oransa (2010), muestra como ejemplo las siguientes:
   - http://myserver.com/empolyee/15
   - http://myserver.com/empolyee/Mohammad/Address
   - http://myserver.com/customers/Ahmed
   - http://myserver.com/customers/ahmed/orders/2
   - http://myserver.com/orders/1000/customer

3. Entender lo que significa hacer un GET, POST, PUT y DELETE en una URI de un determinado recurso. En la tabla 8, se muestra la terminología estándar HTTP, los cuales son análogas a la terminología de base de datos: los métodos CRUD.

Tabla 8. Conjunto estándar de métodos HTTP

Método	Uso	Base de Datos
GET	Leer, posiblemente en caché	Select
POST	Actualizar o crear sin una identificación conocido	Insert
PUT	Actualizar o crear con un ID conocido	Update
DELETE	Eliminar	Delete

Fuente: Adaptado de Oransa (2010).

4. Diseñar y documentar las representaciones de los recursos (formatos de carga de datos). A continuación se muestra un ejemplo de una representación en formato XML, Oransa (2010):

<order self="http://myserver.com/orders/10000/">
<customer ref="http://myserver.com/customers/2/">
<product ref="http://myserver.com/products/200/"/>
<amountvalue=“3"/>
</order>

5. Modelo de relaciones (por ejemplo, contención, referencia, transiciones de estado) entre los recursos con los hipervínculos que se pueden seguir para obtener más detalles.
6. Implementar y desplegar en un servidor web.
7. Probar con un navegador web.

A continuación, la figura 32 muestra el funcionamiento entre aplicaciones RESTful.

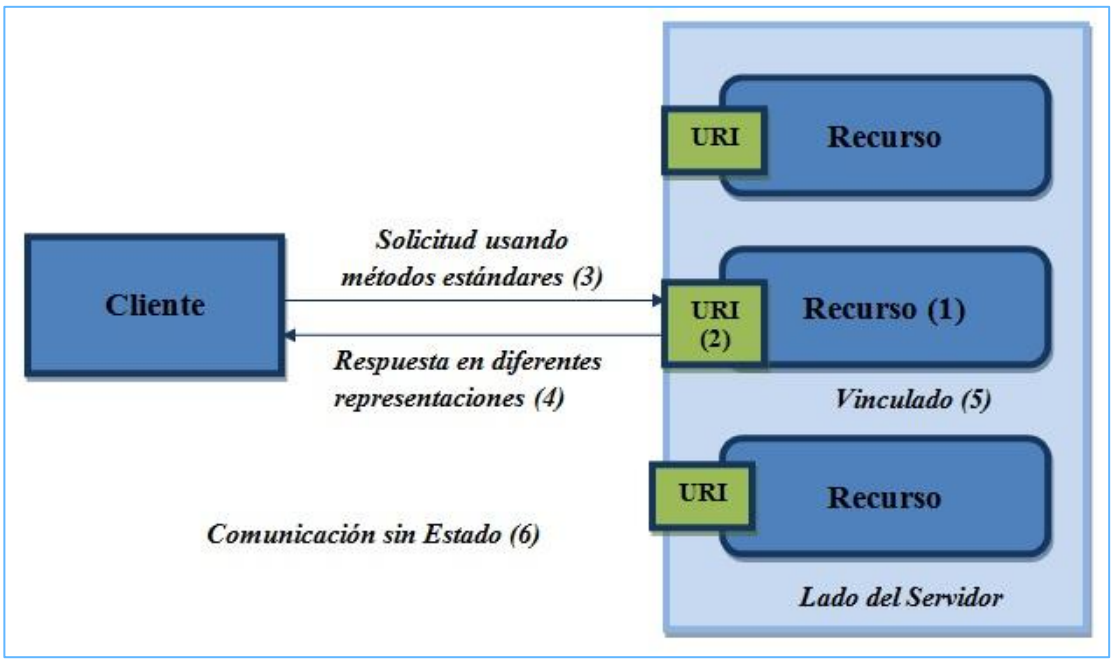


Figura 32. Funcionamiento de REST. Fuente: Adaptado de Oransa (2010).

CAPÍTULO III

MARCO METODOLÓGICO

En el desarrollo de una investigación, se debe indicar el tipo de datos que se requiere indagar, seleccionar los distintos métodos y técnicas que permitirán obtener la información requerida. A fin de cumplir con estos importantes aspectos se procederán a describir el marco metodológico de la presente investigación.

Naturaleza del Estudio

De acuerdo al problema planteado y en función de los objetivos a obtener, la presente investigación se incorpora al tipo de investigación documental de carácter proyectivo bajo la modalidad de Proyecto Especial, el cual el Manual de Trabajos de Grado de Especialización y Maestría y Tesis Doctorales (2010), define:

> El Proyecto Especial consiste en creaciones tangibles, susceptibles de ser utilizadas como soluciones a problemas demostrados, o que respondan a necesidades e intereses de tipo cultural. Se incluyen en esta categoría los trabajos de elaboración de libros de texto y materiales de apoyo educativo, el desarrollo de software, prototipos y de productos tecnológicos en general, así como también los de creación artística y literaria.

Por otro lado, el Manual para la Elaboración del Trabajo Conducente a Grado Académico de Especialización, Maestría y Doctorado UCLA (2002), define la Investigación Monográfica Documental:

> Es entendida como un estudio de problemas tipo teórico-práctico, con el propósito de ampliar y profundizar el conocimiento de su naturaleza. Se basa principalmente en fuentes bibliográficas, documentales y estudios comparados de análisis de problemas que ocurren en la práctica.

En lo referente tipo de investigación de carácter proyectivo, Hurtado (2008), define Investigación Proyectiva, "un tipo de investigación que propone soluciones a una situación determinada a partir de un proceso de indagación. Implica explorar, describir, explicar y proponer alternativas de cambio, mas no necesariamente ejecutar la propuesta".

Diseño de la Investigación

En el marco de la investigación planteada, respecto a proponer un Modelo de Integración basado en Líneas de Producción de Software para Aplicaciones RESTful; se define como diseño de la investigación el plan global de investigación que integra de un modo coherente y adecuadamente correcto técnicas de recogida de datos a utilizar, análisis previstos y objetivos. En otras, palabras el diseño de la investigación intenta dar de una manera clara y no ambigua respuestas a las preguntas planteadas de la misma, Balestrini (2006).

En este sentido, se detallara en etapas el plan de diseño de la investigación:

1. Estudio de las características de los Modelos de Integración de IAE.
2. Investigación de los Modelos de Integración que se adecuan al dominio de aplicaciones RESTful.
3. Comprensión de los beneficios que genera el uso de líneas de producción de software en las aplicaciones RESTful.
4. Construcción del Modelo de Integración basado en líneas de producción de software para aplicaciones RESTful.

Fases de la Investigación

Para lograr la propuesta de un Modelo de Integración basado en Líneas de producción de software para Aplicaciones RESTful, se requiere en líneas generales cumplir al menos las siguientes fases:

Fase I. Caracterización de los Modelos de Integración de IAE

En esta fase se deben describir las características de los modelos de integración de IAE propuestos por Brown (2000), entre ellos: ciclo, semilla, web, flujo, onda, anillo, célula y árbol, en cuanto a procesos de negocios, de manera que una vez caracterizados los modelos, estos sirvan para aclarar la dinámica del proceso de negocio, permitiendo así, proponer cambios adecuados para implementación e integración de nuevas tecnologías.

Fase II. Adecuación de Características de los modelos de IAE en Aplicaciones RESTful

Con esta fase, se pretende buscar similitudes o correlaciones entre las características de los Modelos de IAE y las aplicaciones desarrolladas como RESTful, de manera que el modelo a proponer este dirigido a aplicaciones empresariales con el estilo arquitectural REST, utilizando el modelo propuesto por Mendoza y Otros (2009).

Fase III. Creación de Líneas de Producción de software para Aplicaciones RESTful

Con el propósito de comprender que beneficios que se generan al momento de desarrollar la línea de producción de software para aplicaciones RESTful, la presente fase consiste en hacer una especialización del proceso de la ingeniería del dominio basado en calidad de software ***InDoCas***, en su fase de análisis apoyándose en el Metamodelo de Schreier (2011).

Fase IV. Construcción de la Propuesta

En esta penúltima fase, se diseñara el Modelo tomando en cuenta las características de los Modelos de Integración IAE y la línea de producción de Software para aplicaciones RESTful.

Fase V. Ejemplificación de la Propuesta

En esta última fase, se realizará un caso de estudio con el objetivo de verificar la aplicabilidad del modelo propuesto.

A continuación, se presenta la figura 33 las fases de la investigación definidas para lograr la propuesta.

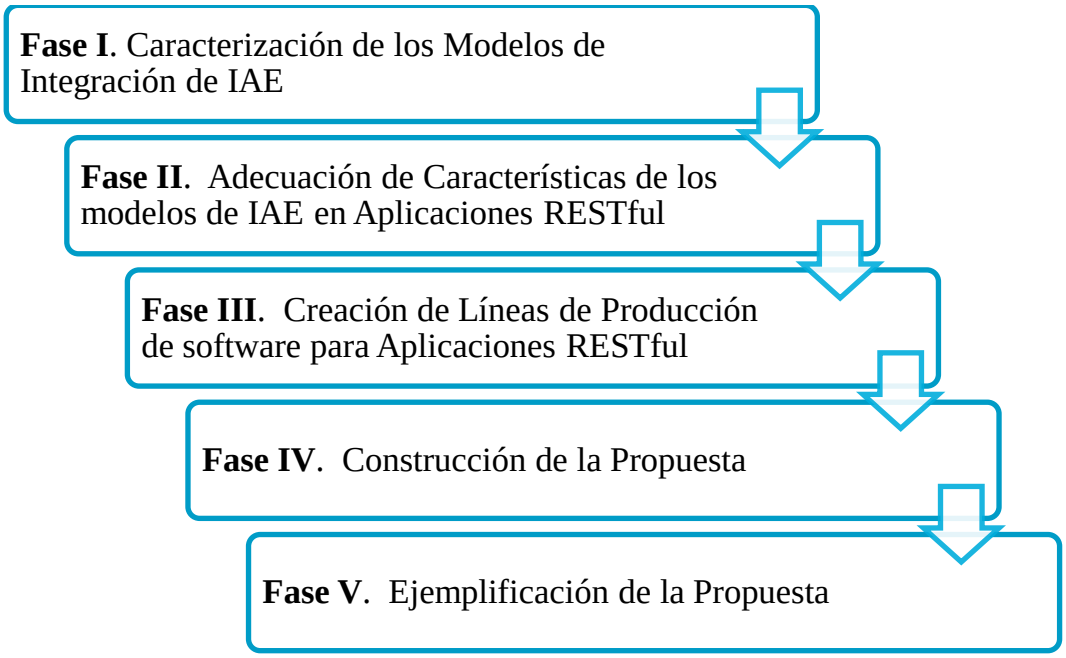


Figura 33. Fases de la Investigación. Fuente: La Autora (2012).

CAPÍTULO IV

PROPUESTA DEL ESTUDIO

Justificación

La aproximación más efectiva para la reutilización es la creación de líneas de productos o familias de productos. Debido a que una línea de producción de software nos ayuda a sacar provecho de los elementos comunes y manejar de manera eficaz las variaciones.

En este sentido, proponer un modelo de integración basado en líneas de producción de software, permite representar una arquitectura para un dominio de aplicaciones RESTful. Los principios de la arquitectura de software son importantes para guiar los procesos de desarrollo, incluso para asegurar que los productos de software incluyan características de calidad como: Escalabilidad (capacidad de las aplicaciones para cambiar su tamaño o configuración para adaptarse a las circunstancias cambiantes), Heterogeneidad (variedad de dispositivos, sistemas operativos, aplicaciones e interfaces de usuario) e Integridad (privacidad y seguridad que brindan las aplicaciones) por mencionar algunas propiedades de calidad que se pueden asegurar en etapas tempranas del desarrollo de aplicaciones.

Asimismo, representa un aporte significativo en la línea de investigación de Ciencias de la Computación específicamente en la Ingeniería de Software puesto que podría incentivar nuevas investigaciones en esta área, facilitando el aprendizaje acerca de arquitectura de software, guías para la construcción de software, ingeniería de dominio, líneas de producción de software, reutilización de componentes en familias de productos de software e inclusive se podría utilizar este modelo como marco de referencia e introducirle las adaptaciones necesarias para el logro de otros objetivos en otras áreas y así disfrutar las ventajas que ofrece el enfoque de líneas de producción como reducción de costos, esfuerzos y tiempos de desarrollo entre otros.

Objetivos

Objetivo General

Proponer un modelo de integración basado en líneas de producción de software para aplicaciones RESTful.

Objetivos Específicos

- Caracterizar los modelos de integración.
- Seleccionar los modelos de integración adecuados para aplicaciones RESTful.
- Diseñar actividades y artefactos en la disciplina análisis del dominio del proceso de ingeniería de dominio basado en calidad **InDoCaS** para aplicaciones RESTful.
- Ejemplificar la propuesta, mediante el proceso de desarrollo de las actividades y artefactos en un dominio determinado.

Descripción de la Propuesta

En la presente investigación se propone un modelo de integración basado en líneas de producción de software para aplicaciones RESTful. Para el logró de la misma se aplican conceptos de: Modelos de Integración, reutilización de activos, ingeniería de dominio, líneas de producción de software y REST, con el propósito de definir un modelo que proporcione una mejor integración y permita formar un conjunto de características que satisfagan las necesidades especificas de un determinado mercado todo esto mediante la creación de una línea de producción de software para aplicaciones RESTful.

En este sentido, la forma de integración se llevara a cabo por medio de la caracterización de los modelos propuesto por Brown (2000), entre ellos: Ciclo, Semilla, Anillo, Célula, Web, Flujo, Onda y Árbol.

Es necesario destacar, que la creación de la línea de producción para aplicaciones RESTful solo se implementará la fase de ingeniería del dominio, utilizando una especialización del proceso InDoCaS (proceso para la Ingeniería de Dominio basado en Calidad de Software), ejecutándose para este modelo la disciplina de análisis del dominio.

Así mismo, la especialización de la línea de producción se manifiesta usando la semántica de la interfaz de recursos basada en operaciones HTTP, los principios y restricciones del estilo arquitectural REST propuesto por Fielding (2000) y algunas características del metamodelo propuesto Schreier (2011).

Por tanto, la propuesta consiste, en primer lugar, en la caracterización de los modelos de integración por Brown (2000) y en segundo lugar, en la especialización del proceso InDoCaS en la disciplina de análisis del dominio, específicamente en la actividad "Identificar los estilos arquitecturales para el dominio", a la cual se agrega una actividad opcional denominada "Generar Recursos" y los artefactos necesarios para trabajar con líneas de producción de software, como: Especificación delos Recursos, Identificación de métodos asociados a los recursos y Especificación de formatos de respuesta de los Recursos.

Estructura de la Propuesta

El modelo propuesto en este trabajo de investigación, está basado en la caracterización y adecuación de los modelos de integración propuestos por Brown (2000) en aplicaciones RESTful y las actividades definidas por el proceso **InDoCaS**, Canelón (2010), en su disciplina análisis del dominio el cual se especializa en incorporar un punto de variación en la actividad "Identificar los estilos arquitecturales para el dominio" y agregar la actividad variante "Generar Recursos" asociando además los artefactos: Especificación de los Recursos, Identificación de métodos de los recursos y Especificación de formatos de respuestas de los Recursos.

Caracterización y Adecuación de los modelos de integración propuestos por Brown (2000) en aplicaciones RESTful

A continuación, la primera parte del modelo consiste en caracterizar de los Modelos de Integración propuestos por Brown (2000), entre estas plantillas tenemos: Ciclo, Semilla, Anillo, Célula, Web, Flujo, Onda y Árbol. En este orden de ideas, la caracterización se realizará en base al momento en que se debe aplicar cada plantilla:

- ***Plantilla Ciclo***: se puede aplicar a los procesos de negocio que se presentan de manera cíclica, como la presentación de informes financieros o los ciclos de desarrollo de productos. Esto se refleja en el código del programa, en las construcciones de procesamiento de bucle, como el pseudocódigo. También se aplica cuando se desea una estructura para la repetición de actividades específicas que se dan en una secuencia particular.
- ***Plantilla Semilla***: puede ser aplicada en la coordinación de un conjunto de diversos expertos para eliminar la duplicación de esfuerzos. Se puede aplicar para reducir el número de salidas para un proceso dado, la reducción de la sobrecarga de información dedicado a la especificación y la producción. O puede ayudarle a diseñar para una mejor utilización de los recursos mediante la especificación de insumos de procesos múltiples que varían en sus modos de entrada.
- ***Plantilla Anillo***: se puede aplicar a las situaciones siguientes: topología de red, descentralización versus centralización y modos de conectividad.
- ***Plantilla Célula***: se puede aplicar a las situaciones siguientes: análisis de interfaces, replicación, encapsulación y herencia, categorización y compartimentación.
- ***Plantilla Web***: se puede utilizar cuando se quiere representar dispositivos en red, diseñar algoritmos de enrutamiento y Redes y comunidades en línea.
- ***Plantilla Flujo***: se puede aplicar a las situaciones siguientes: análisis de flujo de trabajo, análisis de flujo de datos, mejora de procesos de negocio, escenarios de clientes e identificación de las varianzas.

- ***Plantilla Onda***: se puede aplicar a las situaciones siguientes: arquitectura N-niveles, especificación de versiones de productos y diseño para la cooperación y la sinergia.
- ***Plantilla Árbol***: se puede aplicar a las situaciones siguientes: composición y consolidación, análisis complejo de ramificación y sistemas de apoyo a decisiones.

Por otro lado, Mendoza et al. (2009) propone un modelo de especificación basado en características de los modelos propuestos por Brown, el cual está conformando en 3 niveles: Nivel -1: Características, Nivel -2: Sub-características y Nivel -3: Métricas, en la figura 34, se muestra dicho modelo.

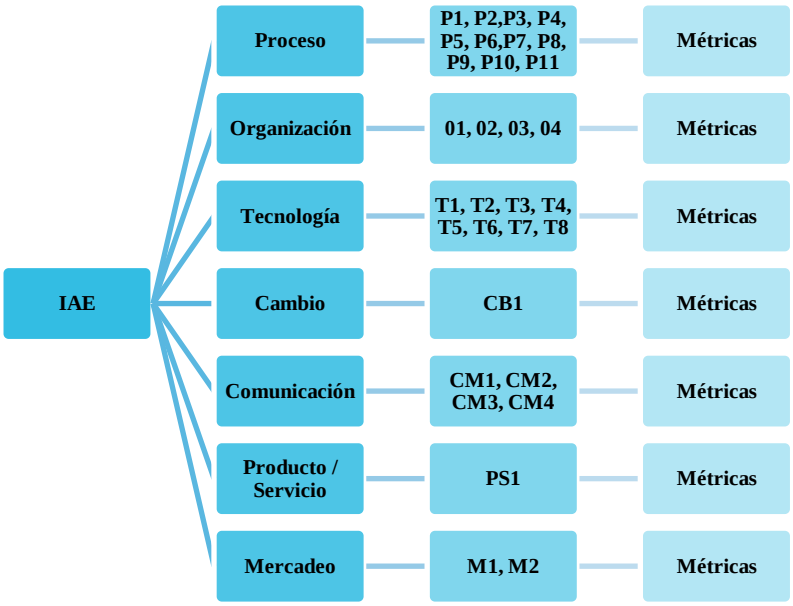


Figura 34. Descomposición de las características en los niveles. Fuente: Adaptado de Mendoza et al. (2009).

Este modelo de especificación basado en características fue aplicado a varios casos de estudio, representados a través de las organizaciones venezolanas, distribuidos en sector secundario y terciario. Dentro de sus hallazgos, se encontró que los modelos semilla, anillo y célula están presentes tanto en empresas grandes y como

en medianas en igual porcentaje. Mientras que, para los modelos Árbol, Web y Flujo el porcentaje varía dependiendo si es mediana o grande la empresa.

Una vez caracterizado los modelos de Brown (2000) y analizado el estudio realizado por Mendoza et al. (2009), con el propósito de determinar cuáles de este modelos posiblemente se adecuan a Aplicaciones RESTful, es necesario mencionar que las Aplicaciones RESTful son todas aquellas aplicaciones desarrolladas bajo el estilo arquitectural REST, estas aplicaciones se ofrecen como Servicios Web RESTful según Richardson y Ruby (2007). En este sentido, los Servicios Web poseen ventajas que los caracterizan como: aprovechan la tecnología Internet existente, por lo que es fácil comunicar aplicaciones ubicadas en servidores remotos, asumiendo un canal seguro mediante la utilización de *https* (*hypertext transfer protocol secure*). Ofrecen un interfaz conceptualmente coherente para distintos canales, como extranets de ventas, intranets, portal de clientes, etc. Por otro lado, es bastante seguro para el servidor ofrecer el acceso a sus servicios mediante peticiones http, ya que aísla relativamente bien la lógica interna de la aplicación de la visión de la parte llamante.

De acuerdo a estas características de los Servicios Web, existen modelos de integración que poseen características, Subcaracterísticas y métricas que apoyan desde el punto de vista tecnológico las decisiones que deben tomar las organizaciones en el momento de aplicar tecnologías nuevas como es el caso de Aplicaciones RESTful. En concordancia con el modelo de especificación basado en características de los Modelos de Integración propuesto por Mendoza y Otros (2009), se han seleccionado las características, Subcaracterísticas y métricas mencionadas anteriormente para la implementación de Aplicaciones RESTful. Seguidamente, la tabla 9, muestra las características, Subcaracterísticas y métricas seleccionadas.

Tabla 9. Características, Subcaracterísticas y Métricas adecuadas para Aplicaciones RESTful

Características	Subcaracterísticas	Métricas
Proceso	Automatización	**Cód**. P5.2 **Descripción**: Elementos Tecnológicos usados en el proceso. **Formulación**: • Correo Electrónico • Boletines de noticias • Video Conferencias • Páginas Web • Otro **Valor**: Puntuación total/5; 1<X<5
	Seguridad	**Cód**. P9.1 **Descripción**: Existencia de la seguridad de la información manejada por el proceso. **Formulación**: • Siempre=5 • Casi siempre=4 • Algunas Veces=3 • Pocas veces=2 • No existe=1
Organización	Estructura	**Cód**. O1.4 **Descripción**: Simultaneidad del funcionamiento de los departamentos organizacionales. **Formulación**: • Completamente=5 • La mayoría=4 • Medianamente=3 • Pocos=2 • Ninguno=1
		Cód. O1.5 **Descripción**: Nivel de Independencia entre Departamentos. **Formulación**: • Completamente=5 • Casi todos=4 • Medianamente=3 • Pocos=2 • Ninguno=1

Tecnología	Automatización	**Cód**. T1.1 **Descripción**: Uso de los Sistemas computarizados. **Formulación**: • Siempre=5 • Casi siempre=4 • Algunas veces=3 • Pocas veces=2 • Nunca=1
	Crecimiento	**Cód**. T2.1 **Descripción**: Existencia de los planes de nuevas tecnologías. **Formulación**: • Muy bien definidos=5 • Bien definidos=4 • Medianamente definidos=3 • Poco definidos=2 • No están definidos=1
	Comercialización	**Cód**. T3.2 **Descripción**: Existencia de Ventas a través de Internet. **Formulación**: • Muy interesada=5 • Interesada=4 • Medianamente interesada=3 • Poco interesada=2 • No está interesada=1
	Red	**Cód**. T4.1 **Descripción**: Existencia de la Página Web propia/anunciada en una. **Formulación**: • Posee su propia página=5 • Aparece anunciado en otra página=4 • Está interesado en obtener su propia página=3 • Está interesado en ser anunciado en otra página=2 • No posee la página, tampoco está interesado en ser anunciado=1
	Calidad	**Cód**.T5.1 **Descripción**. Mejoramiento del uso de

		las herramientas tecnológicas. **Formulación**: • Totalmente=5 • Bastante=4 • Medianamente=3 • Poco=2 • No se puede=1
	Sistemas de Información	**Cód**. T6.1 **Descripción**: Uso de SI. **Formulación**: • Siempre=5 • Casi siempre=4 • Algunas Veces=3 • Pocas Veces=4 • No se requiere=1
		Cód. T6.2 **Descripción**. Definición del Carácter del SI. **Formulación**: • Los resultados que produce son confiables • Tiene seguridad suficiente • Es fácil de usar • Abarca la mayoría de los requerimientos • Otro **Valor**: Puntuación total/5 1<X<5
	Procesamiento	**Cód**. T7.1 **Descripción**: Forma de guardar la información Procesada. **Formulación**: • Una base de datos • Archivos en forma de papel • Disquete/unidad ZIP • CD • Otro **Valor**: Puntación total/5, 1<X<5

Fuente: La Autora (2012).

De lo anteriormente expuesto, y de acuerdo a la tabla de codificación para la identificación de modelos propuesta por Mendoza y Otros (2009), se obtienen la

especificación de modelos de integración para aplicaciones RESTful y tabla de codificación con modelos de integración identificados, estos se muestran la figura 35 y tabla 10 respectivamente.

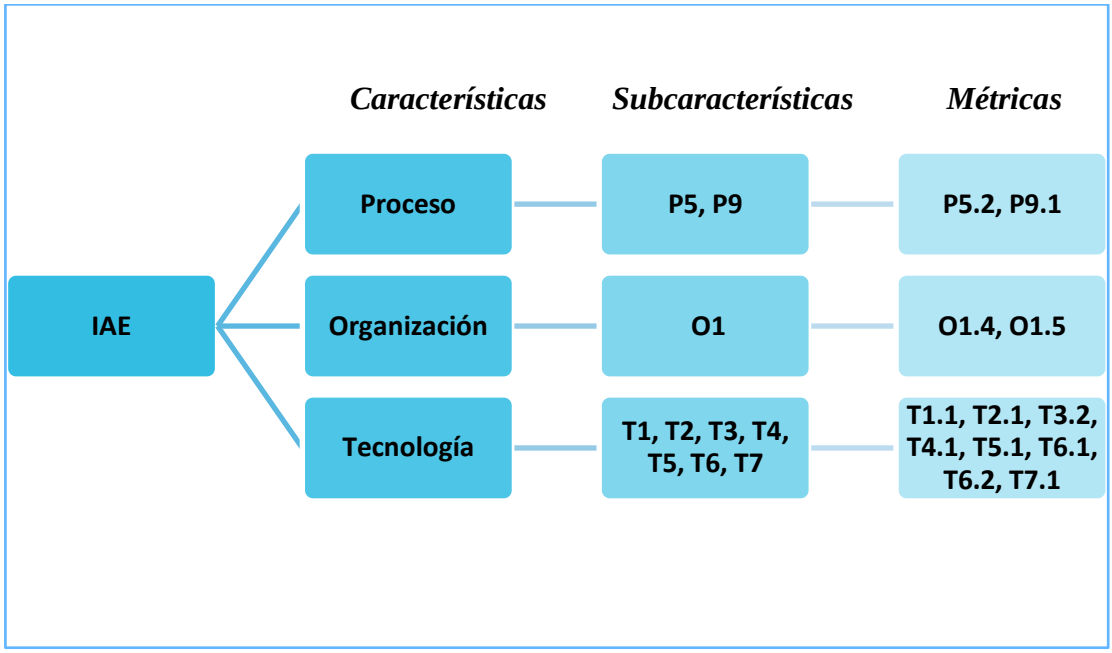


Figura 35. Especificación de los modelos de integración para aplicaciones RESTful. Fuente: La Autora (2012).

Tabla 10. Codificación con Modelos de Integración Identificados

Modelo	Características	Subcaracterísticas	Métrica
Web	Tecnología	T3. Comercialización	T3.2
		T4. Red	T4.1
Onda	Tecnología	T1. Automatización	T1.1
		T5. Calidad	T5.1
	Proceso	P9. Seguridad	P9.1
Anillo	Proceso	P5. Automatización	P5.2
	Tecnología	T2. Crecimiento	T2.1
Célula	Organización	O1. Estructura	O1.4, O1.5
	Tecnología	T6. Sistemas de Información	T6.1, T6.2
		T7. Procesamiento	T7.1

Fuente: La Autora(2012).

Especialización del Proceso para la ingeniería de dominio InDoCaS

A continuación se expone, la segunda parte del modelo propuesto en esta investigación, el cual consiste en la especialización del proceso **InDoCaS** en la disciplina de análisis del dominio, específicamente en la actividad "Identificar los estilos arquitecturales para el dominio", a la cual se agrega una actividad opcional denominada "Generar Recursos" y los artefactos necesarios para trabajar con líneas de producción de software, como: Especificación de los Recursos, Identificación de métodos asociados a los recursos y Especificación de formatos de respuesta de los Recursos.

En lo tocante, cada proyecto emprendido por una organización tiene sus propias características, motivo por el cual se hace necesario ajustar los modelos de procesos y metodologías antes de ser aplicados. De allí, es donde entran en juego los modelos de integración identificados en la tabla 10, estos nos ayudan a aclarar el contexto dentro de la organización desde el punto de vista tecnológico, adicionalmente nos dan un conjunto de opciones a considerar al momento de definir de los requisitos no funcionales de un dominio en particular.

De lo anteriormente expuesto, se infiere en que si los procesos varían también sus métodos, es por esta razón, que se utiliza las líneas de producción de software y así gestionar la variabilidad en el proceso InDoCaS, estableciendo familias de procesos y creando líneas de proceso de software, iniciando por el correcto uso de similitudes entre proceso y aprovechando las diferencias entre ellos.

En la figura 36, se observa el análisis del dominio del proceso **InDoCaS** especializado, utilizando la notación SPEM 2 y resaltando las incorporaciones mencionadas anteriormente.

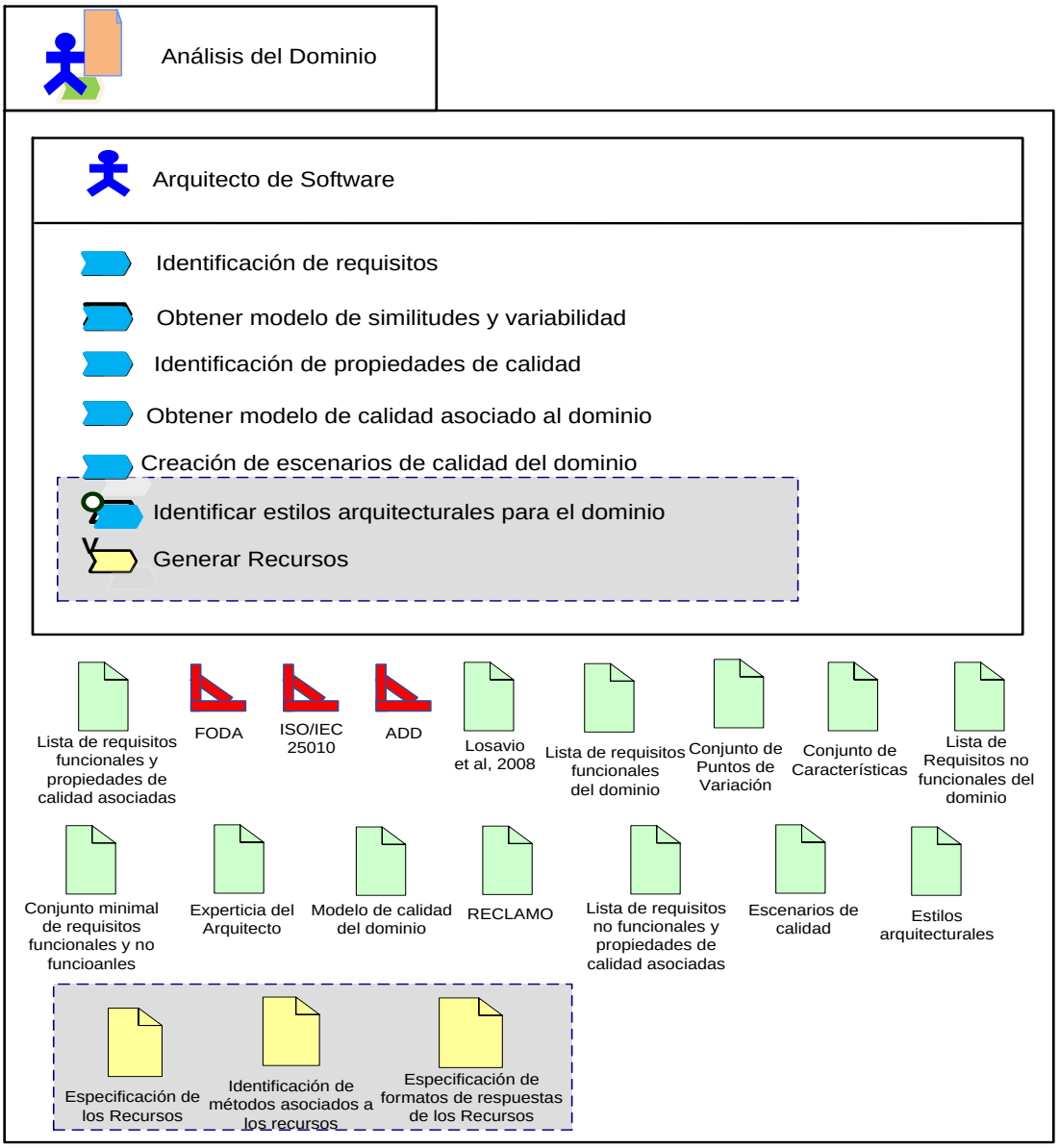


Figura 36. Análisis del dominio InDoCaS especializada. Fuente: La Autora (2012).

Una vez agregada esta nueva actividad y sus artefactos se puede asegurar la construcción de un modelo de integración para aplicaciones RESTful, basándonos en el enfoque de líneas de producción de software. Seguidamente en la figura 37, se muestra el diagrama de actividades de la disciplina análisis del dominio especializado, incorporando la actividad y sus artefactos.

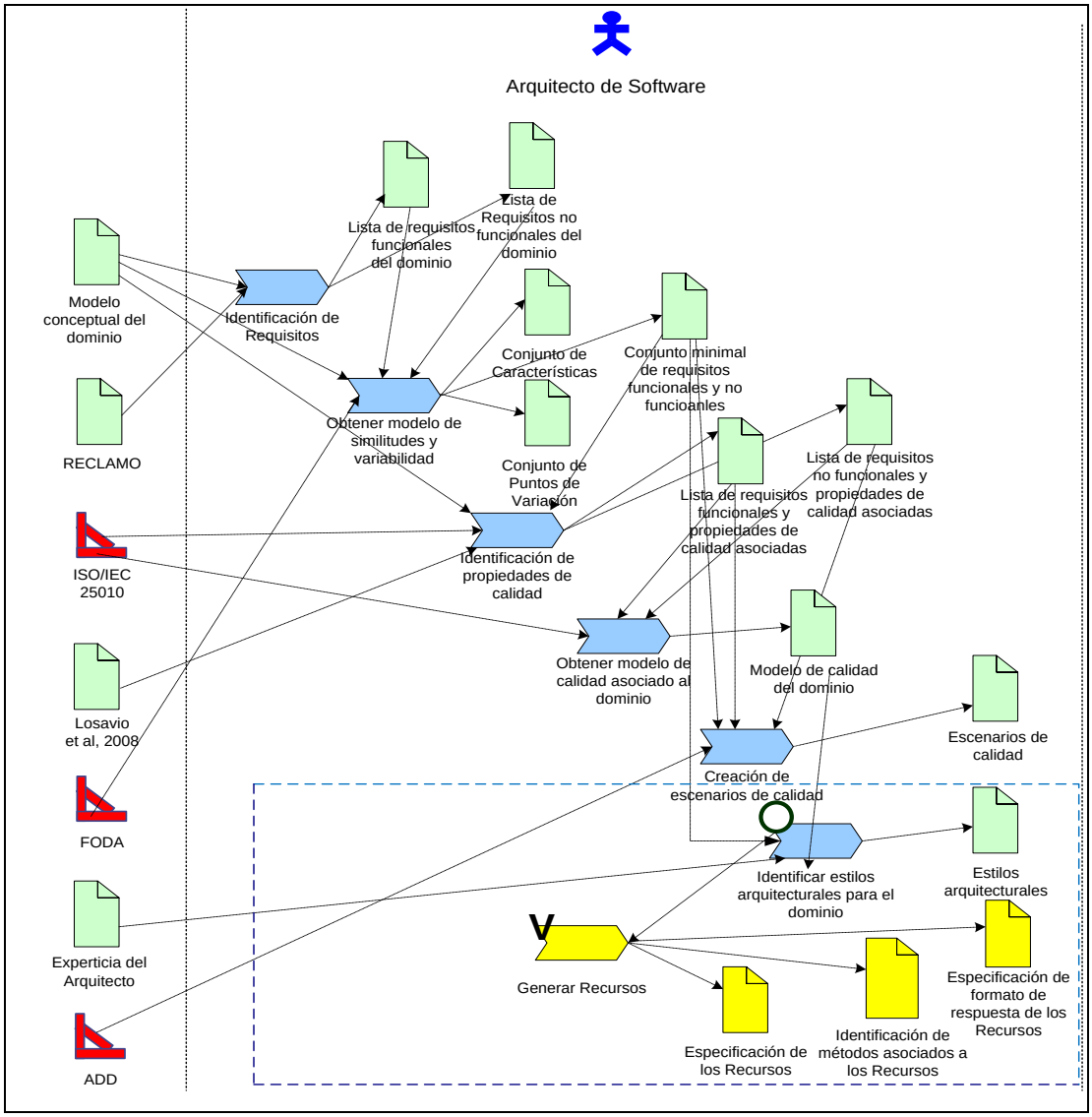


Figura 37. Diagrama de actividades para la disciplina análisis del dominio InDoCaS especializada. Fuente: La Autora (2012).

Definición de la actividad y sus artefactos

A continuación, se define la actividad adicional y cada uno de los artefactos involucrados en el proceso, así mismo las actividades y artefactos se ajustan al estándar definido por InDoCaS para facilitar su presentación y seguimiento.

Actividad A_06a: Generar Recursos

Anteriormente se ha mencionado, que una línea de producción tiene como objetivo principal reducir el tiempo, esfuerzo, costo y complejidad de crear y mantener los productos de la línea, mediante el manejo de los aspectos variables de los productos de la línea a través de los puntos de variación de los activos y los modelos de decisión.

Para tal propósito se hace variar la actividad "Identificar los estilos arquitecturales para el dominio" con el objetivo de conocer si dentro del artefacto "Estilos Arquitecturales" se encuentran las propiedades de adaptación a los cambios de contexto, en otras palabras, propiedades que indiquen que debe haber un proceso de ajuste en el producto. En caso, de existir estos cambios se ejecuta la actividad "Generar Recursos" que se ha identificado con el número A_06a, para reflejar la variación de la actividad A_06 de **InDoCaS** manteniendo la secuencia en su numeración.

En primer lugar, se construye el artefacto "Especificación de los recursos" con el propósito de describir el recurso a utilizar. Seguidamente se elabora la identificación de métodos utilizados por los recursos a partir del "Conjunto minimal de requisitos funcionales". Y por último, se elabora la especificación de formatos de salida de los recursos.

A continuación, se presenta en la tabla 11, un resumen de la definición de la actividad adicional "Generar Recursos".

Tabla 29. Actividad A_06a: Generar Recursos

InDoCaS	
ACTIVIDAD	**DESCRIPCIÓN**
Nombre de la Actividad:	*Generar Recursos*
Responsable:	Arquitecto de Software
Objetivo:	Generar los recursos de acuerdo al estilo arquitectural resultante.
Nro. Identificación:	A_06a
Tipo de documento generado:	Recurso
Técnica(s) utilizada(s):	Usar el conjunto de minimal de requisitos funcionales

	y meta-modelo REST propuesto por Schreier (2012)
Artefacto(s) de entrada:	Conjunto de minimal de requisitos funcionales y Estilos Arquitecturales.
Artefacto(s) de salida:	Especificación de los Recursos, Identificación de Métodos asociados a los recursos y Especificación de formatos de respuesta de los Recursos.

Fuente: La Autora (2012).

De igual manera, se presenta en tablas con formato **InDoCaS**, los artefactos generados por la actividad adicional. Es conveniente resaltar que se han identificado los artefactos adicionales con los números 6.1, 6.2 y 6.3 para indicar que son productos opcionales del proceso InDoCaS y no alterar la secuencia de numeración del mismo.

Artefacto6.1: Especificación del Recurso

El artefacto conocido como "Especificación de los Recursos" contiene información de los recursos a generar, entre ellos nombre del recurso, descripción del recurso, tipo de recurso, enlace con otros recursos. Es necesario profundizar en el tipo de recurso, este elemento ha sido tomado del Metamodelo de REST, propuesto por Schreier (2011), el cual define el tipo de recurso como un término para describir los aspectos comunes de múltiples recursos, en la sección de Anexos se encuentran los Tipos de Recursos definidos por Schreier (2011). A continuación, en la tabla 12, se observa el artefacto 6.1 denominado como "Especificación de los Recursos".

Tabla 12. Artefacto 6.1. Especificación de los Recursos

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	*Especificación delos Recursos*
Constructor:	Arquitecto de Software
Nro. Identificación ACTIVIDAD:	A_06ª
Nro. Identificación ARTEFACTO:	6.1

Formato Asociado:

Contrato de Uso			
Nombre del Recurso	***Descripción***	***Tipo de recurso***	***Enlace con otro recurso***

Nombre del Recurso	La descripción de cómo es su comportamiento "qué hace"	Indica tipo de recurso definidos por Schreier (2011).	Indica con que recursos se relaciona.

Fuente: La Autora(2012).

Artefacto 6.2: Identificación de métodos asociados a los recursos

El artefacto conocido como "Identificación de métodos asociados a los recursos" está relacionado con el mantenimiento de la interfaz uniforme, un método se identifica por su nombre, tiene que estar definido dentro todos los métodos existentes, como por ejemplo, los verbos HTTP. Un Recurso está asociado con un conjunto de métodos compatibles. El elemento método es el responsable de la conducta y determina el conjunto de medios producidos y consumidos. Además, cada método puede definir los parámetros que pueden ser contenidos en un medio consumido o en el identificador de recursos.

En este sentido, se muestra en la tabla 13 el artefacto 6.2 denominado "Identificación de métodos asociados a los recursos".

Tabla 13. Artefacto 6.2. Identificación de métodos asociados a los recursos

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	*Identificación de métodos asociados a los recursos*
Constructor:	Arquitecto de Software
Nro. Identificación ACTIVIDAD:	A_06ª
Nro. Identificación ARTEFACTO:	6.2
Formato Asociado:	

Contrato de Uso		
Nombre del Método	***Nombre del Recurso asociado***	***Parámetros***
Nombre del Método (GET, POST, PUT y DELETE)	Nombre del Recurso actual al cual está enlazado.	Define los parámetros necesarios que para lograr el comportamiento del método.

Fuente: La Autora (2012).

Artefacto 6.3: Especificación del Formato de Respuesta del Recurso

El artefacto conocido como "Especificación del Formato de Respuesta delos Recursos" tiene que ver con la representación de los datos. Los medios permiten la negociación de contenido. Una representación o formato de repuesta se identifica por un nombre, define su medio y los modelos de datos enviados por el servidor. La tabla 14, muestra el resumen del artefacto denominado "Especificación del Formato de Respuesta de los Recursos".

Tabla 14. Artefacto 6.3. Especificación del Formato de Respuesta delos Recursos

<table>
<tr><td colspan="2">InDoCaS</td></tr>
<tr><td>ARTEFACTO</td><td>DESCRIPCIÓN</td></tr>
<tr><td>Nombre:</td><td>Especificación del Formato de Respuesta delos Recursos</td></tr>
<tr><td>Constructor:</td><td>Arquitecto de Software</td></tr>
<tr><td>Nro. Identificación ACTIVIDAD:</td><td>A_06ª</td></tr>
<tr><td>Nro. Identificación ARTEFACTO:</td><td>6.3</td></tr>
<tr><td colspan="2">Formato Asociado:

<table>
<tr><th colspan="2">Contrato de Uso</th></tr>
<tr><th>Recurso</th><th>Formato de Respuesta</th></tr>
<tr><td>Nombre de Recurso</td><td>El formato de los datos utiliza tecnologías como: XML, JSON y RDF.</td></tr>
</table>
</td></tr>
</table>

Fuente: La Autora (2012).

Ejemplificación de la Propuesta

A continuación se mostrará la utilización del modelo de integración propuesto con el propósito de obtener una línea de producción de software para aplicaciones RESTful, aplicado en el Centro de Investigación y Desarrollo Empresarial (CIDESA) la cual es una empresa de servicios que tiene como misión fortalecer la gestión organizacional de sus clientes mediante la implementación de Tecnologías que les permitan innovar sus procesos administrativos, financieros y de recursos humanos. Esta empresa ha venido desarrollando proyectos tecnológicos en los procesos de

modernización de los Gobiernos Regionales y Estadales e Institutos Autónomos. Entre sus productos, se encuentra el Sistema de Información Automatizado para la integración de la Gestión Administrativa, Financiera y de Recursos Humanos (SIGA), el cual está compuesto por subsistemas siguientes: Contabilidad Financiera, Formulación de Presupuesto, Contabilidad Presupuestaria, Compras, Tesorería, Nómina y Personal, Bienes Nacionales, Obras y Contratos, entre Otros.

Así mismo CIDESA ejecuta proyectos que buscan integrar sus aplicaciones con procesos administrativos con el fin de mejorar sus los servicios que prestan a sus clientes. Por tal motivo, usaremos CIDESA y el proceso administrativo de generación de Compromiso en Presupuesto para obtener una línea de producción de software para aplicaciones RESTful, aplicando el proceso InDoCaS.

Del proyecto del sistema administrativo de CIDESA (SIGA), se presentan a continuación los requisitos levantados para el sistema de presupuesto:

- Generar compromiso
  - Generar reporte de Listado de Compromisos
  - Consulta de Compromisos

A continuación se aplicará el proceso InDoCaS al dominio de sistemas administrativos de CIDESA, en la tabla 15 se muestra el grupo de actividades que se realizaran en la disciplina de análisis del dominio. En la especificación se utiliza la nomenclatura establecida en el proceso.

Tabla 15. Actividades del modelo de procesos InDoCaS aplicado al Dominio

DOMINIO SISTEMAS ADMINISTRATIVOS			
Disciplina	**Actividad**	**Nombre de la actividad**	**Tablas Resultantes**
Análisis del Dominio	A_01	Identificación de requisitos	16,17
	A_02	Obtener modelo de similitudes y variabilidad	18,19,20
	A_03	Identificación de las propiedades de calidad	21,22
	A_04	Obtener modelo de calidad asociado al dominio	23

	A_05	Creación de escenarios de calidad del dominio	24
	A_06	Identificar estilos arquitecturales para el dominio	25
	A_06a	Generar Recursos	26, 27, 28

Fuente: La Autora (2012).

Actividades del Análisis del Dominio

Actividad 1. Identificación de Requisitos

La presente actividad se desarrolla usando el modelo de clasificación de requisitos RECLAMO, Chirinos, et al. (2004), el cual proporciona una taxonomía de requisitos que facilita la identificación de las clases de requisitos tanto los funcionales como los no funcionales. En este sentido, la actividad identificación de requisitos, genera dos artefactos los cuales son: listas de requisitos funcionales y no funcionales del dominio, en la tabla 16, se muestran los requisitos funcionales del Sistema Administrativo - Módulo de Presupuesto – Sección: Compromisos de CIDESA.

Tabla 16. Lista de requisitos funcionales del dominio

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

Nro. Identificación	**Descripción del requisito funcional**
1	Generar Compromisos
2	Generar Reportes de Listado de Compromisos
3	Generar Consultas de Compromisos

Fuente: La Autora (2012).

Tabla 17. Lista de requisitos no funcionales del dominio

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

Nro. Identificación	Reglas del Negocio asociadas al dominio	Requisitos no funcionales derivados de las reglas del negocio
	Políticas	
1	Los servicios se proveerán vía web.	- Cumplimiento de estándares, normativas con el fin de garantizar un buen servicio. - Debe funcionar en una plataforma de software libre. - Debe desarrollar bajo un estilo arquitectural ligero y poca configuración.
	Procesamiento	
2	Integridad de los datos	- La información se debe guardar completa y correctamente, cumpliendo con las reglas de negocio de la institución. - El sistema debe garantizar que la información sea resguardada de manera segura.
3	El servicio debe ser provisto vía web	- El acceso a los servicios siempre debe estar disponible, es decir, conexiones disponibles en todo momento y ancho de banda adecuado. -Mantener el rango de respuesta establecido. - Garantizar la entrega el formato de respuesta establecido.
4	Los datos deben estar consolidados	- La interfaces deben estar conectadas a la base de datos. - Los datos se obtendrán en varios formatos de respuesta.
	Implementación	
5	Escalable	- El sistema debe ser capaz de adaptarse a cambios.
6	Restricción de usuarios	- Solo tendrán acceso a los servicios solo aquellos usuarios registrados.

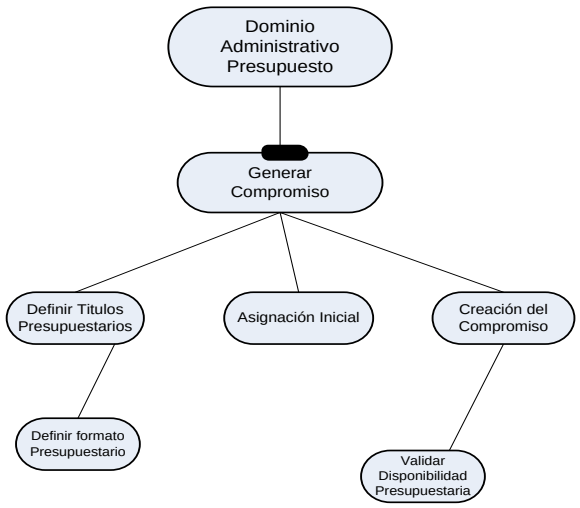
Fuente: La Autora (2012).

Actividad 2: Obtener modelo de similitudes y variabilidad

En la presente actividad, se busca lograr el conjunto de requisitos mínimos y obligatorios en el Sistema Administrativo desarrollado por CIDESA, mediante la realización de un diagrama FODA, del cual se generaran tres artefactos: conjunto de características, conjunto de de puntos de variación y conjunto minimal de requisitos

funcionales y no funcionales. A continuación, se observa en la tabla 18 el modelo de características.

Tabla 18. Conjunto de Características

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	*Conjunto de características*
Constructor:	Analista de Requisitos
Nro. Identificación ACTIVIDAD:	A_02
Nro. Identificación ARTEFACTO:	3
Formato Asociado: 	

Fuente: La Autora (2012).

Seguidamente se consiguen los puntos de variación, permitiendo describir dónde se encuentran diferencias entre las aplicaciones, además indican la variabilidad en las características, ver la tabla 19.

Tabla 19. Conjunto de puntos de variación

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

Característica	***Punto de Variación***
Definir los títulos Presupuestarios	Esta característica depende del formato presupuestario que se haya definido, el cual debería estar compuesto por una categoría y partida de acuerdo ley de presupuesto.
Validar Disponibilidad Presupuestaria	Esta característica depende del nivel de Disponibilidad que haya definido presupuestariamente.

Fuente: La Autora (2012).

Por otro lado, el siguiente artefacto representa el conjunto de requisitos mínimos y obligatorios de la familia de sistemas administrativos para el SIGA-CIDESA, el cual se muestra en la tabla 20.

Tabla. 20. Conjunto minimal de requisitos funcionales y no funcionales

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

Nro. Identificación	***Descripción del requisito funcional***
1	Generar Compromisos
2	Generar Reportes de Listado de Compromisos
3	Generar Consultas de Compromisos

Nro. Identificación	*Descripción del requisito no funcional*
1	- Cumplimiento de estándares, normativas con el fin de garantizar un buen servicio.
2	- Debe funcionar en una plataforma de software libre.
3	- Debe desarrollar bajo un estilo arquitectural ligero y poca configuración.
4	La información se debe guardar completa y correctamente, cumpliendo con las reglas de negocio de la institución.
5	El sistema debe garantizar que la información sea resguardada de manera segura.
6	- El acceso a los servicios siempre debe estar disponible, es decir, conexiones disponibles en todo momento y ancho de banda adecuado.
7	-Mantener el rango de respuesta establecido.
8	- Garantizar la entrega el formato de respuesta establecido.
9	- Los datos se obtendrán en varios formatos de respuesta.
10	- El sistema debe ser capaz de adaptarse a cambios.
11	- Solo tendrán acceso a los servicios solo aquellos usuarios registrados.

Fuente: La Autora (2012).

Actividad 3: Identificación de propiedades de Calidad

En lo tocante a esta actividad, permite identificar las propiedades de calidad asociadas al artefacto "Conjunto minimal de Requisitos funcionales y no funcionales", el cual se planteó en la tabla 20, estas propiedades están ajustadas a las definiciones establecidas en la terminología estándar ISO/IEC 2010. En este orden de ideas, la actividad Identificación de propiedades de Calidad genera dos artefactos: lista de requisitos funcionales y lista de requisitos no funcionales con sus propiedades de calidad asociadas, las cuales se representaran en las tablas 21 y 22 respectivamente.

Tabla 21. Lista de requisitos funcionales con sus propiedades de calidad asociada

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

Formato Asociado:

Nro. Identificación	*Requisitos Funcionales*	*Características de calidad ISO/IEC*	*Atributos (Características)*

		25010	
1	Generar Compromisos	**Confiabilidad** - Disponibilidad **Eficiencia** - Comportamiento en el tiempo **Funcionalidad** - Preciso **Seguridad** - Integridad **Usabilidad** - Operabilidad	- Atributo: Tiempo de espera. Métrica: valor en rango [1..5] - Atributo: ejecución en relación al uso del recurso. - Atributo: tiempo de completitud de tareas Métrica: valor en rango [1..10] -Protección para el origen de datos. -Protección para el proceso de autenticación Métrica: un número en el rango [0..1] - Proporción de las funcionalidades que son entendidas correctamente por el usuario.
2	Generar Reportes de Listado de Compromisos	**Confiabilidad** - Disponibilidad **Eficiencia** - Comportamiento en el tiempo **Funcionalidad** - Preciso **Usabilidad** - Operabilidad	- Atributo: Tiempo de espera. Métrica: valor en rango [1..5] - Atributo: ejecución en relación al uso del recurso. - Atributo: Tiempo de completitud de las tareas. Métrica: un numero en el rango [1..10] -Proporción de las funcionalidades que son entendidas correctamente por el usuario.
3	Generar Consultas de Compromisos	**Confiabilidad** - Disponibilidad **Eficiencia** - Comportamiento en el tiempo **Funcionalidad** - Preciso **Usabilidad** - Operabilidad	- Atributo: Tiempo de espera. Métrica: valor en rango [1..5] - Atributo: ejecución en relación al uso del recurso. - Atributo: Tiempo de completitud de las tareas. Métrica: un numero en el rango [1..10] -Proporción de las funcionalidades que son entendidas correctamente por el usuario.

Fuente: La Autora (2012).

Seguidamente, se describen las propiedades de calidad asociadas a los requisitos no funcionales, los cuales se pueden observar en la tabla 22.

Tabla 21. Lista de requisitos no funcionales con su propiedades de calidad asociada

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

Formato Asociado:

Nro. Identificación	***Reglas del negocio asociadas al Dominio***	***Requisitos no funcionales derivado de las reglas del negocio***	***Propiedades de calidad asociado a los requisitos no funcionales ISO/IEC 25010***	***Atributo (Característica) ISO/IEC 13236***
	Políticas			
1	Los servicios se proveerán vía web.	- Cumplimiento de estándares, normativas con el fin de garantizar un buen servicio. - Debe funcionar en una plataforma de software libre. - Debe desarrollar bajo un estilo arquitectural ligero y poca configuración.	**Confiabilidad** - Disponibilidad **Portabilidad** - Instalabilidad - Adaptabilidad **Mantenibilidad** - Cambialidad	Atributo: tiempo de servicio agregado. Se refiere a la proporción se servicio que la aplicación tiene disponible. Métrica un número de rango [0..1] - Proporción de las funcionalidades que pueden ser adoptadas por el usuario -Atributo: Presencia de mecanismo. -Métrica Booleano. - Adaptación a los cambios de ambientes.
	Procesamiento			
2	Integridad de los datos	- La información se debe guardar completa y correctamente, cumpliendo con las reglas de negocio de la institución.	**Funcionalidad** -Preciso	Atributo: Tiempo de completitud de las tareas. Métrica: valor en rango [0..10] -Presencia de

		- El sistema debe garantizar que la información sea resguardada de manera segura.	**Confiabilidad** - Recuperabilidad -Tolerancia a Fallas	mecanismo: disponible. Métrica Booleano
3	El servicio debe ser provisto vía web.	- El acceso a los servicios siempre debe estar disponible, es decir, conexiones disponibles en todo momento y ancho de banda adecuado. -Mantener el rango de respuesta establecido. - Garantizar la entrega el formato de respuesta establecido.	**Confiabilidad** - Disponibilidad **Eficiencia** -Comportamiento en el tiempo **Eficiencia** -Utilización de los recursos	Atributo: tiempo de servicio agregado, para indicar la disponibilidad de la aplicación. Métrica: un valor entre [0..1] Atributo: Tiempo de espera. Tiempo en segundos Métrica: valor en rango [1..10] Proporción de respuesta deseada.
4	Los datos deben estar consolidados.	- La interfaces deben estar conectadas a la base de datos. - Los datos se obtendrán en varios formatos de respuesta.	**Funcionalidad** -Apropiado -Preciso **Usabilidad** -Operabilidad	Atributo: Tiempo de completitud de tareas. Proporción de las funcionalidades que son entendidas correctamente por los usuarios.
	Implementación			
5	Escalable	- El sistema debe ser capaz de adaptarse a cambios.	**Portabilidad** -Instalabilidad - Adaptabilidad	Proporción de las funciones que pueden ser adoptadas por el usuario
6	Restricción de usuarios	- Solo tendrán acceso a los servicios solo aquellos usuarios registrados.	**Seguridad** -Autenticidad -Confidencialidad	-Presencia de mecanismo: disponible. Métrica Booleano -Protección para el origen de datos. -Protección para el proceso de autenticación Métrica: un número en el rango [0..1]

Fuente: La Autora (2012).

En la siguiente actividad, se utiliza la información generada en las tablas 21 y 22 para definir el modelo de calidad asociado al dominio de Sistemas Administrativos.

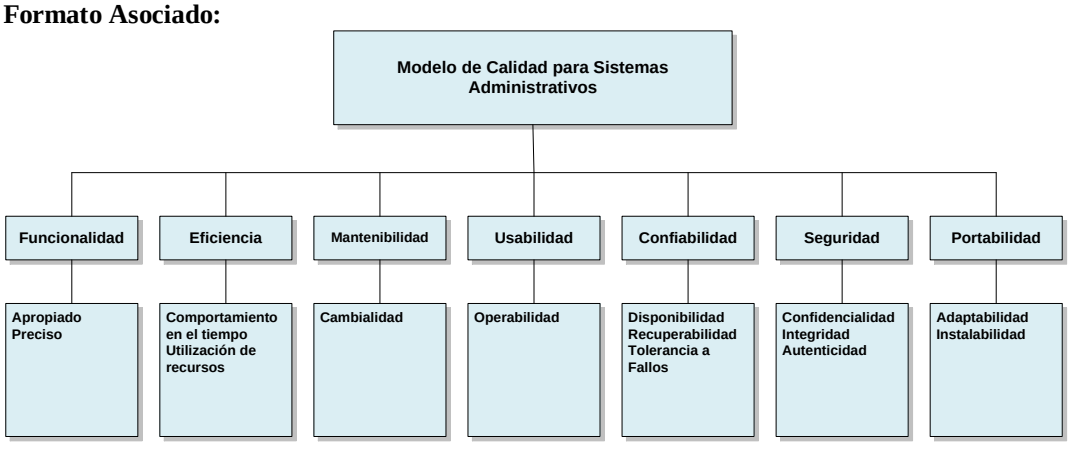
Actividad 4: Obtener modelo de calidad asociado al dominio

Esta actividad permite obtener el artefacto denominado modelo de calidad del dominio el cual representa la calidad de un producto de software del dominio como una expresión acerca de la capacidad del software para ejecutarse y mantener el nivel de servicio especificado. En este sentido, las propiedades de calidad muestran el grado con el cual el software es capaz de proporcionar y mantener dichos servicios. En resumen, el modelo de calidad para el dominio de las aplicaciones de Sistemas administrativos viene representado mediante la tabla 23.

Tabla 23. Modelo de Calidad del Dominio

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

Formato Asociado:



Fuente: La Autora (2012).

Actividad 5: Creación de los escenarios de Calidad del dominio

Esta actividad permite obtener los escenarios de calidad del dominio, enfocándose en las características que dependen de la arquitectura en concordancia con las características de calidad basadas en el estándar ISO/IEC 25010. Asimismo la presente actividad buscar crear escenarios de calidad para representar los intereses de los grupos de evaluación y confirmar que se han evaluado los requisitos deseados de calidad. A continuación, en la tabla 24 se aprecian los escenarios de calidad.

Tabla 24. Escenarios de Calidad

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	*Escenarios de calidad.*
Constructor:	Arquitecto de Software y Analista de Requisitos
Nro. Identificación ACTIVIDAD:	A_05
Nro. Identificación ARTEFACTO:	9

Formato Asociado:

Nro. Escenario	***Requisitos Arquitecturales***	***Propiedades de calidad asociado a los requisitos no funcionales ISO/IEC 25010***	***Atributo (Característica) ISO/IEC 13236***
1	- Cumplimiento de estándares, normativas con el fin de garantizar un buen servicio. - Debe funcionar en una plataforma de software libre. - Debe desarrollarse bajo un estilo arquitectural ligero y poca	**Confiabilidad** - Disponibilidad **Portabilidad** - Instalabilidad - Adaptabilidad **Mantenibilidad** - Cambialidad	Atributo: tiempo de servicio agregado. Se refiere a la proporción se servicio que la aplicación tiene disponible. Métrica un número de rango [0..1] - Proporción de las funcionalidades que pueden ser adoptadas por el usuario -Atributo: Presencia de mecanismo. -Métrica Booleano. - Adaptación a los cambios de ambientes.

	configuración.		
2	- La información se debe guardar completa y correctamente, cumpliendo con las reglas de negocio de la institución. - El sistema debe garantizar que la información sea resguardada de manera segura.	**Funcionalidad** -Preciso **Confiabilidad** - Recuperabilidad -Tolerancia a Fallas	Atributo: Tiempo de completitud de las tareas. Métrica: valor en rango [0..10] -Presencia de mecanismo: disponible. Métrica Booleano
3	- El acceso a los servicios siempre debe estar disponible, es decir, conexiones disponibles en todo momento y ancho de banda adecuado. -Mantener el rango de respuesta establecido. - Garantizar la entrega el formato de respuesta establecido.	**Confiabilidad** - Disponibilidad **Eficiencia** -Comportamiento en el tiempo **Eficiencia** -Utilización de los recursos	Atributo: tiempo de servicio agregado, para indicar la disponibilidad de la aplicación. Métrica: un valor entre [0..1] Atributo: Tiempo de espera. Tiempo en segundos Métrica: valor en rango [1..10] Proporción de respuesta deseada.
4	- La interfaces deben estar conectadas a la base de datos. - Los datos se obtendrán en varios formatos de respuesta.	**Funcionalidad** -Apropiado -Preciso **Usabilidad** -Operabilidad	Atributo: Tiempo de completitud de tareas. Proporción de las funcionalidades que son entendidas correctamente por los usuarios.
5	- El sistema debe ser capaz de adaptarse a cambios.	**Portabilidad** -Instalabilidad - Adaptabilidad	Proporción de las funciones que pueden ser adoptadas por el usuario
6	- Solo tendrán acceso a los servicios solo aquellos usuarios registrados.	**Seguridad** -Autenticidad -Confidencialidad	-Presencia de mecanismo: disponible. Métrica Booleano -Protección para el origen de datos. -Protección para el proceso de autenticación Métrica: un número en el rango [0..1]

Fuente: La Autora (2012).

Actividad 6. Identificar los estilos arquitecturales para el dominio

Esta actividad consiste en identificar los estilos arquitecturales para el dominio, es decir; encontrar los estilos arquitecturales que soportan el diseño arquitectural de las aplicaciones de Sistemas Administrativos en cual se representan los componentes y sus relaciones entre ellos, incluyendo las restricciones, asociaciones y las reglas de diseño. Esto implica seleccionar una librería de los patrones aplicables, en otras palabras, seleccionar aquellos patrones arquitecturales que respondan a la prioridad de los requisitos no funcionales. Un patrón es aplicable si se ajusta a la descripción funcional del problema y solventa un requisito no funcional.

Ahora bien, la identificación de los estilos arquitecturales genera el artefacto estilos arquitecturales de Sistemas Administrativos, para esto tiene como entrada el conjunto mínimal de requisitos no funcionales y el modelo de calidad del dominio, en donde existen características de calidad que deben ser garantizadas por los estilos arquitecturales seleccionados, en particular requisitos como:

- Debe desarrollarse bajo un estilo arquitectural ligero y poca configuración. **Mantenibilidad** (Cambiabilidad).
- La información se debe guardar completa y correctamente, cumpliendo con las reglas de negocio de la institución. **Funcionalidad** (Preciso).
- Los datos se obtendrán en varios formatos de respuesta. **Usabilidad** (Operabilidad).
- El acceso a los servicios siempre debe estar disponible, es decir, conexiones disponibles en todo momento y ancho de banda adecuado. **Confiabilidad** (Disponibilidad).
- Mantener el rango de respuesta establecido. **Eficiencia** (Comportamiento en el tiempo)

Por lo anteriormente expuesto, se infiere que el estilo arquitectural que más se adapta en la solución de estos requisitos es el estilo arquitectural REST, porque está conformado por varias características entre ellas, Fielding (2000):

- **Arquitectura Cliente-Servidor**, la cual mejora la portabilidad de la interfaz de usuario a través de múltiples plataformas y mejora la escalabilidad mediante la simplificación de los componentes del servido.
- **No guarda Estado**, lo cual induce a obtener propiedades de visibilidad, fiabilidad y escalabilidad. La visibilidad mejora debido a que un sistema de monitoreo no tiene que mirar más allá de un dato de solicitud única, a fin de determinar la naturaleza exacta de la solicitud. Ha mejorado la confiabilidad ya que facilita la tarea de recuperación de fallos parciales. La escalabilidad se mejora debido a tener que guardar el estado entre peticiones permite que el componente del servidor de forma rápida los recursos libres, y además simplifica la implementación porque el servidor no tiene que administrar el uso de los recursos en las distintas solicitudes.
- **Maneja Caché**, obteniendo como ventaja el potencial para eliminar parcialmente o completamente algunas interacciones, mejora de la eficiencia, la escalabilidad y el rendimiento percibido del usuario al reducir la latencia promedio de una serie de interacciones.
- **Interfaz uniforme**, se basa en el principio de ingeniería de software de generalidad de la interfaz de componente, la arquitectura general del sistema se simplifica y la visibilidad de las interacciones se mejora; la interfaz REST está diseñado para ser eficiente en la transferencia de datos hipermedias. Con el fin de obtener una interfaz uniforme, múltiples restricciones arquitectónicas son necesarios para guiar el comportamiento de los componentes. REST se define por cuatro restricciones de interfaz: la identificación de los recursos, la manipulación de los recursos a través de las representaciones; auto mensajes descriptivos y, hipermedia como el motor de estado de la aplicación.

Por otro lado, existen frameworks de trabajo que utilizan el estilo arquitectural REST, y que además incorporan otros patrones como Modelo-Vista-Controlador, como es el caso de Ruby On Rails.

En consecuencia, se utilizará el estilo arquitectural REST, como parte del artefacto Estilo arquitectural del Dominio, el cual se observa en la tabla 25.

Tabla 25. Estilos Arquitecturales

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	*Estilos arquitecturales.*
Constructor:	Arquitecto de Software y Analista de requisitos
Nro. Identificación ACTIVIDAD:	A_06
Nro. Identificación ARTEFACTO:	10
REST (Representational Transfer State)	

Fuente: La Autora (2012).

Puesto que el estilo arquitectural seleccionado para la arquitectura de Sistemas Administrativo es REST, las aplicaciones se desarrollaran bajo el perfil de aplicaciones RESTful, porque están basadas en el estilo arquitectural REST, trayendo como consecuencia interfaces que se conocen como recurso, de los cuales se pueden obtener artefactos que pueden ser entradas para el diseño del dominio. En este sentido se aborda la actividad "Generar Recursos".

Actividad 6a: Generar Recursos

Esta actividad se enfoca en definir las restricciones de interfaces, generando como artefactos: la Especificación de los recursos, Identificación de los métodos asociados a los recursos y Especificación de formatos de salida de los recursos. A continuación, la Especificación de los Recursos se presenta en la tabla 26.

Tabla 26. Especificación de los Recursos del dominio

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	*Especificación delos Recursos*
Constructor:	Arquitecto de Software
Nro. Identificación ACTIVIDAD:	A_06ª
Nro. Identificación ARTEFACTO:	11

Formato Asociado:

Contrato de Uso			
Nombre del Recurso	***Descripción***	***Tipo de recurso***	***Enlace con otro recurso***
compromiso	Este recurso se encarga de generar un compromiso presupuestario	Recurso Principal	No aplica

Fuente: La Autora (2012).

Seguidamente, se identifican los métodos a utilizar, en la tabla 27 se puede apreciar el artefacto Identificación de Métodos asociados a los recursos.

Tabla 27. Identificación de Métodos asociados a los recursos del dominio

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	*Identificación de métodos asociados a los recursos*
Constructor:	Arquitecto de Software
Nro. Identificación ACTIVIDAD:	A_06ª
Nro. Identificación ARTEFACTO:	12

Formato Asociado:

Contrato de Uso		
Nombre del Método	***Nombre del Recurso asociado***	***Parámetros***
GET	Compromiso	Numero de Solicitud, Rif del Beneficiario, código presupuestario y monto de la imputación presupuestaria.

Fuente: La Autora (2012).

Por último, se presenta el artefacto Especificación del formato de respuesta de los Recursos, ver la tabla 28.

Tabla 28. Especificación del Formato de Respuesta de los Recursos del dominio

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	*Especificación del Formato de Respuesta de los Recursos*
Constructor:	Arquitecto de Software
Nro. Identificación ACTIVIDAD:	A_06ª
Nro. Identificación ARTEFACTO:	13

Formato Asociado:

Contrato de Uso	
Recurso	***Formato de Respuesta***
Compromiso	XML

Fuente: La Autora (2012).

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

Una vez desarrollado el Modelo de Integración basado en líneas de producción de software para Aplicaciones RESTful y haber realizado un caso de estudio a CIDESA, se llegó a las siguientes conclusiones:

- ✓ En primer lugar, el presente estudio se enfocó en caracterizar los modelos de integración de IAE, para luego identificar cuales modelos se adecuaban a Aplicaciones RESTful, para esto se seleccionaron características, Subcaracterísticas y métricas que por la naturaleza de las aplicaciones RESTful deberían estar presentes en las organizaciones que requieran de las mismas, esta selección me llevó a determinar los modelos asociados a las características IAE, entre los modelos encontrados están: Web, Onda, Anillo y Célula, estos nos ayudan a aclarar el contexto dentro de la organización desde el punto de vista tecnológico, adicionalmente nos dan un conjunto de opciones a considerar al momento de definir de los requisitos no funcionales de un dominio en particular.
- ✓ En segundo lugar, el estudio se enfocó en la creación de una línea de producción de software para aplicaciones RESTful, centrándose en la disciplina de la Ingeniería de Dominio, para lo cual se utilizó el proceso para la ingeniería de dominio basado en calidad de software **InDoCaS**, en su fase de Análisis del Dominio, fase en la cual se especializo la propuesta, para llevar a cabo actividades que compongan una línea de producción de software para Aplicaciones RESTful.
- ✓ De acuerdo, a la especialización realizada al proceso **InDoCaS** fue necesario incorporar actividades correspondientes a la fase de Análisis del Dominio en su Actividad 6, Estilos Arquitecturales instanciando los principios del estilo

arquitectural REST, considerándose este un aporte significativo de la investigación. Se agregó la actividad "Generar Recursos" de la cual se obtienen los siguientes artefactos: "Especificación de los Recursos", "Identificación de los Métodos Asociados a los Recursos" y "Especificación del formato de Respuesta de los Recursos".

- ✓ Por otro lado, la creación de una línea de producción para aplicaciones RESTful, genera beneficios en las aplicaciones RESTful debido está fundamentada en el proceso **InDoCaS**, el cual permite construir la línea de productos basado el modelo de calidad 25010, es decir, se formaliza la especificación de los requisitos de calidad, lo que trae como consecuencia que las aplicaciones RESTful creadas a partir de esta línea de producción tengan intrínsecas las propiedades de calidad necesarias, logrando obtener mayor escalabilidad, interoperabilidad y fiabilidad entre muchas otras.
- ✓ Adicionalmente los beneficios asociados a las líneas de producción de software como tiempo de entrega del producto, reducción de costos, crecimiento controlado de las aplicaciones, reducción de tasas de defectos y mejoras de calidad en los productos son trasladados a los Sistemas que estén basados en Aplicaciones RESTful.
- ✓ Es necesario resaltar, que la realización del caso de estudio aplicando la propuesta a los Sistemas administrativos más específicamente el Modulo de Presupuesto, hizo notar como la línea de producción para aplicaciones RESTful, se acopló satisfactoriamente de acuerdo a los requisitos especificados por la empresa CIDESA. Para la realización de este caso se estudio se seleccionó el proceso de generación de Compromisos y se levantaron los requisitos funcionales y no funcionales del mismo, llegando a establecer su propiedades de calidad asociadas, variabilidad y finalmente seleccionando estilo arquitectural adecuado del cual se derivó la actividad Generar recursos.

Recomendaciones

Tomando en cuenta el estudio realizado, conclusiones y hallazgos, se recomienda lo siguiente:

- ✓ Determinar el método de evaluación a ser aplicado a la especificación de los modelos de integración para Aplicaciones RESTful para la validación del mismo, ya que no fue alcance de esta investigación. Una vez validado la especificación de los modelos de integración aplicarlo en casos de estudio.
- ✓ Continuar el desarrollo de la línea de producción de software para aplicaciones RESTful en sus etapas de diseño en implementación del dominio, de modo que se pueda completar el proceso de ingeniería de dominio.
- ✓ Aplicar la propuesta a otros dominios, esto puede proporcionar información valiosa. Mediante el estudio de un grupo heterogéneo, la experiencia del usuario puede ser evaluada para comprobar en qué medida el sistema se comporta como se esperaba.

GLOSARIO DE TÉRMINOS

Activos de Software: Colección de partes de software (requisitos, diseños, componentes, casos de prueba, arquitecturas, entre otros.) que se configuran y componen de una manera prescrita para producir los productos de la línea.

ADD (Attribute-Driven Design Method): Método de diseño dirigido por atributos: Enfoque para definir una arquitectura de software, al basar el proceso de diseño sobre los atributos de calidad del sistema.

Análisis del dominio: Proceso para capturar y representar la información sobre las aplicaciones en un dominio, específicamente las características comunes, las variaciones, y las razones de la variación.

Arquitectura de Software: Infraestructura del sistema, que incluyen elementos de software, las propiedades externamente visibles de esos elementos, y las relaciones entre ellos.

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 rendimiento, seguridad, modificabilidad, confiabilidad y facilidad de uso tienen una influencia significativa en la arquitectura de software de un sistema.

Característica (feature): Abstracción funcional distintiva e identificable que puede ser empaquetada, implementada, probada, distribuida y mantenida. Son, por tanto, objetos de primera clase en el desarrollo de software orientado a un dominio.

Componente: Entidades tales como clientes, servidores, bases de datos, filtros y capas de un sistema jerárquico. Son además, una parte encapsulada del sistema de software, donde cada uno tiene una interfaz.

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

Dominio: Área de conocimiento o de actividad caracterizada por un conjunto de conceptos y terminología comprensible para los profesionales en esa área.

Estilo Arquitectural: Una especialización de los elementos y tipos de relación, en combinación con un conjunto de restricciones sobre cómo pueden ser utilizados. Véase patrón arquitectural.

Familia de productos de software: Conjunto de productos de software asociados a un dominio determinado.

JSON (JavaScript Object Notation): es un formato ligero para el intercambio de datos, ha surgido como alternativa a XML en AJAX, debido a que es mucho más sencillo escribir un analizador semántico del mismo.

Patrón Arquitectural: Descripción de elementos y tipos de relación, en combinación con un conjunto de restricciones sobre cómo se utilizan.

Recurso: cualquier cosa que sea lo suficientemente importante para ser referenciado como una cosa en sí.

RDF (Resource Descripción Framework): es un framework para metadatos desarrollado W3C, se basa en convertir las declaraciones de recursos en expresiones con la forma sujeto (recurso)- predicado (propiedad)-objeto (valor).

Stakeholder (Actor del Sistema): Persona que tiene interés especial sobre la arquitectura.

Tradeoff (Relaciones de intercambio): Propiedad que afecta a más de un atributo y es un punto de sensibilidad de más de un atributo.

XML (eXtensibleMarkupLanguaje): se define como un metalenguaje extensible de etiquetas, el cual permite definir la gramática de lenguajes específicos y la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

REFERENCIAS BIBLIOGRAFICAS

Balestrini, M. 2006. Como se elabora el proyecto de investigación. BL Consultores Asociados. Caracas - Venezuela.

Barbosa, A. 2010. Testing de Servicios Web para Líneas de Productos Software. Proyecto de Fin de Carrera Ingeniería Informática publicado, Universidad Rey Juan Carlos. España. Extraído el 15 de Junio del 2012 desde http://eciencia.urjc.es/bitstream/10115/4507/1/Memoria-AlbertoBarbosaLeon.pdf

Bass, L., Clements, P. y Kazman, R. 2003 Software Architecture in Practice. SEI Series in Software Engineering. Editorial Addison-Wesley.

Brown, L. 2000. Integration Models: Templates for Business Transformation. Editorial Sams. USA

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. Tesis Doctoral. Universidad Central de Venezuela. Caracas.

Chirinos L., Losavio F., Matteo A. 2004. Identifying Quality-Based Requierements. Informations system Management. Editorial Auerbach Publications.

Clements, P. y Northrop, L.M. 2001. Software product lines: Practices and Patterns. Editorial Addison-Wesley.

Club BPM 2011. Business Process Management. El Libro del BPM 2011. Tecnologías, Conceptos, Enfoques Metodológicos y Estándares. Extraído el 10 de Junio del 2012 desde http://books.google.co.ve/books?id=8Hc5Q00RXEkC&printsec=frontcover&hl=es#v=onepage&q&f=false

Czarnecki, K. y Eisenecker, U.2000. Generative Programming: Methods, Techniques, and Applications. Editorial Addison-Wesley.

Díaz, O. y Trujillo, S. 2010. Líneas de Producto de Software. En Piattini, M. y Garzas, J. (Eds). Fábricas de Software: experiencias, tecnologías, y organización. Editorial RA-MA. Capitulo 3.

Fielding, R. 2000. Architectural styles and the design of network-based software architectures. Tesis doctoral, University of California. Irvine, California.

Frantz, R, Reina, A. y Corchuelo, R. 2011. A Domain-Specific Language to Design Enterprise Application Integration Solutions. Revista International Journal of Cooperative Information System. Vol. 2, N° 2, pp. 143-176.

Gómez, E. 2011. Modelo Arquitectural para Aplicaciones Móviles usando el enfoque de Líneas de Producción Dinámica de Software. Trabajo de Grado. Universidad Centroccidental "Lisandro Alvarado". Barquisimeto. 112 p.

Hinchcliffe, D. 2005. The hidden battle between web services: REST versus SOAP. Imagen extraída el 01 de Julio del 2012 desde http://hinchcliffe.org/archive /2005/02/12/171.aspx

Hofmeister, C., Kruchten, P., Nord, R., Obbink, H., Ran, A., America, P. 2007. A general model of software architecture design derived from five industrial approaches. The journal of system and software.Pág. 106-126. Elsevier inc.

Hurtado de Barrera, J. 2008. El Proyecto de Investigación. Comprensión holística de la metodología y la investigación. Ediciones Quirón. SextaEdición. Caracas, Venezuela.

ISO/IEC 25010 2009. Software Engineering–Software Product Quality Requirements and Evaluation (SQuaRE) – Quality model and guide.

Kang, K., Cohen, S., Hess, J., Novak, W, Peterson, A. 1990. Feature-Oriented Domain Analysis (FODA) Feasibility Study.Software Engineering Institute, Carnegie Mellon University. Pittsburgh.

Krueger, C. 2006. Introduction to the emerging Practice Software Product Line Development. Revista Methods & Tools.Vol. 14 N° 3.

Linthicum, D. 1999. Enterprise Application Integration. Editorial Addison Wesley. Estados Unidos.

Losavio, F., Matteo, A., Rahamut, R. 2008. Unifying Quality Standards to Capture Architectural Knowledge for Web Services Domain. DMS 2008: 65-70.

Mendoza, L. Pérez, M. y Titaeva, I. 2009. Modelo basado en características para identificar modelos de integración de aplicaciones empresariales. Estudio de caso en Venezuela. Revista Avances en Sistemas e Informática. Vol 6. No. 3, Venezuela.

Montilva, J. 2006. Desarrollo de Software Basado en Líneas de Productos de Software. Conferencia DVP, IEEE Computer Society Región 9, Capítulo Argentina. Extraído el 30 de Julio del 2012 desde http://www.ieee.org.ar/downloads/2006-montilva-productos-prt.pdf

Montilva, J., Arapé, N. y Colmenares, J. 2003. Desarrollo de Software Basado en Componentes. IV Congr. De Automatización y Control (CAC03). Mérida.

Navarro, R 2007. REST vs Web Services. Departamento de Sistemas Informáticos y Computación (DSIC). Universidad Politécnica de Valencia. Valencia (España). 19 p.

Northrop, L. 2008. Software Product Lines Essentials. Presentación de Software Engineering Institute Carnegie Mellon University. Pittsburgh. Extraído el 15 deJunio del 2012 desde http://www.sei.cmu.edu/library/assets/spl-essentials.pdf

Oransa, O. 2010. RESTful Web Services and ROA.Ponencia presentada en Java Developer Conference, Egipto. Extraído el 02 de Julio del 2012 desde http://jdc2010.egjug.org/sites/default/files/sites/jdc2010.egjug.org/files/RESTful%20WS%20and%20ROA_0.pdf

Paramio, C. 2007. REST: Representational State Transfer, en Ruby on Rails. Ponencia presentada en Proceedings of the FLOSS International Conference 2007. Cádiz. Extraído el 18 de junio del 2012 desde http://libros.metabiblioteca.org/handle/001/128

Pardo, J. y Valero, J. 2011. Propuesta de Interacción entre Sistemas. RevistaGeomática UD.GEO. N° 5:29-38.

Pautasso, C. 2008. REST vs. SOAP: Making the Right Architectural Decision. Ponencia presentada en SOA Symposium, Amsterdam. Extraído el 02 de julio del 2012 desde http://www.jopera.org/files/soa-amsterdam-restws-pautasso-talk.pdf

Pautasso, C., Zimmermann, O. y Leymann, F. 2008. RESTful Web Services vs, "Big" Web Services: Making the Right Architectural Decision. Revista WWW '08: Proceeding of the 17th international conference on World Wide Web. USA. pp. 805-814.

Pohl, K., Böckle, G. y Van der Linden, F. 2005. Software Product Line Engineering. Editorial Springer.Berlín

Pressman, R. 2005. Ingeniería del Software: Un Enfoque Práctico. Sexta edición. Editorial McGraw-Hill. New york

Richardson, L. y Ruby, S. 2007. RESTful Web Services. Editorial O'Reilly. United States of America.

Ruiz, F. y Verdugo, J. 2008. Guía de Uso de SPEM 2 con EPF Composer. IDC. Universidad de Castilla – La Mancha. Escuela Superior de Informática Departamento de Tecnologías y Sistemas de Información. España.

Ruh, W., Maginnis, F. y Brown, W. 2001. Enterprise Application Integration: A Wiley Tech Brief. Editorial John wiley&Sons. Extraído el 20 de junio del 2012 desde http://books.google.co.ve/books?id=s7FQg8L6scgC&printsec=frontcover&dq=Enterprise+Application+Integration&hl=es&sa=X&ei=ll7vT6XvF-200QHFh837Ag&ved=0CEYQ6wEwAg#v=onepage&q=Enterprise%20Application%20Integration&f=false

Schreier, S. 2011. Modeling RESTful applications. Proceedings of the Second International Workshop on RESTful Design. WS-REST '11, New York, NY, USA, ACM (2011) 15–21. Extraído el 30 de Junio del 2012 desde http://www.ws-rest.org/2011/proc/a4-schreier.pdf

Sharma, R., Stearns, B. y Ng, T. 2001. J2EE Connector Architecture and Enterprise Application Integration. Editorial Addison Wesley. United States of America. Extraído el 16 de Junio del 2012 desde http://books.google.co.ve/books?id=8D2EpHum3O0C&printsec=frontcover&dq=J2EE+Connector+Architecture+and+Enterprise+Application+Integration&hl=es&sa=X&ei=HV7vT7bJBcrm0gGpmIT7Ag&ved=0CC4Q6AEwAA#v=onepage&q=J2EE%20Connector%20Architecture%20and%20Enterprise%20Application%20Integration&f=false

Sommerville, I. 2006. Ingeniería del Software. Séptima Edición. España. Editorial Pearson Educación. Madrid

Universidad Centroccidental Lisandro Alvarado (UCLA). 2002. Manual para la Elaboración del Trabajo Conducente a Grado Académico de Especialización, Maestría y Doctorado. Barquisimeto-Venezuela.

Universidad Pedagógica Experimental Libertador (UPEL). 2010. Manual de Trabajos de Grado de Especialización y Maestría y Tesis Doctorales. Fondo Editorial de la Universidad Pedagógica Experimental Libertador. 4ta edición. Caracas-Venezuela.

Wirdemann, R. y Baustert, T. 2007. Desarrollo REST con Rails. Trabajo distribuido bajo Licencia Creative Commons Reconocimiento-Sin Obra Derivada 2.0. España. Extraído el 12 de Junio del 2012 desde http://www.b-simple.de/download/restful_rails_es.pdf

ANEXOS

A. CURRÍCULO VITAE DEL AUTOR

Desireé Leiling Martínez

Cursante del Postgrado en Ciencias de la Computación Mención Ingeniería de software de la Universidad Centroccidental "Lisandro Alvarado". Nació en San Fernando, Estado Apure, el 10 de Septiembre de 1982. Realizó estudios de Educación Primaria, Secundaria y Diversificada en la Unidad Educativa "Clarisa Este de Trejo", San Fernando de Apure-Venezuela, donde obtiene el título de Bachiller en Ciencias. Posteriormente inicia su carrera universitaria en la Universidad Centroccidental "Lisandro Alvarado" allí obtiene el título de "Ingeniero en Informática" en el año 2006. Entre tanto, obtiene los certificados de: **Operador de Windows 98, Office 97** avalado por el Centro Louisiana Systems School en Abr. 2000, **1er Nivel de Auxiliar contable** avalado por Gerencia Técnica 2000 en Feb. 2001, **7mo nivel aprobado Inglés Básico** avalado por FUNDAUC en Nov. 2.001, **Programador en Visual Fox Pro (básico)** avalado G&T Sistemas en Jun. 2003, **Técnico en Reparación, Mantenimiento y Soporte de Pc's** avalado por Mercadística Data Center en Oct. 2004, **Edición de páginas web, tratamiento de imágenes y animaciones para el web** avalado por Mercadística Data Center en Nov. 2005, **Plataforma de Estaciones de Trabajo P&G** avalado por Centro de Computación D.M.G.R.en Nov. 2.006, **SWEP5** Centro de Computación D.M.G.R. en Nov. 2006, **Diplomado en Docencia Universitaria** avalado por UPEL-IPBen Abr. 2012. Cuadro de Honor UCLA Lapso 2003 – II Nov. 2004 y Por haber obtenido la calificación sobresaliente de 19 puntos en la asignatura ***Contabilidad I*** en la carrera Ingeniería en Informática. Posee experiencia laboral como Administrador del Portal Web Procter Contigo en Sistemas Ventor desde Nov. 2006 a Feb. 2007, como Analista, Programador de Sistemas en Software Libre en CIDE S.A desde Mar. 2007 hasta la actualidad, como Docente instructor tiempo convencional 8 horas en Universidad Centroccidental Lisandro Alvarado desde Jul. 2011 hasta la actualidad.