UNIVERSIDAD CENTROCCIDENTAL
"LISANDRO ALVARADO"

UNA LÍNEA DE PRODUCCIÓN DE SOFTWARE PARA SISTEMAS TRANSACCIONALES. UNA APLICACIÓN AL PROCESO DE DESARROLLO DE SOFTWARE EN LA COORDINACIÓN NACIONAL DE TECNOLOGÍA DE INFORMACIÓN DE LA U.N.E.X.P.O.

LEIBAN ALBERTY RIVERO COLMENÁREZ

Barquisimeto, 2011

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

UNA LÍNEA DE PRODUCCIÓN DE SOFTWARE PARA SISTEMAS TRANSACCIONALES. UNA APLICACIÓN AL PROCESO DE DESARROLLO DE SOFTWARE EN LA COORDINACIÓN NACIONAL DE TECNOLOGÍA DE INFORMACIÓN DE LA U.N.E.X.P.O.

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

Por: LEIBAN ALBERTY RIVERO COLMENÁREZ

Barquisimeto, 2011

UNA LÍNEA DE PRODUCCIÓN DE SOFTWARE PARA SISTEMAS TRANSACCIONALES. UNA APLICACIÓN AL PROCESO DE DESARROLLO DE SOFTWARE EN LA COORDINACIÓN NACIONAL DE TECNOLOGÍA DE INFORMACIÓN DE LA U.N.E.X.P.O.

Por: LEIBAN ALBERTY RIVERO COLMENÁREZ

Trabajo de grado aprobado

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

_____

(Jurado 3)

Barquisimeto, ___ de _____ de 2011

Dedicatoria

A Dios,
mi esposa,
mi hijo,
mis padres y
hermanos

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLA

Tabla **Pág.**

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

UNA LÍNEA DE PRODUCCIÓN DE SOFTWARE PARA SISTEMAS TRANSACCIONALES. UNA APLICACIÓN AL PROCESO DE DESARROLLO DE SOFTWARE EN LA COORDINACIÓN NACIONAL TECNOLOGÍA DE INFORMACIÓN DE LA U.N.E.X.P.O.

Autor: Ing. Leiban Alberty Rivero Colmenárez

Tutor: Dr. Rodolfo Antonio Canelón Osal

RESUMEN

Los sistemas de información emergen como soporte a las necesidades dentro de las organizaciones. Entre los tipos de sistemas de información están los sistemas de procesamiento transaccional, que sirven para el registro de las operaciones diarias y la generación de reportes, presentando información con características de importancia, relevancia, claridad, sencillez y oportunidad de tal forma que sea útil para las personas a quienes se les entrega. Cuando una empresa crece o las circunstancias la llevan a cambiar, los sistemas transaccionales deben incrementar su capacidad y ajustarse, pero estas alteraciones requieren el uso apropiado de metodologías para el desarrollo de software, de modo de no paralizar los procesos de la organización. Las metodologías traen beneficios como costos más bajos, desarrollo del software más rápido con menores riesgos e incremento de la confiabilidad del sistema. Otro inconveniente, se localiza en el desarrollo de software, porque un crecimiento no controlado conduciría al aumento de tiempo en mantenimiento y costo de desarrollo. Un enfoque de líneas de producción de software ayuda en la gestión de los cambios a través de la reutilización de componente de software, reduciendo costos y previendo los tiempos de desarrollo. Por tal motivo, la investigación que se presenta a continuación tiene como propósito, diseñar una línea de producción de software para sistemas transaccionales, para tal fin se expande el proceso de Ingeniería de Dominio basado en Calidad de Software denominado **I**n**D**o**C**a**S** agregándole las actividades correspondiente a la implementación del dominio, instanciando el método WATCH – Component para construir dichas actividades. Finalmente, se aplica el proceso **I**n**D**o**C**a**S** expandido a ciertos sistemas transaccionales que maneja la coordinación nacional de tecnología de la información de la UNEXPO como caso de estudio.

Palabras Claves: Arquitectura de Software, Ingeniería de Dominio, Línea de Producción de Software, calidad de software, Sistemas transaccionales, **I**n**D**o**C**a**S**, WATCH-Component.

INTRODUCCIÓN

Los sistemas de información y las tecnologías de información han cambiado la forma en que operan las organizaciones actuales. A través de su uso se logran importantes mejoras, pues automatizan los procesos operativos, suministran una plataforma de información necesaria para la toma de decisiones y su implantación logra ciertas ventajas competitivas.

Las tecnologías de la información han sido definidas como la integración y convergencia entre la computación, las telecomunicaciones y las técnicas para el procesamiento de datos, donde sus principales componentes son: el factor humano, los contenidos de la información, el equipamiento, la infraestructura, el software y los mecanismos de intercambio de información, los elementos de política y regulaciones, además de los recursos financieros.

La información se ha colocado como uno de los principales recursos que poseen las empresas actualmente. Los entes que se encargan de tomar decisiones comprenden que la información no es sólo un subproducto de la conducción empresarial, sino que a su vez alimenta al negocio y puede ser uno de los factores críticos para la determinación del éxito o fracaso de ésta.

Por tal razón, los sistemas transaccionales son importantes, ya que ayudan a las empresas a manejar sus operaciones cotidianas. Cabe destacar, que estos sistemas operan sobre la médula de cualquier empresa o institución, entre los que se tienen: sistemas de ingreso de órdenes, inventario, fabricación, nómina y contabilidad, entre otros. Debido a su volumen e importancia en la organización, los sistemas transaccionales han sido extendidos, revisados, mejorados y mantenidos al punto que hoy, ellos están completamente integrados a las empresas.

Esta evolución de los sistemas transaccionales trajo consigo algunos inconvenientes en su desarrollo y mantenimiento, como retrasos en la entrega de proyectos, aumentos de los costos de producción del software, fallas en los productos

entregados y poca documentación sobre los mismos. Esto debido, generalmente, a la carencia y uso correcto de métodos y/o estándares adecuados que se requieren para los procesos de desarrollo de software, mínima reutilización de código y de software existente, inexistencia de control sobre cambio de requisitos, dependencia de personas para determinados proyectos, escasas pruebas realizadas al software y la falta de un plan estratégico a seguir.

Por consiguiente, esto ha inducido al área de desarrollo de software a investigar y crear metodologías de desarrollo, orientado al paradigma de la reutilización de componentes de software para intentar satisfacer estas metas. Con ese propósito se han creado a través del tiempo técnicas de reutilización como los módulos, la programación orientada a objeto, la programación orientada a componente, servicios web y por último las líneas de producción de software.

En el proceso general de las líneas de producción de software, se distinguen dos actividades principales, Ingeniería de dominio y Aplicación. En la primera actividad, se establece la plataforma reutilizable y por tanto se define las similitudes y diferencias de la línea de producción. En la segunda actividad, se desarrollan las aplicaciones de la línea de producción derivados de la plataforma establecida en la ingeniería de dominio.

Adicionalmente, las líneas de producción de software suscitan otros beneficios como la disminución de costo de producción, reducción en el tiempo promedio de creación y entrega de nuevos productos, reducción de productos defectuoso y mejoras en el valor del producto que influyen sobre la competitividad de las organizaciones. Entre estos beneficios, se encuentran los relacionados a la calidad de software y según (Krueger, 2006), ésta se pueden medir de dos maneras: la primera como la forma adecuada en que cada producto responde a las necesidades del cliente; La segunda como la tasa de defectos encontrados en cada uno de los productos de la línea, que se pueden mejorar mediante técnicas de software de la línea de producción.

Por otra parte, la calidad de software se define como la totalidad de rasgos y características de un producto de software que le confieren su aptitud para satisfacer necesidades explícitas o implícitas (ISO/IEC, 2001). Para establecer claramente las necesidades explícitas o implícitas, se han definido modelos de calidad, tales como ISO/IEC[1] 25010 (ISO/IEC, 2009), el cual puede ser utilizado para especificar los requisitos de calidad y proporcionar bases para cuantificarlos en términos de medidas específicas. Los modelos de calidad, en general, son estructuras jerárquicas, donde las características de calidad de alto nivel son refinadas en subcaracterísticas, hasta identificar propiedades o atributos medibles (Canelón et al, 2009).

Adicionalmente, para aplicar los procesos generales de la ingeniería de líneas de producción de software se han creados técnicas y métodos, entre los que se mencionan: Attribute-Driven Design (ADD), Siemens' 4 view (S4V), Rational Unified Process (RUP 4+1), Organización y Procesos de Arquitecturas de negocios (BAPO), Modelo de separación de arquitecturas (ASC), Modelo general para el diseño de arquitecturas, proceso para la ingeniería del dominio basado en calidad de software (**I**n**D**o**C**a**S**), entre otros. Se selecciona el proceso **I**n**D**o**C**a**S**, debido a que usa el modelo general para el diseño de arquitecturas, que utiliza los métodos ADD, RUP 4+1, BAPO y ASC, y lo combina con el modelo de calidad ISO/IEC 25010, con el fin de obtener una arquitectura base para una familia de productos, utilizando como disciplinas principales el análisis y diseño del dominio (Canelón, 2010).

De esta manera, la presente investigación aborda el estudio en el área de desarrollo de software a fin de proponer una incorporación al método **I**n**D**o**C**a**S** que permita desarrollar las tres disciplina de la Ingeniería de Dominio que son: análisis, diseño e implementación del dominio, para luego diseñar una línea de producción de software para sistemas transaccionales para la Coordinación Nacional de Tecnología de la Información de la Universidad Nacional Experimental Politécnica "Antonio José de Sucre" en aras de aumentar la calidad de las aplicaciones de software y las respuestas a las necesidades de los usuarios.

El presente trabajo de investigación está estructurado en cinco (5) capítulos:

[1]ISO/IEC: siglas de International Organization for Standardization / International Electrotechnical Commission

En el Capítulo I, se plantea el problema en cuanto a la dificultad de desarrollar software y la falta de calidad de estos en el área de los sistemas transaccionales. Se presentan los objetivos del presente estudio, su justificación e importancia, así como el alcance y limitaciones del mismo.

A continuación, el Capítulo II presenta los antecedentes encontrados en el área, y la revisión teórica en los que se fundamenta la investigación, específicamente en lo que respecta a calidad de software, sistemas transaccionales, enfoque de líneas de producción de software, el proceso Ingeniería del Dominio basado en Calidad de Software (**I**n**D**o**C**a**S**), los métodos WATCH y WATCH- Component.

Seguidamente, el Capítulo III comprende el marco metodológico, está conformado por el tipo de la investigación, la unidad en estudio, la población y muestra considerada, las técnicas de recolección de datos y los instrumentos utilizados.

El Capítulo IV contiene la propuesta del estudio, donde se indica la justificación, los objetivos tanto general como los específicos de la propuesta; posteriormente se explica en detalle la incorporación de las actividades relacionadas a la implementación del dominio en el proceso **I**n**D**o**C**a**S**, utilizando el método WATCH – Component para crear el proceso **I**n**D**o**C**a**S** expandido que cubrirá las disciplinas de análisis, diseño e implementación de la Ingeniería de Dominio. Además, se aplicará el proceso **I**n**D**o**C**a**S** expandido para diseñar una línea de producción de software para sistemas transaccionales en la Coordinación Nacional de Tecnología de Información como caso de estudio.

Finalmente, en el Capítulo V se detallan las conclusiones y recomendaciones derivadas del presente trabajo de investigación.

CAPÍTULO I

EL PROBLEMA

Planteamiento del Problema

Los sistemas de información en general se concentran en el desarrollo, uso y gestión de la infraestructura tecnológica en una organización. Según (Urbina, 2009), “en la era post-industrial, la era de la información, el enfoque de las compañías ha cambiado de la orientación hacia el producto a la orientación hacia el conocimiento”, en este sentido el mercado compite hoy en día en términos de proceso e innovación, en lugar del producto.

Asimismo, un activo preciado dentro de una compañía es su información, representada en su personal, experiencias, conocimientos, innovaciones (patentes, derechos de autor, secreto empresarial). Para poder competir, las organizaciones deben poseer una infraestructura de información estable, en cuyo núcleo se sitúa la infraestructura de la tecnología de información. De tal manera que los sistemas de información buscan la forma de mejorar el uso de la tecnología que soporta el flujo de información dentro de la organización.

Cabe destacar que, el entorno es importante para desarrollar sistemas que cubran las necesidades de las organizaciones, al respecto (Laudon et al, 2006) mencionan que existen dos entornos definidos, un entorno transaccional, donde “el procesamiento de transacciones consiste en captar, manipular, almacenar los datos y también, en la preparación de documentos” y un entorno decisional, “que tiene lugar en la toma de decisiones. Las decisiones se toman a todos los niveles y en todas las

áreas, por lo que todos los sistemas de la organización deben estar preparados para asistir en esta tarea".

Según (Bernstein, 2009), "una transacción es una interacción en el mundo real, por lo general entre una empresa y una persona, donde algo se han intercambiado. Podría tratarse de un intercambio de dinero, productos, información, solicitudes de servicio, y así sucesivamente". Por lo general, los asientos contables deben registrar lo sucedido. A menudo esta contabilidad se realiza por un computador utilizando un sistema de procesamiento de transacciones, para obtener una mejor escalabilidad, confiabilidad y costo. Esto conlleva a que este tipo de sistemas de información, debe registrar fielmente los hechos económicos con la mayor cantidad de información que necesita la empresa, ajustándose a las reglas del negocio. Estas reglas, proporcionan la manera de procesar los datos.

Es importante señalar, que cuando una empresa crece, los sistemas transaccionales deben incrementar su capacidad para ejecutar las operaciones, quizás la organización desee agregarle una funcionalidad y no tirar la implementación actual para empezar de nuevo. Adicionalmente, los sistemas transaccionales están enmarcados por las reglas de negocio, éstas pueden provenir de diversas fuentes: carácter normativo, otras son las políticas de la organización, del ambiente donde se encuentre la empresa y la más importantes, las necesidades del cliente. Cuando algunas de estas fuentes cambia o el negocio crece, por lo general, afectan las características de los sistemas, obligando a los desarrolladores a modificarlos para que puedan adaptarse a las nuevas necesidades del entorno.

Estas modificaciones pueden ocasionar problemas, puesto que una modificación mal realizada puede paralizar los procesos operativos críticos, y por consecuencia traer perdidas financieras a las organizaciones. Es por eso, que estas adaptaciones en el desarrollo de un sistema deben estar bien orientadas, porque un cambio no orientado, no solo afecta el área donde se está modificando, sino también reforma las áreas que estén relacionadas. Según (Dorman y Boyd, 1997), "una alteración realizada a un objeto principal puede causar un efecto dominó a todos los objetos de

menor jerarquía, es decir, provocarían una cadena de modificaciones a numerosos componentes vinculados y sería difícil predecir los efectos de los cambios ejecutados". Por otro lado, el uso de un componente común de forma incorrecta puede tener consecuencias impredecibles. Un error al cambiar el código en cualquier nivel, puede tener consecuencia en lugares inesperados, inclusive en los que no tienen relación alguna. Este efecto dominó es más destructivo que los cambios realizados cerca del área.

Este es uno de los motivos, por el cual los procesos de desarrollo de software han buscado estrategias para la reutilización. Según (Sommerville, 2005), "esta reutilización es posible a diferentes niveles (desde funciones simples a aplicaciones completas)" y (Montilva et al, 2003) señalan que la reutilización de software "es un proceso de la Ingeniería de Software que conlleva al uso recurrente de activos de software en la especificación, análisis, diseño, implementación y pruebas de una aplicación o sistema de software".

Continuando en el área de la reutilización, (Sommerville, 2005), señala que "existen actualmente diferentes técnicas y métodos como son: los patrones de diseño, desarrollo basado en componente, marcos de aplicaciones (*Framework*), construcción de capas con interfaces a sistemas heredados, sistemas orientados a servicios, líneas de productos de software, integración COTS *(Component of Commercially available Off-The-Shelf)*, aplicaciones verticales configurables, librerías de programas, generadores de programas, desarrollo orientado a aspectos, entre otros". Entre estos métodos, una de las aproximaciones más efectivas para la reutilización es la creación de líneas de productos de software o familias de aplicaciones. Cada aplicación especializada proviene de una arquitectura común.

Estas técnicas y métodos de reutilización traen beneficios como costos más bajos, desarrollo del software más rápido y menores riesgos, incremento de la confiabilidad del sistema y los especialistas pueden ser utilizados de forma más efectiva concentrando su experiencia en el diseño de componentes reutilizables (Sommerville, 2005).

No obstante, el mismo (Sommerville, 2005) sostiene que hay ciertos problemas asociados a la reutilización, como los siguientes:

- Incremento en los costos de mantenimiento. Esto es debe a que si el código fuente de un software o componente reutilizado no está disponible, entonces los costos de mantenimientos pueden incrementarse ya que los elementos reutilizados pueden ser menos compatibles con los cambios del sistema.

- Falta de soporte de las herramientas. Los conjuntos de herramientas CASE (*Computer Aided Software Engineering*), ingeniería de software asistida por computadora, no soportan el desarrollo con reutilización. Puede ser difícil o imposible integrar estas herramientas con un sistema de librería de componentes.

- Síndrome "reinventar la rueda" (reescribir componentes).

- Creación y mantenimiento de una librería de componentes. Puede ser costoso construir una librería de componentes reutilizable y asegurar que los desarrolladores de software puedan usarla. Las técnicas actuales para clasificar, catalogar y recuperar componentes de software todavía están inmaduras.

- Tiempo en búsquedas, comprensión y adaptación de componentes reutilizables. Los componentes de software tienen que buscarse en una librería, entenderse y algunas veces, adaptarse al trabajo en un entorno nuevo.

Un ejemplo en particular, son los costos asociados al estudio de un componente para comprobar si es apropiado para su reutilización en una situación concreta y asegurar su confiabilidad. Estos costos adicionales pueden impedir la introducción de la reutilización y puede implicar que las reducciones de los costos totales de desarrollo mediante reutilización sean menores que las de los costos anticipados. En tal sentido, la reutilización de software debe ser bien planeada e implantada en la organización a través de un proyecto de reutilización.

En relación al desarrollo de proyectos de Tecnologías de Información a nivel mundial, (Goldsmith, 2007) publica que "sólo el 34% de los proyectos se completaron en el tiempo y costo estimado, esto debido a que muchos de los

proyectos están destinados al fracaso porque la administración ha dictado arbitrariamente un presupuesto y unas actividades que no guardan relación con el trabajo a realizar y un mal levantamiento de los requisitos solicitados por el cliente".

En Venezuela, un estudio realizado por (Rivero et al, 2007), muestra que "el 51% dice emplear un modelo propio, es decir, han desarrollado su propio modelo de procesos o utilizan una mezcla de varios de los modelos conocidos". Sin embargo, se pudo detectar que muchas empresas que usan un modelo propio, por lo general, no tienen el modelo documentado a un nivel de detalle que facilite su uso corporativo.

El resultado mostrado de este estudio, hace ver la carencia de información en la utilización de técnicas y/o estándares tanto en la documentación como en la desarrollo de los sistemas, propiciando además, que haya dependencia de personas para determinados proyectos que tengan conocimientos de los métodos.

Siguiendo con el mismo estudio, "el 68% de la muestra señaló que no es obligatorio el uso de modelos de procesos en los proyectos", es decir, que el desarrollo de los sistemas depende de cómo el desarrollador interpreta el método usado en su empresa. En este aspecto, los autores concluyen que la mayoría de las empresas INS (Industria Nacional de Software) no están empleando modelos y métodos maduros en su proceso productivo, lo cual limita significativamente la calidad de los productos que estas empresas generan.

Un ejemplo de lo mencionado anteriormente, se encuentra en la Coordinación Nacional de Tecnología de la Información de la UNEXPO[2], donde se usa una mezcla de varios métodos y/o técnicas para el desarrollo de los proyectos de tecnología de información, pero la documentación existente carece de detalles y no es obligatorio el uso de los métodos y/o técnicas en el proceso de desarrollo de software.

Por consiguiente, se muestra como la mayoría de los proyectos de desarrollo ejecutados se exceden en costo y en los plazos acordados, los productos resultantes tienen numerosos defectos, presentan carencias en la calidad de las aplicaciones y poca documentación de las mismas. Asimismo, estos efectos producen un alto

[2] UNEXPO: Universidad Nacional Experimental Politécnica "Antonio José de Sucre"

consumo de tiempo para el mantenimiento de sistemas y respuestas lentas a las nuevas funcionalidades y módulos solicitados por los usuarios, traduciéndose en desconfianza de los clientes en el desarrollo de los sistemas, especialmente cuando se realizan cambios de requisitos a los proyectos de desarrollo.

Por tal motivo, se hace importante el estudio de un enfoque que proporcione a una organización sostener el crecimiento de sus sistemas manteniendo y mejorando la calidad de los mismos y aumentar la previsión en cuanto al costo y tiempo de ejecución de posibles cambios. Adicionalmente, obtener un mayor control de configuraciones de los diversos productos a través de la reutilización de activos de software y de código.

Por lo antes expuesto, es necesario el estudio para diseñar una plataforma de sistemas transaccionales con un enfoque de líneas de producción software, puesto que permite a una organización determinar y reutilizar activos de software, así como también mejorar la calidad de los mismos.

Se hace énfasis en el uso de las líneas de producción de software, porque según (Montilva, 2006), trae mejoras en el tiempo de entrega del producto (time *to market*), y reducciones de los costos de ingeniería, las tasas de defectos y el tiempo promedio de creación y entrega de nuevos productos. Además, se obtiene un incremento en cuanto al portafolio de productos y el número total de productos que pueden ser efectivamente desplegados y mantenidos.

De igual manera, por ser un enfoque de líneas de producción de software es necesario un proceso de desarrollo del software, por lo cual se han escrito métodos y técnicas con el fin de crear la línea de producción de software. Entre los métodos más representativos de la industria se muestran los siguientes: Attribute-Driven Design (ADD), Siemens' 4 view (S4V), Rational Unified Process (RUP 4+1), Organización y Procesos de Arquitecturas de negocios (BAPO), Modelo de separación de arquitecturas (ASC), Modelo general para el diseño de arquitecturas y proceso para la ingeniería del dominio basado en calidad de software (**I**n**D**o**C**a**S**).

De la técnicas antes mencionada, se selecciona como base el proceso **I**n**D**o**C**a**S**, debido a que usa el modelo general para el diseño de arquitecturas, que utiliza los métodos ADD, RUP 4+1, BAPO y ASC, y lo combina con el modelo de calidad ISO/IEC 25010, característica que no se ve, en forma explícita en otros modelos de procesos existentes, con el fin de obtener una arquitectura base para una familia de productos, utilizando como disciplinas principales el análisis y diseño del dominio (Canelón, 2010). No obstante, el proceso solo ejecuta dos de las tres disciplinas de la ingeniería del dominio, lo cual conlleva, a buscar métodos y/o técnicas para construir las actividades necesarias para completar la disciplina de la implementación del dominio.

En atención a lo anteriormente expuesto, se propone un diseño de línea de producción de software para sistemas transaccionales utilizando como base el proceso **I**n**D**o**C**a**S** y agregándole las actividades necesarias para realizar la implementación del dominio, para que abarque las disciplinas el análisis, diseño e implementación de la Ingeniería de Dominio, con el fin de construir los activos de software que formará parte de la línea de producción de software, así como también mejorar la calidad de los mismos.

Sobre la base de lo antes planteado, surgen las siguientes interrogantes que contribuirán al desarrollo de esta investigación:

- ¿Cuáles son las características esenciales de los sistemas transaccionales?
- ¿De qué manera las líneas de producción de software benefician el desarrollo de los sistemas transaccionales?
- ¿De qué forma completar el proceso **I**n**D**o**C**a**S** para desarrollar el enfoque de las líneas de producción de software en sistemas transaccionales?
- ¿Es posible llevar a cabo el diseño de una línea de producción de software para sistemas transaccionales a la Coordinación Nacional de Tecnología de Información de la UNEXPO?

- ¿Cómo proponer una línea de producción de software para sistemas transaccionales a la Coordinación Nacional de Tecnología de Información de la UNEXPO?
- ¿Qué activos de software definen la base para la línea de producción de software de sistemas transaccionales?

Finalmente, es propio señalar que dichas interrogantes alcanzarán sus respuestas a través de la consecución de los objetivos específicos expuestos en el presente estudio.

Objetivos

Objetivo General

Proponer una línea de producción de software para sistemas transaccionales con una aplicación al proceso de desarrollo de software en la Coordinación Nacional de Tecnología de Información de la Universidad Nacional Experimental Politécnica “Antonio José de Sucre” (UNEXPO), utilizando el proceso expandido para la Ingeniería del Dominio basado en Calidad de Software (**I**n**D**o**C**a**S**).

Objetivos Específicos

1. Caracterizar los sistemas transaccionales.
2. Estudiar el enfoque de líneas de producción de software.
3. Añadir las actividades correspondientes a la implementación del dominio al proceso para la Ingeniería del Dominio basado en Calidad de Software (**I**n**D**o**C**a**S**).
4. Construir la arquitectura de software para los sistemas transaccionales.
5. Analizar la factibilidad de la propuesta de la línea de producción de software para sistemas transaccionales a la coordinación nacional de tecnología de información de la Universidad Nacional Experimental Politécnica “Antonio José de Sucre”.
6. Aplicar el proceso expandido de **I**n**D**o**C**a**S** a la línea de producción de software para sistemas transaccionales a la coordinación nacional de tecnología de información de la Universidad Nacional Experimental Politécnica “Antonio José de Sucre”.

Justificación e Importancia

En primer lugar, es necesario señalar que el software es un activo intangible, es decir, su crecimiento no depende directamente de la inversión de dinero, sino de la forma en que se desarrolla el mismo. Por eso su evaluación no es directa, es necesario entonces estudiar el proceso del desarrollo, la manera que se realiza y los efectos que tenga en las organizaciones.

Del mismo modo, el desarrollo del software se ha orientado, según (Pohl et al, 2005), a dos tipos de productos, aquellos producidos individualmente y los producidos en masa, es decir, las aplicaciones estándares identificadas en un dominio. En general, cada uno de estos tipos de productos tiene sus inconvenientes. Los productos de software individuales son bastante caros, mientras que los productos de software estándar carecen de suficientes variaciones, intentando cubrir la creciente demanda de productos individualizados, es decir, cubrir las necesidades específicas de los clientes.

Asimismo, existe cierta complejidad al desarrollo y mantenimiento del software, elevando los costos de producción, retardando los tiempos de entrega y presentando fallas en el producto. Estos afectan el negocio y de manera importante si dichos sistemas están relacionados con la parte operativa de la organización (compra, venta, inventario, entre otros).

Por tal razón, se justifica la propuesta de una línea de producción de software para sistemas transaccionales, puesto que sus beneficios, tales como la disminución de costo de producción, reducción en el tiempo promedio de creación, entrega de nuevos productos y de productos defectuosos, así como mejoras en el valor del producto influyen sobre la competitividad de las organizaciones. Adicionalmente, una línea de producción de software ayuda a mejorar la gestión del desarrollo y la calidad del software que se realiza.

Estos beneficios también contribuirán con el trabajo a los usuarios, porque una aplicación que no presente falla en su elaboración y sea entregada a tiempo, mejorará

significativamente los procesos de la organización debido a que las actualizaciones de sus productos de software se harán en el tiempo previsto, ayudándole a conseguir los objetivos planteados y mejoras en los servicios prestados.

Para el diseño de la línea de producción de software, se utilizará el proceso para la Ingeniería del Dominio basado en Calidad de Software (**I**n**D**o**C**a**S**), debido a que, asocia el modelo de calidad ISO/IEC 25010 al dominio, propiedad no encontrada de forma clara en métodos y técnicas existentes. No obstante, el método **I**n**D**o**C**a**S**, solo alcanza las disciplinas del análisis y diseño de la Ingeniería del dominio, en consecuencia se hace necesario, el desarrollo de las actividades y artefactos correspondiente a la disciplina de la implementación del dominio, para abarcar las tres disciplinas (Análisis, diseño e Implementación del Dominio) de la Ingeniería del Dominio.

Por consiguiente, se espera que el resultado de la investigación sea de importancia en el área de los sistemas transaccionales y líneas de producción de software en general, siendo referencia para impulsar nuevos proyectos en la comunidad científica. Asimismo, en el área industrial, al agregar ventajas competitivas a las organizaciones cuyas operaciones básicas estés soportadas por aplicaciones de software, que funcionan correctamente y se adapten a los crecientes cambios del entorno.

Con relación al aporte a la industria de desarrollo de software, la propuesta de línea de producción de software para Sistemas Transaccionales puede ser incorporada en las empresas como parte de sus activos de software en sus ambientes de desarrollo de aplicaciones, puesto que el uso de modelos de desarrollo de software, permite ahorrar tiempos de desarrollo lo que se traduce en disminución en los tiempos de entrega y costos asociados al producto.

Por otra parte, el proceso **I**n**D**o**C**a**S,** al cual se le agregará la disciplina de implementación del dominio de la Ingeniería de Dominio, podría ser usado como referencia para la construcción de líneas de producción de software en otros dominios distinto a los sistemas transaccionales tanto en el área académica como en el área industrial.

Para finalizar los usuarios de sistemas transaccionales podrían experimentar la reducción de tiempo en las versiones y actualizaciones de sus aplicaciones o servicios y un aumento en la calidad de los mismos.

Alcance y Limitaciones

El alcance de la presente investigación radica en las tres disciplinas (Análisis, diseño e Implementación del Dominio) de la Ingeniería del Dominio, para esto se utiliza como base el proceso para la Ingeniería del Dominio basado en Calidad de Software (**I**n**D**o**C**a**S**), que cubre las disciplinas del análisis y diseño de la Ingeniería del dominio, agregándole las actividades y artefactos correspondiente a la disciplina de la implementación del dominio, para abarcar las disciplinas de la Ingeniería del Dominio. Con está expansión del proceso **I**n**D**o**C**a**S**, se realiza el diseño de una línea de producción de software para sistemas transaccionales para la coordinación nacional de tecnología de información de la UNEXPO.

Para comenzar, en la actividad de ingeniería del dominio, se caracteriza el dominio de los sistemas transaccionales hasta conseguir los requisitos funcionales y no funcionales, añadiéndole las propiedades de calidad correspondiente obteniendo el modelo de calidad ISO 25010. Seguidamente en el diseño del dominio se investigan y determinan los estilos arquitecturales que contribuyen a la solución, se evalúan las arquitecturas candidatas para comprobar que cumplen con los requisitos y finalmente se desarrollan los componentes y administran los repositorios de activos de software. El producto de esta actividad, se resume en componentes de software, requisitos reutilizables y configurables, modelos de análisis y diseño de software.

Para finalizar se aplicará la línea de producción de software resultante en un caso de estudio aplicado a la Coordinación Nacional de Tecnologías de Información de la UNEXPO.

En relación a las limitaciones de la investigación, es propio señalar que se enfoca en los sistemas transaccionales, los cuales, deben cumplir ciertas características como: atomicidad, consistencia, aislamiento y durabilidad, además, deben ser capaces de responder rápidamente, poseer mecanismos de recuperación y respaldo de datos en caso de fallas y ser inflexibles, es decir, no pueden aceptar información distinta a la establecida. En cuanto al enfoque de líneas de producción de software, se plantearán las actividades relacionadas a la ingeniería de dominio y se obtendrán los artefactos correspondientes a dicha disciplina.

Las características de los sistemas transaccionales son importantes para el desarrollo y mantenimiento de estas aplicaciones. Si un desarrollo o modificación a este tipo de aplicaciones, fuese realizado sin cumplir estas reglas, podría afectar a los procesos del negocio, y por lo consiguiente, el funcionamiento de la organización.

Adicionalmente, se incluirán las actividades y artefactos correspondientes a la disciplina de la implementación del dominio para incorporarlo al método **I**n**D**o**C**a**S**, ya que el método solo alcanzan las disciplinas del análisis y diseño de la Ingeniería del dominio, con el objeto de abarcar las tres disciplinas (Análisis, diseño e Implementación del Dominio) de la Ingeniería del Dominio.

Por otro lado, en lo que respecta a las líneas de producción de software se observa escasez en los métodos que guían de manera detallada la construcción y obtención de los componentes necesarios, de la misma manera como sucede en el campo de la calidad de software, donde la obtención del modelo de calidad se hace de manera subjetiva, apelando en muchos casos a la pericia del diseñador.

CAPÍTULO II

MARCO TEÓRICO

Antecedentes

Los sistemas de información han evolucionado en el tiempo, ya que en los inicios su función principal era la de recopilar información para ayudar a la toma de decisiones, y ahora su finalidad es dar soporte a los procesos básicos de la organización. Como se observa en la figura 1, los primero sistemas que empezaron a desarrollarse fueron precisamente, los sistemas transaccionales.

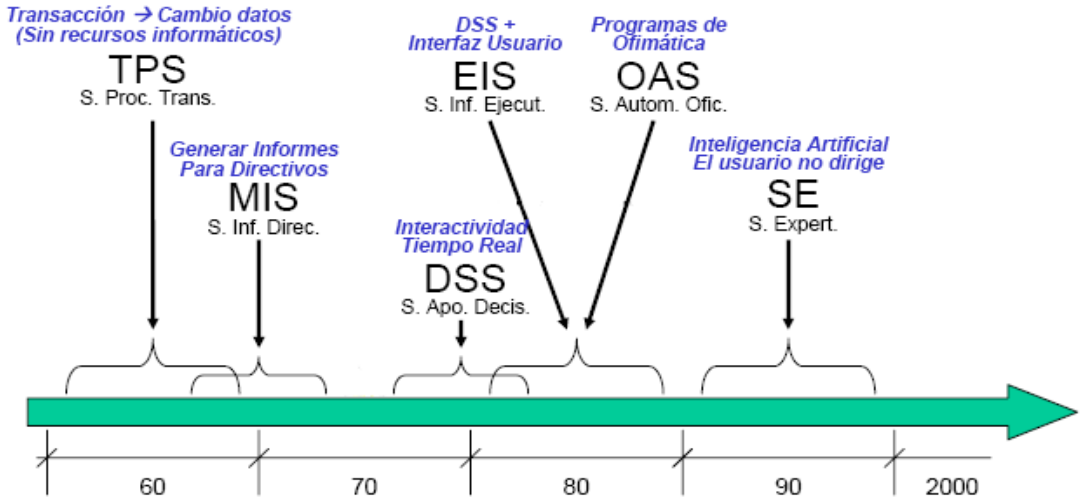


Figura 1. Evolución de los sistemas de información (Laudon, 2006).

Esta evolución de los sistemas, ha inducido al área de desarrollo de software a investigar y crear metodologías de desarrollo de software, con la finalidad de reducir el tiempo de creación y entrega de nuevos productos, los defectos por productos,

esfuerzo requerido para desarrollar y mantener las aplicaciones y el costo de producción. Para ello se ha intentando usar el paradigma de la reutilización de componentes de software para intentar satisfacer estas metas. Con ese propósito se han creado a través del tiempo, técnicas de reutilización como los módulos, la programación orientada a objeto, la programación orientada a componente, servicios web y recientemente, la línea de producción de software (Ver figura 2).

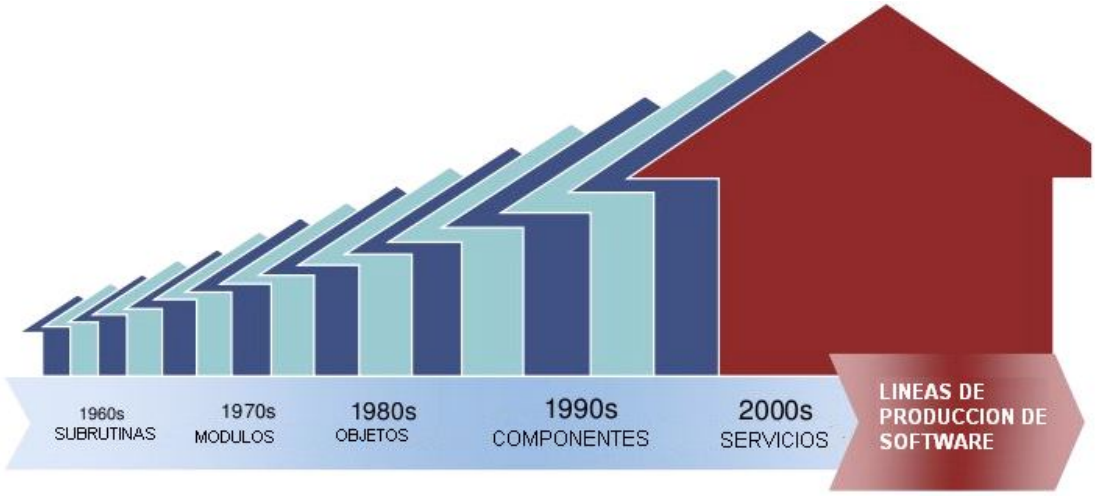


Figura 2. Evolución de la reutilización (Northrop, 2008)

Para la elaboración de este proyecto de investigación se hizo necesaria la revisión bibliográfica y de trabajos de grado en cuanto desarrollo de sistemas transaccionales y líneas de producción de software con el fin de formar un grupo de antecedentes válidos relacionados con las variables en estudio.

A continuación, se presentan algunos trabajos realizados al área de metodología de desarrollo de sistemas transaccionales.

Antecedentes de Línea de Producción de Software para Sistemas Transaccionales

En el trabajo de Morisawa y Torii(2001)[a], en cuya investigación denominada "A Practical Method to Select an Architectural Style of Product Lines for Distributed Processing Systems" (Un método práctico para seleccionar un estilo arquitectural de Líneas de Productos para Sistemas de Procesamiento Distribuido), proponen un método sencillo pero práctico para seleccionar un estilo apropiado de arquitectura para el desarrollo de sistemas, basado en una matriz dividida en dos grupos la ubicación de los datos, si es centralizado o distribuido y tipo de procesamiento si es asíncronos o síncrono (Ver figura 3).

Tipo de procesamiento entre C/S	Tipo de dato (*1) / (*2)	Centralizado	Distribuido	
			Procesamiento Síncrono	Procesamiento Asíncrono
Procesamiento Síncrono	Tipo de transacción	Estilo Transacción Centralizada	Estilo Transacción Distribuida	Estilo Transacción Asíncrona
	Tipo de Consulta	Estilo de Consulta Centralizada	Estilo de Consulta Distribuida	Estilo de Consulta Asíncrona
Procesamiento Asíncrono	Tipo de Notificación	Estilo Notificación Centralizada	Estilo Notificación Distribuida	Estilo Notificación Asíncrona

(*1) Tipo de procesamiento entre servidores (*2) Tipo de Mensaje

Figura 3. Estilos Arquitecturales (Morisawa y Torii, 2001)[a]

Seguidamente los mismo autores, Morisawa y Torii(2001)[b], proponen "An Architectural Style of Product Lines for Distributed Processing Systems, and Practical Selection Method" (Un estilo arquitectónico de líneas de productos de Distributed Sistemas de procesamiento, y un Método práctico de Selección), con el objetivo de simplificar el proceso de toma de estas decisiones para seleccionar la arquitectura de un sistema de aplicación en un entorno de computación distribuida, desarrollando un estilo de arquitectural para sistemas distribuidos de procesamiento..

Estos autores, con ambos estudios apoyan este trabajo de investigación puesto que clasifican las líneas de productos para sistemas distribuidos en nueve categorías de procesamiento basados en la ubicación de almacenamiento de datos y el estilo de

procesamiento entre el cliente y el servidor, aspecto que comprende a su vez a los sistemas transaccionales. Hacen un aporte importante respecto a la arquitectura. Sin embargo, carece de aspectos de calidad en los productos derivados de la línea y de claridad en cuanto a las actividades centrales del proceso de línea de producción de software.

En la publicación World Wide Web del año 2004, Papazoglou (2003) propone en su artículo "Web Services and Business Transactions", un modelo de transacciones de varios niveles que proporciona la independencia necesaria para ordenar los recursos que participan, por ejemplo, bases de datos locales y servidores, de las organizaciones que realizan transacciones de negocios que se componen de la interacción de servicios web. También se presenta una taxonomía de los negocios electrónicos de transacción con características tales como los criterios de atomicidad no convencionales, la necesidad de apoyar las intercambio de negocios y la necesidad de distinguir entre tres fases básicas de transacciones comerciales.

En este caso, el autor hace un aporte importante en cuanto al estado del arte de las transacciones, los sistemas transaccionales y los elementos involucrados. Este modelo de varios niveles es de gran utilidad en la investigación, ya que aporta una idea para elicitar los requisitos de los sistemas transaccionales. Sin embargo, no se toman en cuenta los atributos de calidad, ni las ventajas que ofrecen los modelos de calidad para evaluar los sistemas.

En la tesis de Montaldo (2005), se propone en su trabajo "Patrones de Diseño de Arquitecturas de Software Enterprise[3]", un sistema reusable (framework), realizado bajo la plataforma Java, como solución al problema de construcción de sistemas con una arquitectura de software de tipo Empresarial, entendiendo por Empresarial, los sistemas cliente /servidor de tres o más capas como por ejemplo Comercio electrónicos, Sistemas Transaccionales, ERP[4] entre otras aplicaciones.

La investigación anterior es de importancia para este proyecto, puesto que muestra un diseño de arquitectura para sistemas tipo Empresarial, así como también los

[3] Enterprise: Traducido al español es Empresarial.
[4] ERP: siglas de Enterprise resource planning system. Sistemas de planificación de recursos empresariales.

patrones de diseño a usar en esta arquitectura, la cual, puede ser de utilidad a la hora del diseño de la línea de producción de software para sistemas transaccionales.

Antecedentes de Proceso de Desarrollo de Software para Líneas de Producción de Software

Montilva et al., (2000), crearon un marco metodológico llamado "modelo WATCH", debido a la necesidad de un modelo de procesos para el desarrollo de proyectos de software en pequeños y medianas empresas de desarrollo. Este método, en su última versión Montilva et al, (2008), es un marco metodológico que describe los procesos técnicos, gerenciales y de soporte que deben emplear los equipos de trabajo que tendrán a su cargo el desarrollo de aplicaciones de software empresarial. Adicionalmente, este antecedente nos da tres modelos a seguir, los cuales son: modelo de productos, modelo de actores y modelo de procesos.

El trabajo anterior se toma como antecedente ya que aporta una amplia información de cómo integrar los procesos de gestión de proyectos y los procesos técnicos relacionados a las actividades de análisis, diseño e implementación de una aplicación. Sin embargo, la investigación está orientada a las actividades de la ingeniería de aplicación, es decir, al desarrollo del software y esta limitado a un proyecto en particular.

Esta investigación sirve de apoyo al presente proyecto ya que presenta dentro del proceso diseño detallado, el subproceso Diseño de componentes en donde se describe las actividades y técnicas utilizadas para realizar la especificación detallada de cada componente de la aplicación. Adicionalmente, nos muestra técnicas y/o métodos para realizar planes de pruebas y verificación en las diferentes etapas del desarrollo de software (Análisis, Diseño e Implementación).

Seguidamente, Hammar y Montilva (2003), realizaron una variación del método WATCH que llamaron WATCH – Component, en la tesis denominada "Aspectos Metodológicos del desarrollo y reutilización de componentes de software", en donde

proponen una serie de actividades relacionadas al desarrollo de un componente de software reutilizable. El aporte de esta investigación, es que proporciona cinco actividades asociadas a los procesos gerenciales, la cual nos permiten crear, desarrollar, seleccionar y/o mantener un componente reutilizable. No obstante, este método esta enfocado en el desarrollo y puesta a punto de un componente determinado.

Para esta investigación es muy importante este antecedente puesto que las cinco actividades proporcionada por el método WATCH – Component puede ayudarnos al diseño detallado de componentes e Interfaces reusable. Además, nos muestra como construir y realizar pruebas al componente desarrollado. Por otra parte, este método puede ser instanciado y usados con otros métodos y/o técnicas, lo cual podría ser usado en desarrollo de procesos para línea de producción de software.

Más adelante, Durán et al (2008), en un artículo llamado "Un marco de trabajo de una fábrica de software para el reuso del diseño arquitectónico y de componentes de software", presenta un marco de trabajo de una fábrica de software. También, ilustra la propuesta a través de un caso de estudio para los modelos de componentes de EJB[5] y COM[6].

La importancia de este estudio como aporte para la investigación, se basa en el enfoque de incrementar el nivel de reutilización en dos dimensiones: diseño arquitectónico y componentes de software, para eso utilizan tres vistas Componentes y Conectores (C&C) llamadas: C&C conceptual, C&C ingenieril y C&C de software. En estas vistas, nos describe la construcción, diseño e implementación de componentes de software para reuso, partiendo de una arquitectura de software predefinida.

Luego, Canelón (2010), en su tesis doctoral propone "Un proceso para la Ingeniería del Dominio basado en Calidad de Software", donde presenta un proceso llamado **I**n**D**o**C**a**S** para la ingeniería de dominio basado en un método general de diseño arquitectónico y se fundamenta en el modelo de calidad de software ISO/IEC 25010. El objetivo fundamental de **I**n**D**o**C**a**S** es obtener una arquitectura base para

[5] EJB: siglas de Enterprise JavaBeans.
[6] COM: siglas de Component Object Model.

una familia de productos, aplicando dos de las tres disciplinas de la ingeniería de dominio que son el análisis y diseño del dominio. También, ilustra la propuesta a través de un caso de estudio para las aplicaciones de aprendizaje móvil sensible al contexto.

La transcendencia de la investigación anterior, es que permite crear arquitecturas bases para familias de productos cuidando la calidad desde el principio del desarrollo de la línea de producción de software, es decir, desde la identificación de los requisitos hasta la selección de la arquitectura. El déficit de la investigación es que no presenta las actividades relacionadas a la implementación del dominio.

Este antecedente es muy importante para esta investigación, ya que los productos de software derivados del análisis del dominio y diseño del dominio son fácilmente reutilizables y ajustables a otros métodos. Adicionalmente, las actividades del proceso **I**n**D**o**C**a**S** están relacionados al modelo de calidad ISO/IEC 25010, permitiendo obtener la arquitectura base de la línea de productos con calidad de software.

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

Tabla 1: Resumen de los antecedentes

Área de Estudio	Antecedente	Objetivos	Aporte	Limitaciones
Líneas de Producción de Software para sistemas Transaccionales	Morisawa y Torii(2001)[a], *A Practical Method to Select an Architectural Style of Product Lines for Distributed Processing Systems.* The 2001 International Conference on	Proporcionar una clasificación de los estilos arquitecturales para sistemas de procesamiento distribuido, con el fin proveer herramientas probadas de software, para la construcción de sistemas estables	propone un método sencillo y práctico para seleccionar un estilo arquitectónico adecuado para el desarrollo de un sistema de aplicación	Carece de aspectos de calidad claros en los productos derivados de la línea y de claridad en cuanto a las actividades centrales del proceso de línea de producción de software.

	Parallel and Distributed Processing Techniques and Applications (PDPTA'2001), Las Vegas.	con un menor costo en menos tiempo		
	Morisawa Y., Torii K. (2001)[b]. *Architectural Styles for Distributed Processing Systems and Practical Selection Method*. Proceedings of the 8th European software engineering conference held jointly with 9th ACM SIGSOFT international symposium on Foundations of software engineering. Viena. Austria.	Simplificar el proceso de toma de estas decisiones para seleccionar la arquitectura de un sistema de aplicación en un entorno de computación distribuida, desarrollando un estilo de arquitectural para sistemas distribuidos de procesamiento	Propone un método para seleccionar un estilo arquitectónico adecuado para el desarrollo de un sistema de aplicación. Se diferencia al anterior, a que detallan mejor las actividades a realizar.	Carece de aspectos de calidad claros en los productos derivados de la línea y de claridad en cuanto a las actividades centrales del proceso de línea de producción de software.
	Papazoglou M. (2003). *Web Services and Business Transactions*. Springer Netherlands. Volume 6, Number 1. marzo de 2003.	Presentar un marco de transacciones de negocio basadas en redes de servicios web, colaboración y exponer sus necesidades, características esenciales y módulos.	Aporta información en cuanto a las transacciones web, los sistemas transaccionales web y los elementos involucrados.	No se toman en cuenta los atributos de calidad, ni las ventajas que ofrecen los modelos de calidad para evaluar los sistemas. Solo toma como referencia las transacciones web.
	Montaldo, F. (2005). *Patrones de Diseño de Arquitecturas de Software Enterprise*.	Presentar un marco de trabajo (Framework), bajo plataforma java, para la construcción de sistema con una	Diseño de la arquitectura y patrones de diseño para sistemas tipo Enterprise (Comercio	No esta orientado a un enfoque de línea de producción de software. No se toman en cuenta los

	Tesis de Grado. Departamento de Computación, Facultad de Ingeniería, Universidad de Buenos Aires. Argentina.	arquitectura de software de tipo Enterprise	Electronico, ERP, Sistemas Transaccionales)	atributos de calidad y modelo de calidad.
Proceso de Desarrollo de Software para Líneas de Producción de Software	Montilva J., Hazam K. y Gharawi J. (2000). *The Watch Model for development business software in small and midsize organization.* In IV World Multiconference on Systemics, Cybernetics and Informatics (SCI`2000). Orlando, Florida (USA). Actualizado en el 2008 por Montilva J., Barrios J. y Rivero M.	Presentar marco metodológico que describe los procesos técnicos, gerenciales y de soporte que deben emplear los equipos de trabajo que tendrán a su cargo el desarrollo de aplicaciones de software empresarial.	Describe las actividades y técnicas utilizadas para realizar la especificación detallada de cada componente de la aplicación. También, nos muestra técnicas y/o métodos para realizar planes de pruebas y verificación en las diferentes etapas del desarrollo de software (Análisis, Diseño e Implementación)	No se muestra claramente los atributos de calidad, y modelos de calidad a usar. Esta orientado a la ingeniería de la aplicación
	Hamar V. y Montilva J. (2003). *Aspectos Metodológicos del Desarrollo y reutilización de Componentes de Software*. Tesis de la Universidad de Los Andes. Mérida.	Producir componentes de software reutilizables, centrado en el desarrollo individual del componente.	Describe las actividades y técnicas para el diseño detallado de un componente e Interfaz reusable. Además, nos muestra como construir y realizar pruebas al componente desarrollado.	No se muestra claramente los atributos de calidad, y modelos de calidad a usar, para los levantamientos de requisitos.
	Durán H., Meda M., Sapien A., Piñón L. (2008). *Un marco de trabajo de una fábrica de*	Presentar un marco de trabajo de una fábrica de software para la construcción de componente de	El marco de trabajo esta compuesto por tres vistas Componentes y Conectores	No se muestra claramente los atributos de calidad, y modelos de calidad a usar,

	software para el reuso del diseño arquitectónico y de componentes de software. Tecnociencias. Volumen II Nro 1 Enero-Abril 2008. Chihuahua. México.	software reutilizables.	(C&C) llamadas: C&C conceptual, C&C ingenieril y C&C de software. En estas vistas, nos describe la construcción, diseño e implementación de componentes de software para la reutilización, partiendo de una arquitectura de software predefinida.	para los levantamientos de requisitos. No presenta actividades para la prueba de componentes.
	Canelón R (2010) *Un proceso para la ingeniería del dominio basado en calidad de software. Una aplicación al dominio del aprendizaje móvil sensible al contexto*. Tesis Doctoral. UCV	Obtener una arquitectura base para una familia de productos a través de un proceso para la ingeniería de dominio cuyas disciplinas principales son el análisis y el diseño del dominio.	Aporta un proceso para la ingeniería del dominio basado en calidad de software, siguiendo un enfoque de líneas de producto de software, abarcando las disciplinas de Análisis y Diseño del Dominio.	No esta desarrollado la disciplina de Implementación del dominio del proceso de Ingeniería de Dominio.

Fuente: Autor de la investigación

Bases Teóricas

A continuación se presentan los resultados de la revisión bibliográfica relacionados con el tópico de investigación propuesto y con los objetivos del estudio.

Ingeniería de Línea de Producción de Software

La ingeniería de la línea producción de software ha resultado ser la metodología usada para el desarrollo de una diversidad de productos de software a costos más bajos, en menor tiempo y con mayor calidad. Numerosos informes mencionan los logros importantes y la experiencia adquirida mediante la introducción de líneas de producción de software en la industria del software (Pohl et al, 2005). Según este mismo autor, "es un paradigma para el desarrollo de aplicaciones de software, utilizando plataformas y más personalización".

Según (Pohl et al, 2005), el paradigma de ingeniería línea de producción de software se separa dos procesos generales (Ver figura 4):

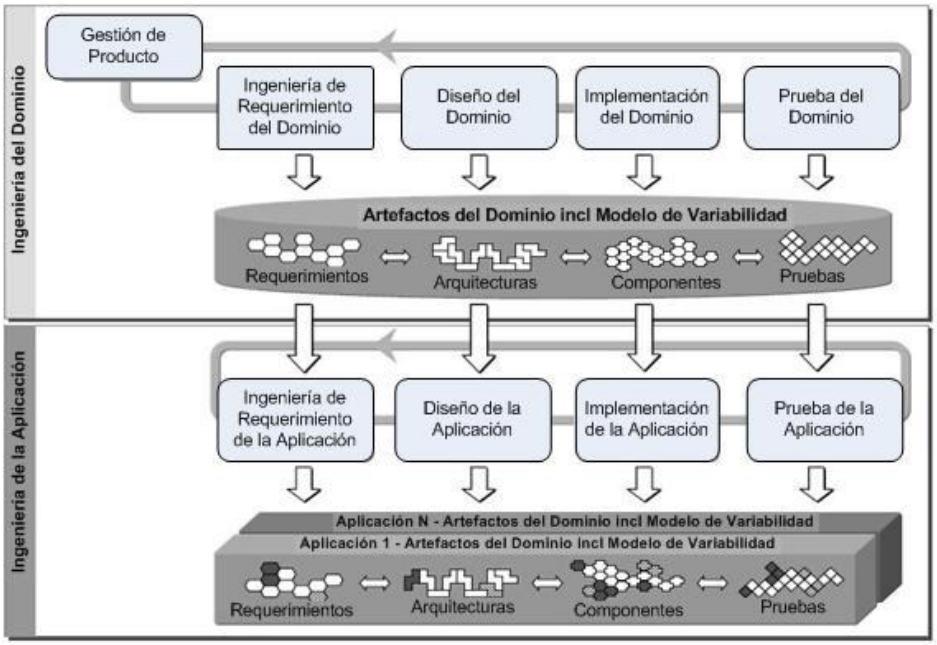


Figura 4. Procesos, subprocesos y salidas de la ingeniería de líneas de producción de software (Pohl et al, 2005).

Ingeniería de dominio: es el proceso de ingeniería de la línea de producción de software en el que se definen y realiza el carácter común y la variabilidad de la línea de producción. Está compuesto de cinco sub-procesos llamados: gestión de productos, los requisitos de ingeniería dominio, diseño de dominio, la construcción de dominio y la prueba de dominio. El proceso de ingeniería de dominio define las características comunes y la variabilidad de la línea de producción de software, los conjuntos de aplicaciones de la línea de producción de software que están previstas, es decir, delimitar el alcance de la línea de producción de software y construir objetos reutilizables que realizan la variabilidad deseada.

Ingeniería de la Aplicación: es el proceso de ingeniería de la línea de producción de software en donde las aplicaciones de la línea de producción se construyen mediante la reutilización de objetos de dominio y explotación de la variabilidad de la línea de producción. Este proceso esta integrado por los sub-procesos de aplicación de ingeniería de requisitos, diseño de la aplicación, la construcción de aplicaciones y

pruebas de aplicaciones. Los objetivos principales de la ingeniería de la aplicación son:

- Una alta reutilización, como sea posible, de los bienes de dominio a la hora de definir y desarrollar una aplicación de la línea de producción.
- Explotar la convergencia y la variabilidad de la línea de producción de software durante el desarrollo de una aplicación de línea de producción.
- Documentos relacionados a los artefactos de la aplicación, es decir, los requisitos de aplicación, la arquitectura, componentes, y las pruebas, conectados con los objetos del dominio.

Línea de Producción de Software

Según (Krueger, 2006), las líneas de producción de software se pueden describir en términos de cuatro conceptos simples:

- Entradas de activos de software: una colección de activos de software (requisitos, componentes de código fuente, casos de prueba, arquitectura y documentación), se pueden configurar y combinar de diferentes maneras para crear todos los productos en una línea de producción. Cada uno de los activos tiene un papel bien definido dentro de una arquitectura común para la línea de producción. Para dar cabida a la variación entre los productos, algunos de los activos pueden ser opcionales y otros pueden tener puntos internos de la variación que se pueden configurar de diferentes maneras para proporcionar un comportamiento diferente.
- El modelo de decisión y las decisiones de producto: El modelo de decisión describe las características opcionales y variables para los productos en la línea de producción. Cada producto en la línea de producción es un poco ambigua por las decisiones de su producto y las opciones existentes para cada una de las características opcionales y variables en el modelo de decisión.

- El mecanismo de la producción y el proceso: los medios para la composición y configuración de los productos procedentes de las entradas de los activos de software. Estas decisiones se utilizan durante la producción para determinar cuales insumos de activos de software serán usados y cómo configurar los puntos de variación dentro de esos activos.

- Salidas de productos de software: la colección de todos los productos que se pueden producir para la línea de producción. El alcance de la línea de producción está determinada por el conjunto de productos de software que puede ser producidos a partir de los activos de software y el modelo de decisión.

A continuación, en la figura 5, se muestra las relaciones de los conceptos básicos.

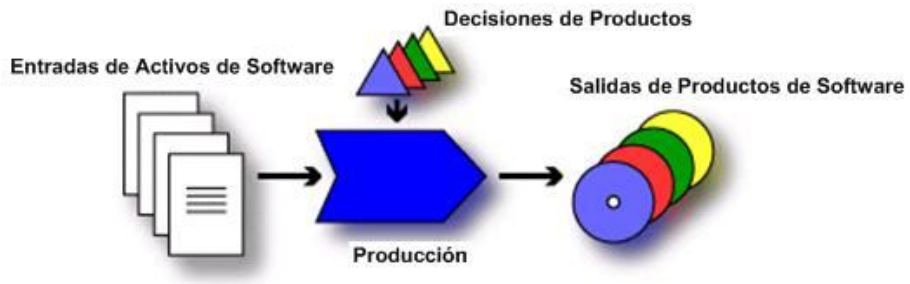


Figura 5. Conceptos Básicos de línea de producción de software Fuente: (Krueger, 2006).

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.

Ahora le presentamos 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 estratégicos de negocios:
  - Reducción en el tiempo de entrega (time-to-market), 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 reputación de la empresa.
  - Mayores márgenes de ganancias.
  - Mejor calidad de los productos.
  - Mayor escalabilidad del modelo de negocios en términos de productos y mercados y agilidad para expandir el negocio a nuevos mercados.

Framework de Líneas de Producción de Software

De acuerdo al Framework de Prácticas Líneas de Producción de Software, según (Northrop, 2008), lo define como un marco conceptual que puede servir de base para un plan de tecnología y mejoras de productos destinados a lograr los objetivos de la línea de desarrollo. Este marco esta compuesto por tres (3) actividades esenciales, distribuidas en veintinueve (29) áreas de práctica. Las actividades esenciales son las siguientes: Desarrollo de Activos Centrales (Core Asset Development), Desarrollo del Producto (Product Development) y la Gestión (Management). Adicionalmente, se sugiere aplicar las veintinueve (29) áreas de práctica que se encuentran distribuidas en tres (3) categorías que son Ingeniería de Software (Software Engineering), Gestión Técnica (Technical Management) y Gestión Organizacional (Organizational Management). Ver Figura 6.

ÁREAS PRACTICAS		
Ingeniería de Software	Gestión Técnica	Gestión Organizacional
Definición de la Arquitectura	Gestión de Configuración	Construcción del Caso de Negocio
Evaluación de la Arquitectura	Recolección de datos, indicadores y seguimiento	Gestión de Relaciones con el Cliente
Desarrollo de Componente	Análisis de Marca/Compra/Minas/ Comision	Desarrollo de una estrategia de Adquisición
Utilización de COTS	Definición de Procesos	Financiamiento
Activos de Minería Existentes	Alcance	Lanzamiento e Institucionalización
Ingeniería de Requerimientos	Técnicas de Planificación	Análisis de Mercado
Integración de Sistemas de Software	Técnicas de Gestión de Riesgos	Operaciones
Pruebas	Herramientas de Soporte	Planificación Organizacional
Entendimiento del Dominio Correspondiente		Gestión de Riesgos Organizacional
		Estructuración de la Organización
		Predicción de Tecnología
		Entrenamiento

Figura 6. Framework de Prácticas Líneas de Producción de Software. Áreas prácticas. Fuente: (Northrop, 2008).

En (Domínguez et al, 2005), citando a (Clement et al, 2001), describe a continuación las actividades esenciales:

- **Desarrollo de Activos Centrales (Core Asset Development)**: Esta actividad se enfoca en la producción de recursos generales para cualquier producto. Aquí se define el alcance de la línea de productos, el plan de producción, los componentes, la arquitectura, los requerimientos, así como también las restricciones, los estilos, patrones y frameworks, entre otros. Se configura también el inventario de activos (assets) preexistentes.

- **Desarrollo del Producto (Product Development):** Esta actividad se encarga de ensamblar los activos desarrollados o almacenados en la actividad anterior para crear productos acordes con los requerimientos del mercado; sin embargo, este proceso raramente es lineal. La creación de productos implica una continua relación con el alcance, el plan de producción y los activos centrales (core assets).

- **Gestión (Management):** La dirección juega un rol crítico en el éxito de la línea de producción. Las actividades deben tener unos recursos asignados,

coordinados y supervisados. La dirección, tanto a nivel técnico como organizacional debe estar fuertemente asociada al esfuerzo de la línea de producción de software. Pero la dirección no sólo está dirigida hacia adentro, también se debe tener en cuenta la relación con los proveedores y consumidores.

Debe señalarse, que el autor, muestra una posible orientación para aplicar las veintinueve (29) practicas, una guía de cuatro (4) pasos, la cual, nos permitirán llevar a cabo la aplicación de una línea de producción de software. Estos pasos son los siguientes (Ver Figura 7):

- Revisar los Casos de Estudios.
- Revisar y elaborar Patrones.
- Probar las técnicas.
- Realizar un plan de estudio para entrenar a los miembros del equipo de desarrollo.

Figura 7. Framework de Prácticas Líneas de Producción de Software. Orientación para la ejecución de las Áreas Prácticas. Fuente: (Northrop, 2008).

Calidad del Software

Actualmente los sistemas de información son fundamentales en casi todo los ámbitos de la vida de la personas y su correcto funcionamiento es crucial para el éxito de los negocios, la comunicación entre las personas y para la seguridad de las mismas, por tanto, desarrollar o seleccionar sistemas de software de alta calidad constituye un aspecto importante. Se define calidad del producto de software como el conjunto total de características de una entidad que le confieren la capacidad de satisfacer las necesidades establecidas y las necesidades implícitas (ISO 9126-1,2001), desde donde se extraen una serie de parámetros básicos que deben tomarse en cuenta, al desarrollar software de calidad.

Modelo de Calidad

En la actualidad los sistemas basados en computadoras son fundamentales para casi todos los ámbitos de la vida de las personas y su correcto funcionamiento es crucial para el éxito de los negocios, la comunicación, la seguridad, la educación, entre otros, por tanto, desarrollar o seleccionar sistemas de software de alta calidad constituye un aspecto bien importante.

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.

Si se desea construir productos y servicios de buena calidad para el consumidor será necesario especificar: qué calidad de producto o servicio se requiere (calidad que desea el cliente), una planificación que incorpore calidad en el diseño y métodos de producción que permitan calidad en el desarrollo.

A finales de la década del 80 e inicios del 90 se hizo énfasis en los conceptos de calidad de producto y satisfacción del usuario desde diversos enfoques, particularmente a la valoración y certificación de la calidad de procesos de producción de software; como lo son CMM (Capability Maturity Model), modelo de evaluación de procesos desarrollado por la Universidad Carneige-Mellon, SPICE (Software Process Improvement and Capability Determination), modelo para la mejora y evaluación de los procesos de desarrollo y mantenimiento de sistemas y productos de software, SQUID (Software Quality in the Development Process), modelo que apoya el control, planificación y evaluación de la calidad de software entre otros; sin embargo, también es conocido que los modelos de calidad ya eran reconocidos en la comunidad científica al final de la década del 70 como los descritos por McCall y Boehm.

Modelo de Calidad ISO/IEC 25010

Por su parte, 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 2007. 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 8:

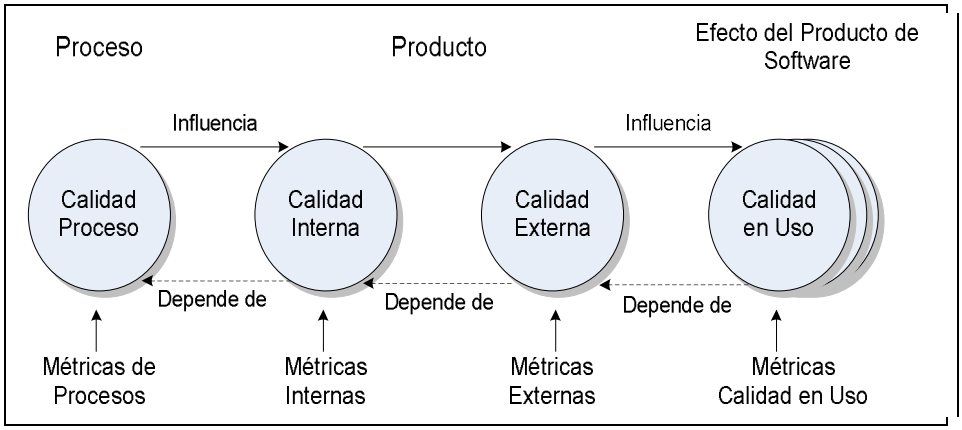


Figura 8. Enfoques de Calidad. Fuente: (ISO/IEC 25010, 2009).

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

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

La primera parte del modelo especifica ocho características para la calidad interna y externa como se muestra en la figura 9, que se subdividen más a fondo en sub-características, que se manifiestan externamente cuando el software se utiliza como parte de un sistema informático, y es resultado de las cualidades internas del software. Este estándar internacional no elabora el modelo para la calidad interna y externa de bajo del nivel de sub-características.

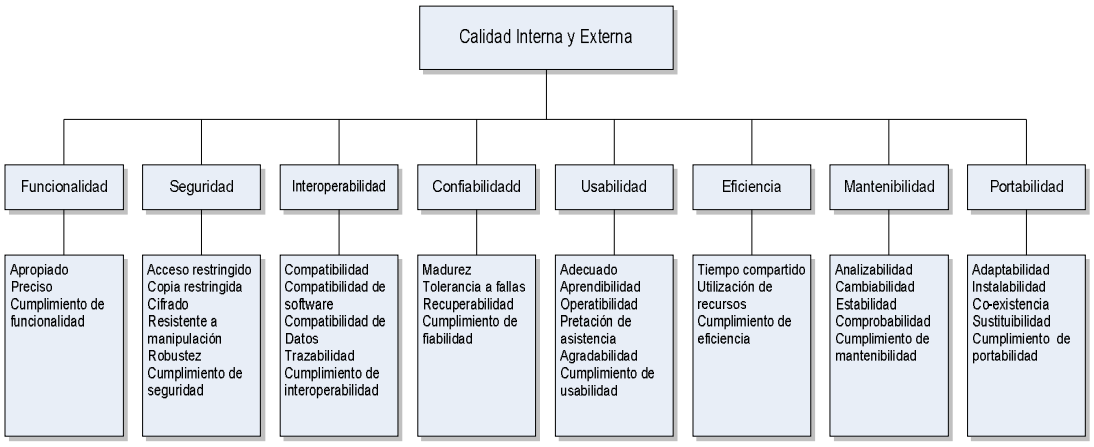


Figura 9. Características y subcaracterísticas en ISO 25010. Fuente: (ISO/IEC 25010, 2009).

La segunda parte del modelo especifica cinco características de la calidad en uso, como se muestra en la figura 10, que es el efecto combinado para el usuario de las ocho características de la calidad del producto de software. Las características definidas son aplicables a todo tipo de software.

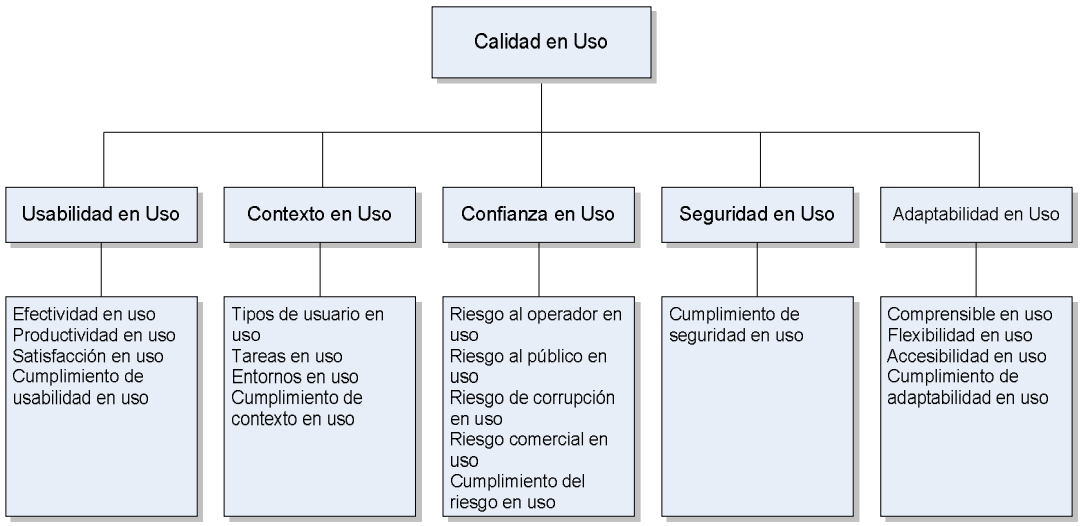


Figura 10. Características y subcaracterísticas en calidad de uso. Fuente: (ISO/IEC 25010, 2007).

Un modelo de la calidad se puede utilizar para apoyar la especificación y la evaluación del software de diversas perspectivas por aquellas personas asociadas al proceso de adquisición, requisitos, desarrollo, uso, evaluación, soporte, mantenimiento, garantía de calidad y auditoria del software. Puede, por ejemplo, ser utilizado por los desarrolladores, compradores, personal evaluador de la garantía de calidad y evaluadores independientes, particularmente aquellos responsables de especificar y de evaluar la calidad del producto de software.

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

El proceso para la Ingeniería del Dominio basado en Calidad de Software denominado **InDoCaS** (Canelón, 2010), tiene como objetivo obtener una arquitectura base para una familia, aplicando las actividades correspondiente a las disciplinas de análisis y diseño del dominio. Este proceso desarrollado para la ingeniería de dominio puede ser utilizado en el enfoque de desarrollo de línea de producción de software, el cual puede ser instanciado para un dominio especifico y los activos de software producidos pueden ser reutilizados para generar un producto de una familia particular del dominio.

Según (Canelón, 2010), La estructura del proceso está basada en lo siguiente:

- RECLAMO (Requirement Clasification Model): Modelo de clasificación de requisitos propuesto por Chirinos y otros. (Chirinos et al., 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. (Losavio et al., 2008).

- FODA (Feature-Oriented Domain Analisys): análisis del dominio orientado a rasgos, desarrollado por el SEI (Software Engineering Institute) para el modelo de variabiliad. (Kang et al., 1990).

- Un proceso general para el diseño arquitectónico del dominio propuesto por Hofmeister y otros. (Hofmeister et al., 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 Analysis Method): Método para elegir una arquitectura para un sistema, utilizado en InDoCaS para la evaluación arquitectural.

Para la presentación de este proceso, se utiliza la notación SPEM 2, en la figura 11 se muestran las disciplinas del proceso **InDoCaS**, mencionadas anteriormente.

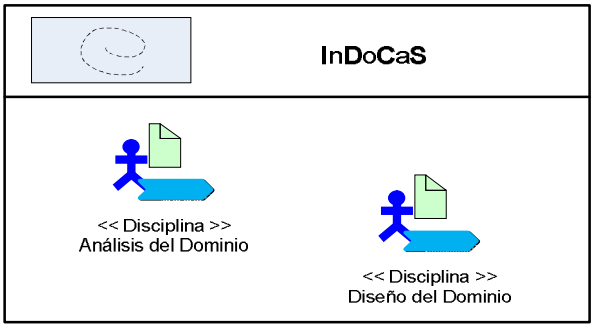


Figura 11. Disciplinas del proceso **InDoCaS** (*Canelón, 2010*).

Adicionalmente, **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 las tablas 2 y 3:

Tabla 2. 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 3, en el caso de las actividades se utiliza: *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 3. Esquema de Artefacto **I**n**D**o**C**a**S**

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

Fuente: Canelón, 2010.

Por su parte en la tabla 3, se muestran los elementos para definir un artefacto **I**n**D**o**C**a**S** a saber: 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).

A continuación presentaremos las descripciones de las disciplinas Análisis del Dominio y Diseño del Dominio de la ingeniería de dominio, puesto que representa la base de la presente investigación ya que proporcionan los componentes que forman la arquitectura base de la línea de producto.

Análisis del Dominio (InDoCaS)

En la fase del análisis del dominio del proceso **I**n**D**o**C**a**S** se definen los aspectos comunes a todas las familias del dominio y los particulares de cada una de ellas. Este subproceso permite la caracterización del dominio, identifica las propiedades de calidad que deben ser garantizadas y define el modelo de calidad asociado al dominio que brinda soporte al resto de las fases. La disciplina de análisis del dominio está conformada por seis actividades: identificación de requisitos, obtener modelo de similitudes y variabilidad, identificación de propiedades de calidad asociadas al conjunto minimal de requisitos funcionales y no funcionales, obtener el modelo de calidad asociado al dominio, creación de escenarios de calidad del dominio e identificar los estilos arquitecturales para el dominio (Ver figura 12).

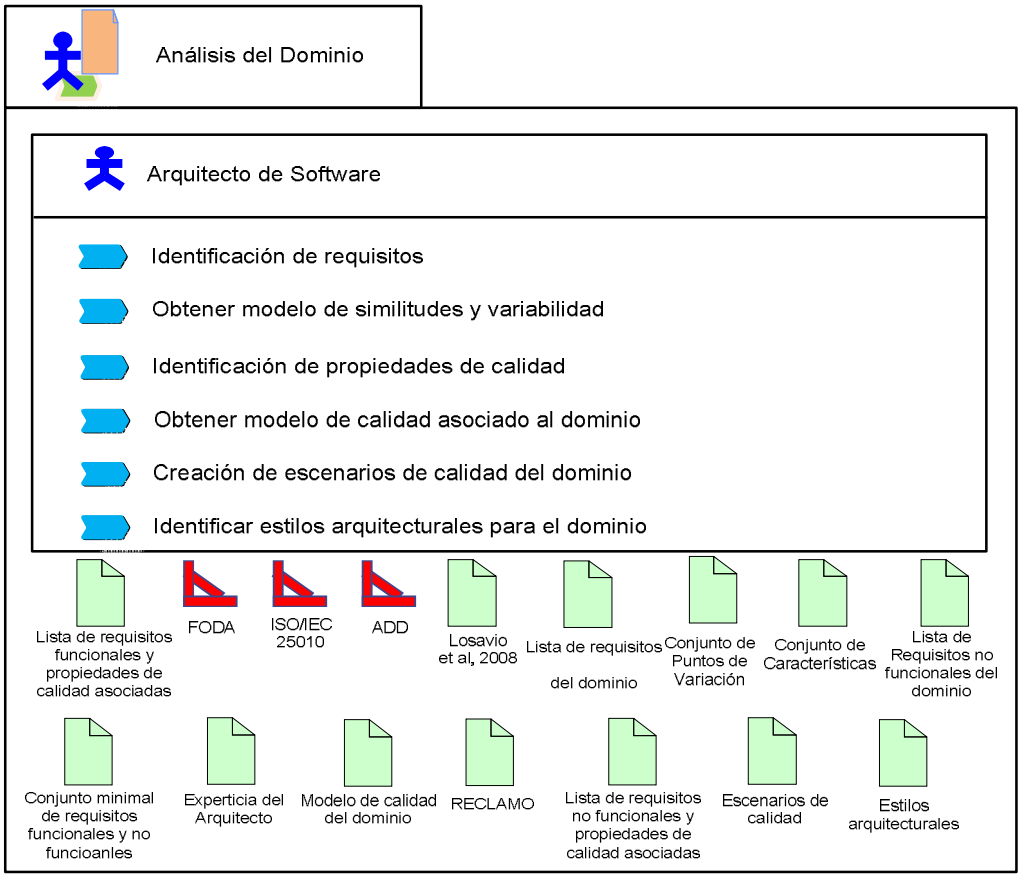


Figura N° 12. Análisis del dominio **InDoCaS** (Canelón, 2010)

En la figura 13, se muestra el diagrama de actividades de la disciplina análisis del dominio, en el cual se muestran las entradas y las salidas de cada una de las actividades y la secuencia de ejecución, del mismo modo se indica qué técnicas intervienen en cada una de las actividades.

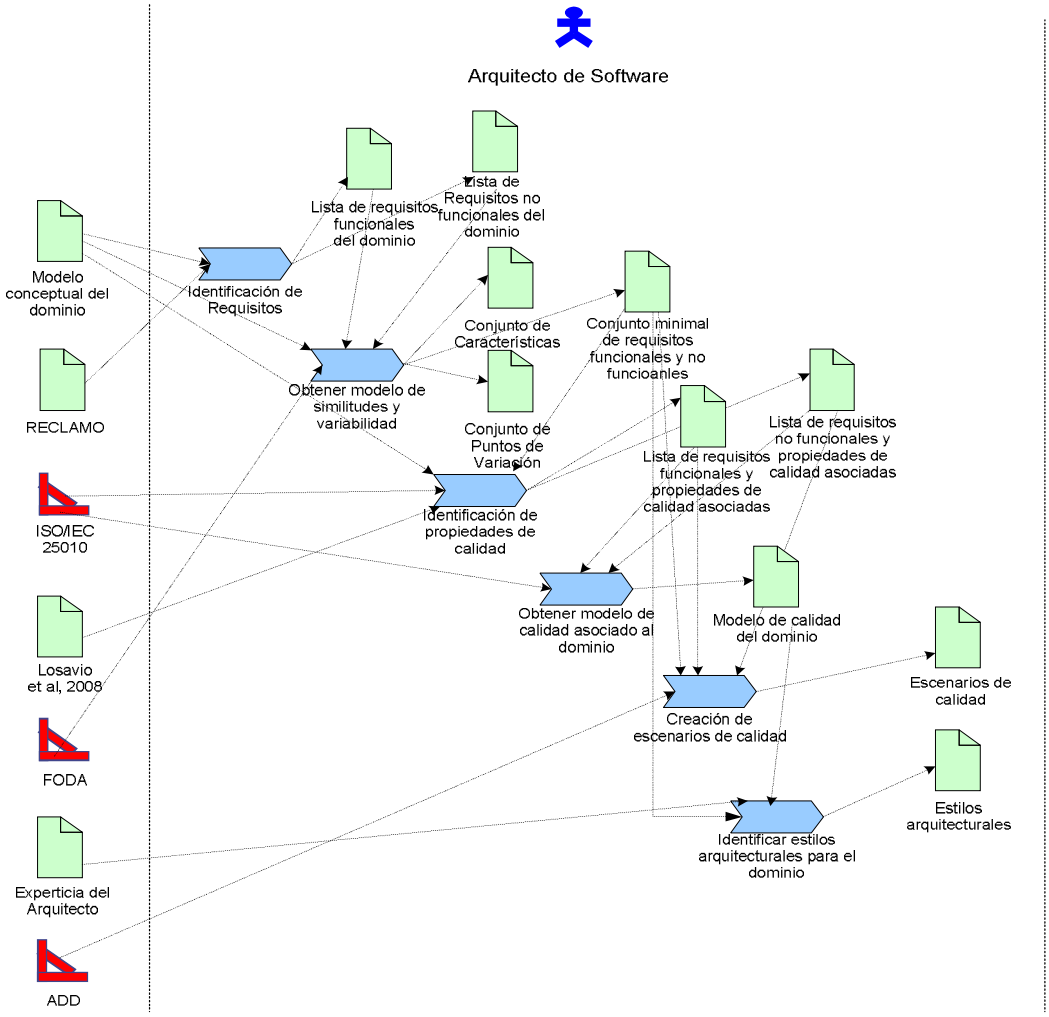


Figura 13. InDoCaS: Diagrama de actividades de la Disciplina de análisis del dominio. Fuente: (Canelón, 2010).

Diseño del Dominio (InDoCaS)

En la fase del diseño del dominio, se especializa las actividades para la síntesis arquitectural y evaluación arquitectural del dominio (Ver figura 14), proponiendo y adaptando un conjuntos de técnicas específicas para la construcción de dichas actividades y artefactos.

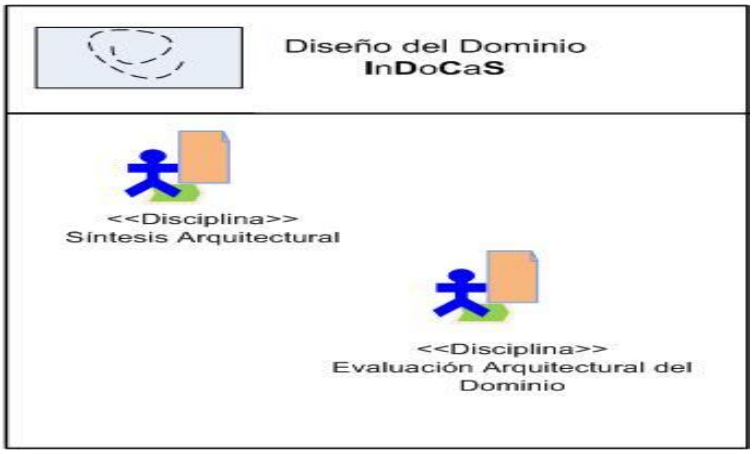


Figura N° 14. InDoCaS: Disciplina de Diseño del dominio (Canelón, 2010)

En la figura 15, se presentan el conjunto de actividades y artefactos que componen la síntesis arquitectural.

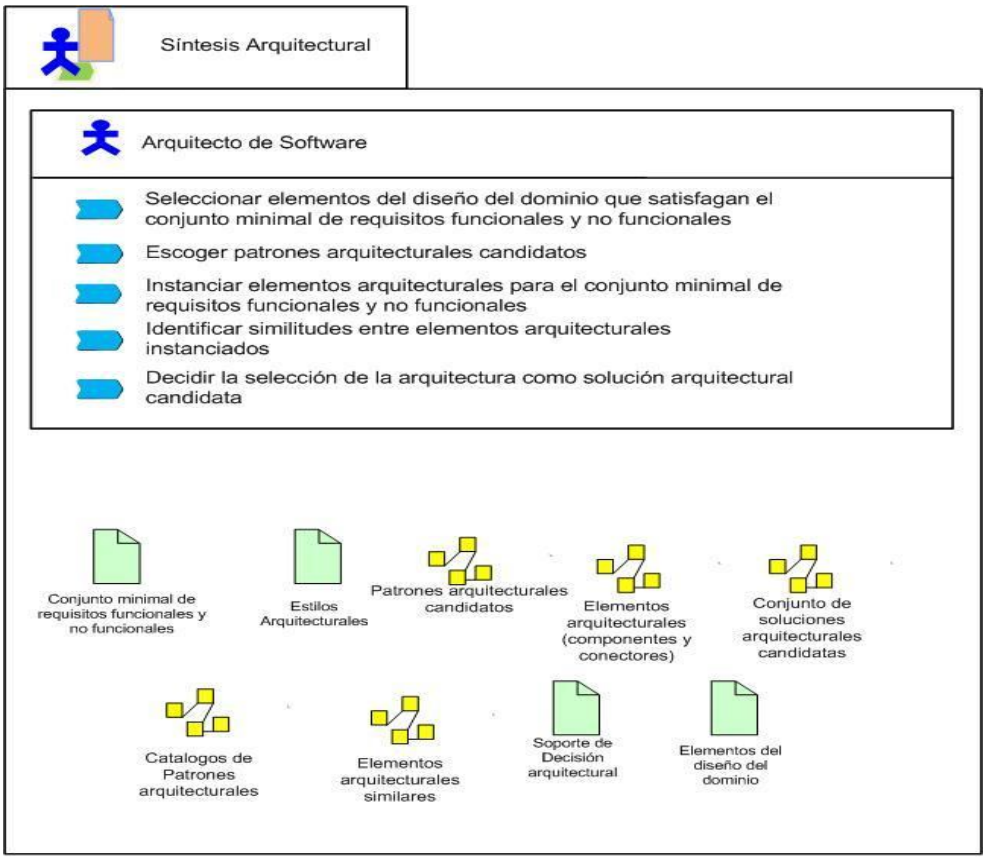


Figura N° 15. InDoCaS: Disciplina de Diseño del dominio, subproceso para Síntesis Arquitectural (Canelón, 2010)

La síntesis arquitectural es el núcleo del diseño de la arquitectura, este subproceso propone soluciones consistentes a un conjunto de requisitos arquitecturales, por lo que se traslada del problema a un espacio de solución. Como se muestra en la figura 16, las entradas a esta fase son los artefactos: Conjunto minimal de requisitos funcionales y no funcionales, Estilos arquitecturales y Catálogos de patrones arquitecturales obtenidos en la disciplina del Análisis del Dominio.

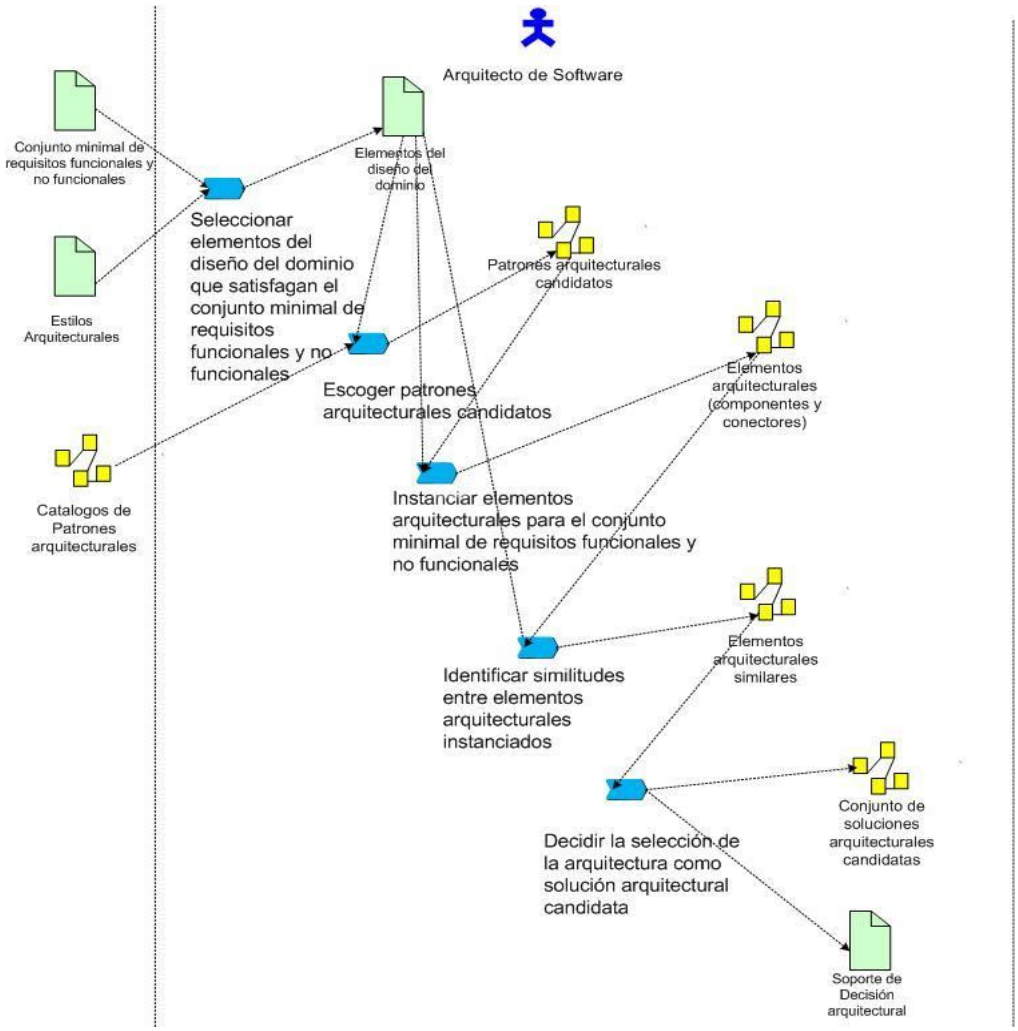


Figura N° 16. InDoCaS: Diagrama de actividades subproceso para Síntesis arquitectural (Canelón, 2010)

La salida de la síntesis arquitectural identificada como conjunto de soluciones candidatas, son arquitecturas candidatas para la solución al conjunto de requisitos arquitecturales pudiendo ser soluciones alternativas, incluso parciales (partes de arquitectura). En ellas se reflejan las decisiones de diseño acerca de la estructura de software, reflejadas inicialmente en el artefacto estilo arquitectural, obtenido en la disciplina del Análisis del Dominio y en este subproceso se incorpora la información del proceso de decisión.

En las actividades de la evaluación arquitectural del dominio (Ver figura 17), se busca analizar e identificar riesgos potenciales en su estructura y sus propiedades, que puedan afectar al sistema de software resultante, verificar que los requisitos no funcionales estén presentes en la arquitectura, así como determinar en que grado se satisfacen los atributos de calidad.

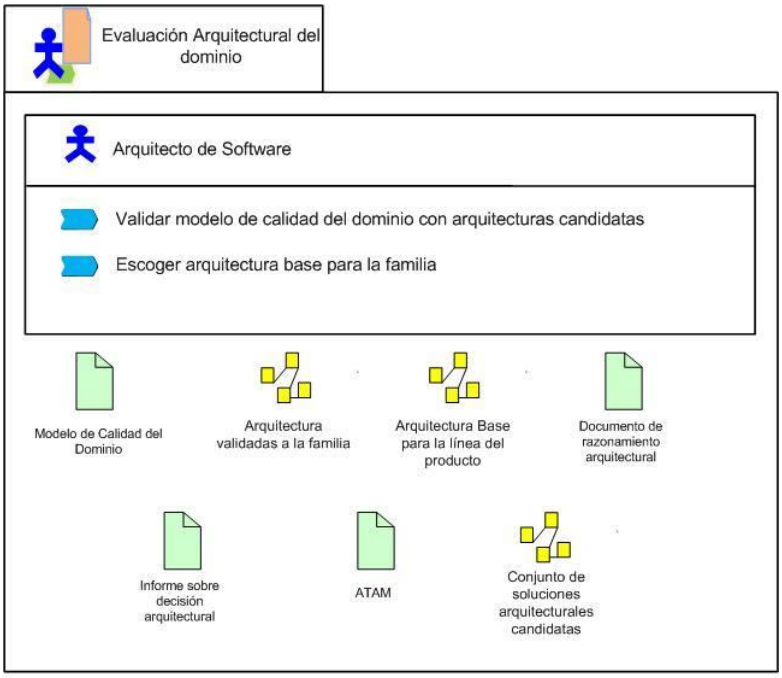


Figura N° 17. InDoCaS: la disciplina de Diseño del dominio, subproceso para Evaluación arquitectural del Dominio (Canelón, 2010)

En la figura 18, se presentan las dependencias encontradas en la evaluación arquitectural del dominio.

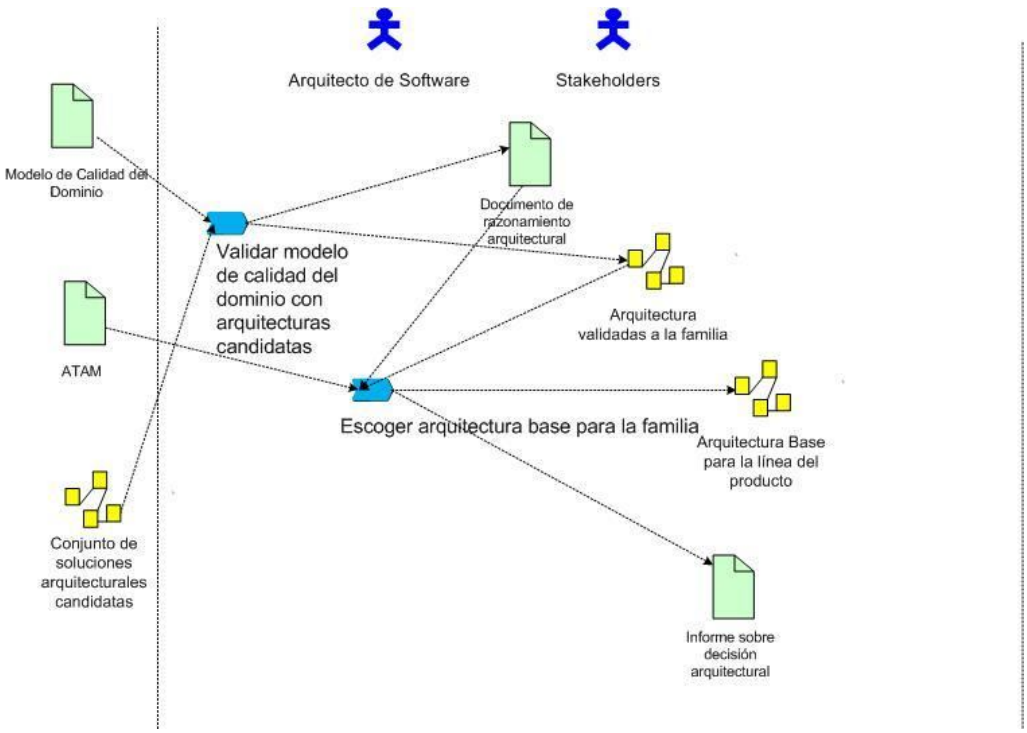


Figura N° 18. InDoCaS: Diagrama de actividades subproceso para Evaluación arquitectural del Dominio (Canelón, 2010)

En la tabla 4, se resumen las actividades de las disciplina análisis del dominio y diseño del dominio, asociando los artefactos de entrada a cada actividad y los artefactos que produce la misma.

Tabla 4. Resumen de Actividades y Artefactos de **InDoCaS**

Disciplina	**Actividades**		**Artefactos**	
Análisis del Dominio	**Nro.**	**Nombre de la actividad**	**Artefactos de Entradas**	**Artefactos de Salida**
	A_01	Identificación de Requisitos	- Modelo conceptual del dominio. - Reclamo.	- Lista de requisitos funcionales. - Lista de requisitos no funcionales.
	A_02	Obtener modelo de similitudes y variabilidad	- Modelo conceptual del dominio. - Lista de requisitos funcionales.	- Conjunto de características. - Conjunto minimal de requisitos funcionales

				- Lista de requisitos no funcionales.	y no funcionales. - Conjunto de puntos de variación
		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.
		A_04	Obtener modelo de calidad asociado al 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.
		A_05	Creación de escenarios de calidad del dominio	- Modelo de calidad del dominio como. - 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.
		A_06	Identificar los estilos arquitecturales para el dominio	- Conjunto minimal de requisitos funcionales y no funcionales. - Modelo de calidad del dominio como.	-Estilos arquitecturales.
Diseño del Dominio	**Síntesis Arquitectural**	A_07	Seleccionar los elementos del diseño del dominio que satisfagan el conjunto minimal de requisitos funcionales y no funcionales	- Conjunto minimal de requisitos funcionales y no funcionales. -Estilos arquitecturales.	-Elementos del diseño del dominio
		A_08	Escoger patrones arquitecturales candidatos	-Catalogo de Patrones arquitecturales. -Elementos del diseño del dominio.	-Patrones arquitecturales candidatos.
		A_09	Instanciar elementos	-Elementos del diseño	-Elementos

			arquitecturales para los elementos del diseño del dominio	del dominio. -Patrones arquitecturales candidatos.	Arquitecturales (Componentes y conectores).
		A_10	Identificar similitudes entre los elementos arquitecturales instanciados	-Elementos del diseño del dominio. -Elementos Arquitecturales (Componentes y conectores).	-Elementos Arquitecturales Similares.
		A_11	Decidir la selección de la arquitectura como solución arquitectural candidata	-Elementos Arquitecturales Similares.	-Conjunto de soluciones arquitecturales candidatas. -Soporte de decisión arquitectural.
	Evaluación Arquitectural del dominio	A_12	Validar modelo de calidad del dominio con arquitectura candidatas	- Modelo de calidad del dominio. -Conjunto de soluciones arquitecturales candidatas.	-Documento de razonamiento arquitectural. -Arquitectura Validadas a la familia.
		A_13	Escoger arquitectura base para la familia	-Documento de razonamiento arquitectural. -Arquitectura Validadas a la familia.	-Arquitectura base para la línea del producto. -Informe sobre decisión arquitectural.

Fuente: Canelón, 2010.

Método WATCH

El método WATCH surgió de la necesidad de un modelo de proceso para el desarrollo de proyectos de software pequeños y medianos desarrollado por pequeñas empresas, este fue el resultado de la integración de modelos de desarrollo de software tales como: espiral, V y orientado a objetos descrito por Bruegge y Dutoit, (2000). El estándar (IEEE 1074, 1995) para el desarrollo de procesos de ciclos de vida permitió el marco metodológico necesario para desarrollar al modelo WATCH. El método WATCH en su más recientemente versión, es un marco metodológico que describe los procesos técnicos, gerenciales y de soporte que deben emplear los equipos de trabajo que tendrán a su cargo el desarrollo de aplicaciones de software empresarial Montilva et al., (2008).

Un marco metodológico es un patrón que debe ser instanciado, es decir adaptado cada vez que se use. Cada equipo de trabajo deberá usar el método como un patrón o plantilla metodológica, a partir de la cual dicho equipo debe elaborar el proceso específico de desarrollo de la aplicación que se desea producir.

El método WATCH está fundamentado en las mejores prácticas de la Ingeniería de Software y la Gestión de Proyectos. Cubre todo el ciclo de vida de las aplicaciones; desde el modelado del dominio de la aplicación, pasando por la definición de los requisitos de los usuarios, hasta la puesta en operación de la aplicación.

Este método incluye, también, una descripción de los procesos de gerencia del proyecto que se aplicarán para garantizar que el proyecto se ejecute en el tiempo previsto, dentro del presupuesto acordado y según los estándares de calidad establecidos.

En este método se emplearon, como marcos de referencia para la elaboración de los elementos que integran el método, los siguientes estándares, prácticas y modelos:

- El modelo CMMI-SW(Capability Maturity Model Integration) del Instituto de Ingeniería de Software – SEI[7] (CMMI, 2005).
- El cuerpo de conocimientos de la Ingeniería de Software (SWEBOK) de la Sociedad de Computación de la IEEE[8].
- El cuerpo de conocimientos PMBOK (Project Management Body of Knowledge) del Instituto de Gestión de Proyectos (PMI[9], 2000).
- Estándares de desarrollo de software de la Sociedad de Computación de la IEEE.
- Modelos de procesos de los enfoques de desarrollo de software siguientes:
    - Desarrollo guiado por modelos (Model Driven Development).
    - Desarrollo guiado por pruebas (Test Driven Development).

[7] SEI: siglas de Software Engineering Institute.
[8] IEEE: siglas de Institute of Electrical and Electronics Engineers.
[9] PMI: siglas de Project Management Institute

- Las mejores prácticas de la Ingeniería de Software (Krutchen, 2000):
    - Desarrollo iterativo, incremental y versionado.
    - Ingeniería de Requisitos.
    - Arquitecturas basadas en componentes de software.
    - Uso de lenguajes de modelado visual: UML y UML Business.
    - Gestión integral del proyecto.
    - Verificación y validación de la calidad de los productos y procesos.
    - Gestión de la configuración (control de cambios).

El método WATCH está compuesto por tres modelos fundamentales:

- Modelo de productos que describe los productos intermedios y finales que se generan, mediante el uso del método, durante el desarrollo de una aplicación empresarial.

- Modelo de actores que identifica a los actores interesados (stakeholders) en el desarrollo de una aplicación y describe cómo deben estructurarse los equipos de desarrollo y cuáles deben ser los roles y responsabilidades de sus integrantes

- Modelo de procesos que describe detalladamente los procesos técnicos, gerenciales y de soporte que los equipos de desarrollo deberán emplear para elaborar las aplicaciones.

Las actividades del método WATCH se muestran gráficamente en la figura 19 y su estructura emplea la metáfora de un reloj de pulsera para describir el orden de ejecución de los procesos técnicos de desarrollo de aplicaciones; los procesos gerenciales están ubicados en el centro de la estructura para indicar explícitamente que ellos programan, dirigen y controlan el proceso de desarrollo. Los procesos técnicos están ubicados en el entorno siguiendo la forma como se mueven las agujas de un reloj. Ello indica que el orden de ejecución de las fases técnicas se realiza en el

sentido de las agujas. Los procesos gerenciales pueden, sin embargo, invertir el orden de ejecución para repetir algunas de las fases anteriores.

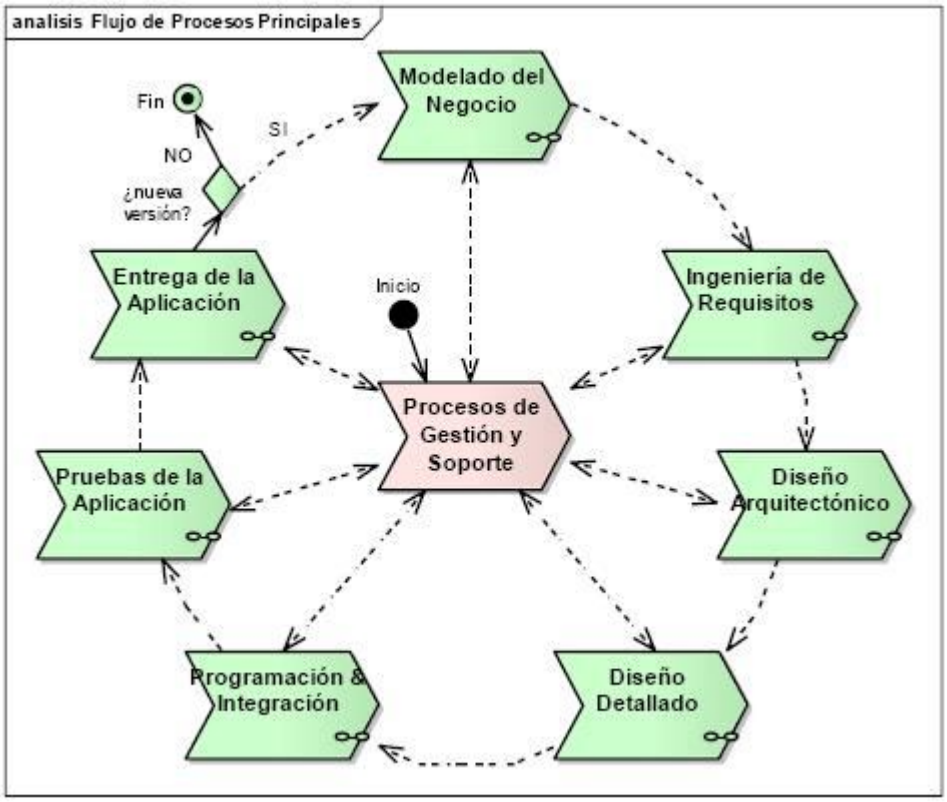


Figura N° 19. Método WATCH. (Montilva et al, 2008)

Método WATCH - Component

Es un método de desarrollo especializado en producir componentes de software reutilizable derivado de una variación al método WATCH (Hammar y Montilva, 2003), el mismo se centra en el desarrollo individual del componente. En la figura 20, se puede ver las actividades que lo componen.

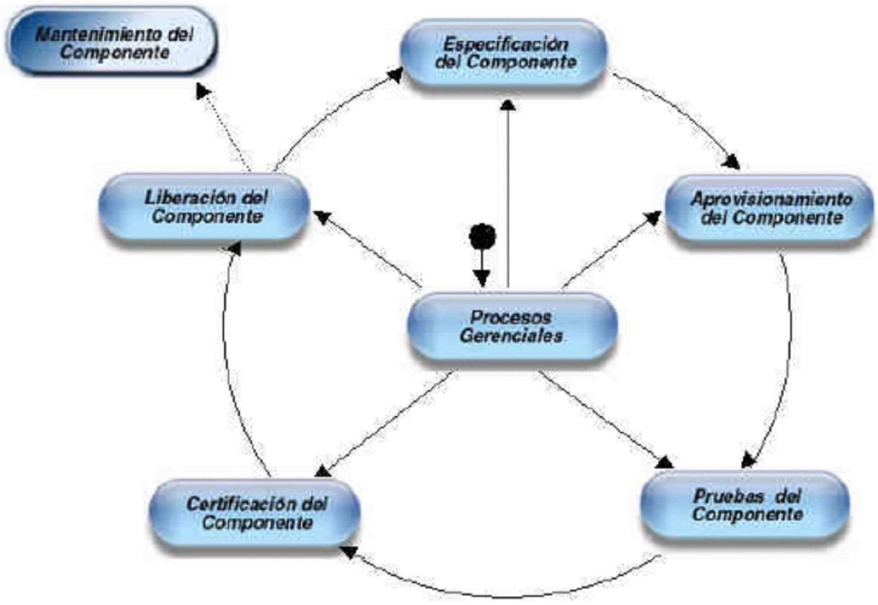


Figura N° 20. Método WATCH-COMPONENT. (Hammar y Montilva, 2003)

Este método consta de tres modelos (Hammar y Montilva, 2003):

- Modelo del producto: con el modelado de un componente de software reutilizable, se definen los componentes y se logran establecer sus propiedades, se establece su forma en cada una de las etapas propuestas por Cheesman y Daniles, (2001) y las etapas por las que va a pasar el componente y que deben ser tomadas en cuenta en el modelo de procesos, para crear así una relación entre ellos.

- Modelo del proceso: Se definen cada una de las actividades necesarias para producir el componente de software.

- Modelo del grupo de desarrollo: describe los actores y roles del grupo de desarrollo del componente.

Aunque se siguen los principios del método WATCH, este método incluye dos nuevos aspectos como son:

- La toma de decisiones acerca de las posibilidades del aprovisionamiento de los componentes en las fases respectivas y dependiendo de estas decisiones establece los puntos a seguir en cada una de las etapas.

- Esta diseñado para modelar el ciclo de vida de un solo componente reutilizable y no una aplicación integrada por componentes.

Sistema de Información

Los sistemas de información, han sido, son y serán la piedra fundamental de toda organización para la toma de decisiones, y es que a través de éstos sistemas, las organizaciones tienen una gestión más eficiente, "El valor de la informática o de los sistemas de información está dado por las mejoras en la eficiencia de la gestión de la organización que se sirven de esos sistemas de información", (Fábregas y Bauza, 1991).

Tipo de Sistema de Información

Según (Laudon, 2006), los sistemas de información pueden clasificarse en cuanto a la función a la que vayan destinados o el tipo de usuario final del mismo en:

- **Sistema de procesamiento de transacciones (TPS).-** Gestiona la información referente a las transacciones producidas en una empresa u organización.

- **Sistemas de información gerencial (MIS).-** Orientados a solucionar problemas empresariales en general.

- **Sistemas de soporte a decisiones (DSS).-** Herramienta para realizar el análisis de las diferentes variables de negocio con la finalidad de apoyar el proceso de toma de decisiones.

- **Sistemas de información ejecutiva (EIS).-** Herramienta orientada a usuarios de nivel gerencial, que permite monitorizar el estado de las variables de un área o unidad de la empresa a partir de información interna y externa a la misma.

- **Sistemas de automatización de oficinas (OAS).-** Aplicaciones destinadas a ayudar al trabajo diario del administrativo de una empresa u organización.

- **Sistema experto (SE).-** Emulan el comportamiento de un experto en un dominio concreto.

- **Sistema Planificación de Recursos (ERP).-** Integran la información y los procesos de una organización en un solo sistema.

Transacciones

Las transacciones son interacciones que pueden llevar a cabo entre entidades separadas u objetos, a menudo relacionadas con el intercambio de objetos valiosos, tales como información, bienes, servicios y dinero. Según (Moreno et al, 2003), una transacción se define "como una unidad de consistencia y de recuperación, por lo tanto, cuando cada acceso a la base de datos es hecho usando la interface de transacción, se garantiza su estado consistente".

Si se ejecuta una secuencia de operaciones dentro de una transacción se asegura que esas operaciones cumplan con las propiedades ACID(Atomicidad, Consistencia, Aislamiento y Durabilidad) (Moreno et al, 2003):

- Atomicidad (Atomicity): todos los efectos de todas las operaciones de la transacción en la base de datos resultan con éxito o ninguno de ellas. Por lo tanto la transacción es indivisible, atómica, unidad de trabajo.

- Consistencia (Consistency): una transacción deja la base de datos en un estado consistente, asumiendo que estaba consistente cuando comenzó.

- Aislamiento (Isolation): esta propiedad garantiza que la ejecución concurrente de transacciones no introducirá inconsistencias en la base de datos.

- Durabilidad (Durability): Los efectos de una transacción que ha hecho un Commit son permanentes en la base de datos, y garantiza que van a sobrevivir a subsecuentes errores.

Tipos de Transacciones

Según (Moreno et al, 2003), las transacciones se dividen en dos grupos: en planas y de largas vidas:

- Transacción Plana:

Se llama transacción plana porque todas las acciones que se llevan a cabo dentro de los límites de una transacción, ocurren al mismo nivel. Puede contener un número arbitrario de acciones que son tomadas como un todo y consideradas como única acción. Las primitivas necesarias para manejar este tipo de transacciones son: comenzar transacción, grabar transacción y abortar. La unidad de recuperación es la misma transacción en su totalidad, no se pueden grabar ni recuperar partes, el conjunto de acciones es indivisible, la filosofía es realizar "todo o nada", y en un único nivel. Son útiles para modelar actividades breves (Moreno et al, 2003).

- Transacciones de Larga Vida:

Las transacciones de larga vida son transacciones de larga duración. Pueden prolongarse durante días, semanas, meses o hasta años, por lo que son factibles los problemas si se aplican las mismas propiedades que a las transacciones planas. Debido a estos motivos, al automatizar los procesos de negocios, se propone utilizar transacciones de larga vida (Moreno et al, 2003).

a)Modelo Extendido para Transacciones de Larga Vida

Se utilizan modelos extendidos de transacciones para aplicaciones avanzadas, por ejemplo las orientadas a procesos, en el cual las transacciones tienen una estructura interna, donde intervienen varios actores, y toman relativamente largo tiempo para ser completadas. En estos casos las transacciones planas con sus propiedades ACID no son adecuadas por los siguientes motivos:

- La atomicidad estricta, implica que los errores durante la ejecución de una transacción de larga vida, puedan causar grandes volúmenes de trabajo al deshacer el resultado de una transacción completa (Rollback). Además se presenta la grave falencia de perder una posiblemente gran cantidad de trabajo realizado en la aplicación de Workflow durante ese tiempo.

- El aIslamiento estricto, implica que resultados intermedios de una transacción de larga vida se mantengan privados a la transacción, y no puedan ser usados para ejecutar transacciones concurrentemente hasta que la transacción halla finalizado, es decir que haya hecho un Commit. En una aplicación de Workflow con varios actores compitiendo por los recursos a través del tiempo, esta es una restricción demasiado significativa.

- Una estructura plana no permite un proceso transaccional estructurado o asignar partes de una transacción a diferentes actores (Moreno et al, 2003).

b)Modelo de SAGAS

En este modelo se define una transacción T como un conjunto de subtransacciones T1,...,Tn, donde cada componente de esa subtransacción, T1...Tn, tienen todas las propiedades ACID de las transacciones tradicionales, y estos pueden interrelacionarse entre los componentes de las subtransacciones . Sin embargo cada subtransacción del SAGAS tiene que ser ejecutado en un orden predefinido, como se presentan en la Figura 21. Para cada subtransacción Tk (1<= k < n), se define una subtransacción compensadora. Una transacción compensadora CTk semánticamente deshace los efectos de la transacción Tk, por ejemplo si se corre T1 el compensador es CT1, y el

CT1 semánticamente deshace los efectos de la transacción T1. El estado de la base de datos después de ejecutar la subtransacción Tk y luego el compensador CTk tiene como resultado la identidad. Para completar un SAGAS, se debe ejecutar de forma exitosa toda la secuencia o los efectos de las subtransacciones que realizaron Commit son deshechas por una secuencia de subtransacciones compensadoras. Las subtransacciones compensadoras son ejecutadas en orden inverso de los componentes de subtransacciones. La última subtransacción Tn no requiere una subtransacción compensadora, ya que cuando Tn realiza el Commit no hay otra subtransacción que pueda ejecutar y por lo tanto SAGAS realiza el Commit final (Moreno et al, 2003).

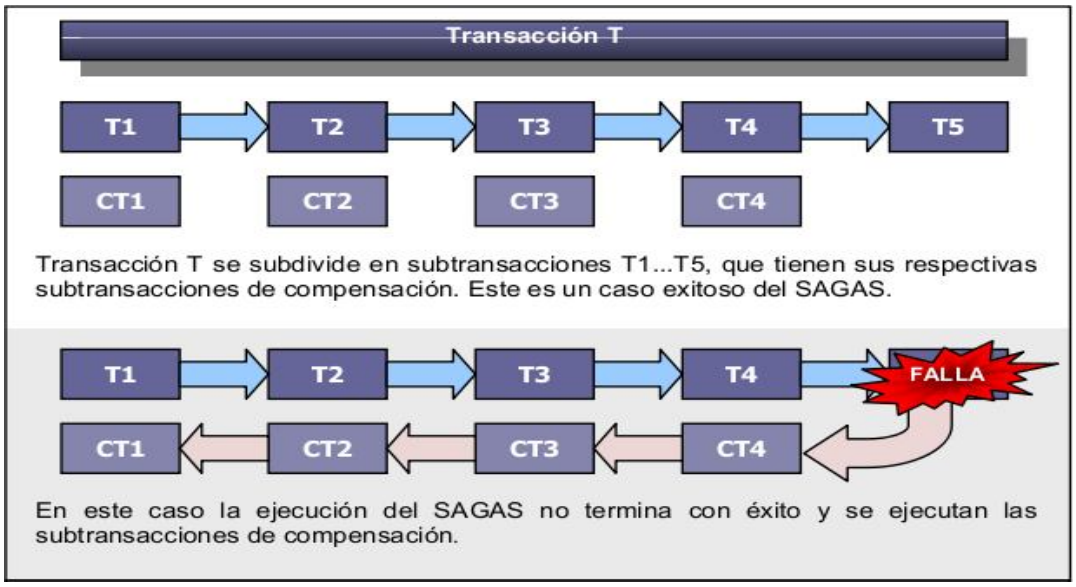


Figura 21. Ejemplo del modelo Sagas utilizados para transacciones (Moreno et al, 2003).

c)Modelo Transacción Nested

Consiste en una transacción T que se divide subtransacciones T1...Tn, en forma jerárquica formando un árbol de transacción. Una subtransacción que tiene subtransacciones es llamado padre, y sus subtransacciones son llamados hijos. Las hojas del árbol son transacciones planas. Cada subtransacción puede hacer un Commit o un Abort siempre que se consideren los siguientes aspectos:

oUna subtransacción hija puede empezar después que el padre halla empezado.

oUna subtransacción padre debe terminar sólo después que todos los hijos hallan terminado.

oSi una subtransacción padre es abortada, todos los hijos son abortados.

oSi una subtransacción hija es abortada, el padre puede elegir la forma de recuperarse, por ejemplo:

oIgnorar la falla y proceder con otras tareas, en este caso el hijo es considerado no vital.

oReintentar la subtransacción que falló.

oEjecutar otra subtransacción que ejecuta una acción alternativa.

oAbortar.

Todas las propiedades ACID son preservadas en este modelo. En este caso se mantiene el aislamiento entre cada subtransacción y en caso de una falla se realiza el Rollback sin efectos en otras transacciones.

Sin embargo, las subtransacciones si bien son atómicas y aisladas entre si, no son durables. Aún si una subtransacción realiza el Commit, sus efectos van a ser deshechos cuando el padre aborte. Las modificaciones hechas por una subtransacción llegan a ser permanentes sólo después que la raíz del árbol de transacción realiza el Commit. Similarmente los resultados de una subtransacción que realizó el Commit pueden ser usados por otra subtransacción antes que el padre realice el Commit, pero son exteriorizados sólo después del Commit de toda la transacción (Moreno et al, 2003).

d)Modelo Transacción Open Nested

Este modelo extiende el modelo NESTED, permitiendo que los requerimientos de aislamiento no sean tan rígidos, haciendo que los resultados de subtransacciones que realizaron el Commit sean visibles para otra transacción NESTED ejecutada concurrentemente. De esta manera un alto grado de concurrencia es obtenido. Para

evitar inconsistencias en el uso de los resultados de subtransacciones que realizaron un Commit, sólo a las subtransacciones que conmutan las que realizaron el Commit se les permite usar los resultados. Dos transacciones se dice que conmutan, si los efectos, por ejemplo, las salidas y el estado final de la base de datos, son independientes del orden en que son ejecutadas. En sistemas convencionales, solo operaciones de lectura conmutan. Basado en ésta semántica, sin embargo, pueden también definirse operaciones de modificación como conmutativas (Moreno et al, 2003).

e)Subtransacciones Multi-Padre

Extiende el modelo de transacción NESTED permitiendo que cada subtransacción tenga un número arbitrario de transacciones padres. Una subtransacción puede ser ejecutada en paralelo con algunas o todas las transacciones padres. Por lo que el modelo permite múltiples transacciones para comenzar una subtransacción en cooperación. La principal ventaja del modelo es que la combinación de eventos puede ser incrustado dentro de transacciones (Moreno et al, 2003).

f)Modelo Nested Sagas

Este modelo permite estructuras NESTED dentro de cada subtransacción de SAGAS, definiendo interiormente cada una de ellas como estructuras del modelo de transacción NESTED (Moreno et al, 2003).

g)Modelo de Transacciones Flexible

Este modelo ha sido propuesto como un modelo de transacción adecuado para ambientes de múltiples bases de datos. Es un conjunto de tareas, con un conjunto de funcionalidades equivalentes para cada subtransacción y un conjunto de dependencias de ejecución sobre las transacciones, incluyendo dependencias de fallas, dependencia de éxito, o dependencias externas. Para disminuir la rigidez de los requerimientos de aislamiento, usa compensación y disminuye la rigidez de los requerimientos de

atomicidad, permitiendo al diseñador de la transacción especificar estados aceptables para la terminación de la transacción flexible, en cual algunas subtransacciones pueden abortar (Moreno et al, 2003).

Sistemas Transaccionales

Según la definición del sitio web (Diccionario de Informática, 2010), los sistemas transaccionales "es un tipo de sistema de información diseñado para recolectar, almacenar, modificar y recuperar todo tipo de información que es generada por las transacciones en una organización. Una transacción es un evento o proceso que genera o modifica la información que se encuentran eventualmente almacenados en un sistema de información". Estos sistemas presentan las siguientes propiedades:

Propiedades de los sistemas transaccionales

- Automatizan tareas operativas en una organización, permitiendo ahorrar en personal.
- Suelen dirigirse especialmente al área de ventas, finanzas, marketing, administración y recursos humanos.
- Suelen ser los primeros sistemas de información que se implementan en una organización.
- Sus cálculos y procesos suelen ser simples.
- Se suelen utilizar para cargar grandes bases de datos.
- Los beneficios de este tipo de sistemas en una organización son rápidamente visibles.
- Estos sistemas son optimizados para almacenar grandes volúmenes de datos, pero no para analizar los mismos.

Características de un sistema transaccional

Las características de este tipo de sistemas se presentan a continuación:

- Para que un sistema informático pueda ser considerado como un sistema transaccional, debe poseer las propiedades de las transacciones ACID (Atomic, Consistent, Isolated, Durable).
- Rapidez: deben ser capaces de responder rápidamente, en general la respuesta no debe ser mayor a un par de segundos.
- Fiabilidad: deben ser altamente fiables, de lo contrario podría afectar a clientes, al negocio, a la reputación de la organización, etc. En caso de fallas, debe tener mecanismos de recuperación y de respaldo de datos.
- Inflexibilidad: no pueden aceptar información distinta a la establecida. Por ejemplo, el sistema transaccional de una aerolínea debe aceptar reservas de múltiples agencias de viajes. Cada reserva debe contener los mismos datos obligatorios, con determinadas características (Diccionario de Informática, 2010).

CAPÍTULO III

MARCO METODOLÓGICO

Tipo de Investigación

El presente estudio responde a un trabajo de campo, enmarcado en la modalidad de proyecto factible, definido por (UPEL, 2006) como “la investigación, elaboración y desarrollo de una propuesta de un modelo operativo viable para solucionar un problema” y está apoyado por la investigación de tipo descriptivo documental, ya que “el propósito del investigador es describir situaciones y eventos. Esto es, decir cómo es y se manifiesta determinado fenómeno”, (Hernández Sampieri, 1999), en otras palabras, se describirá los procesos de desarrollo de software actuales para detectar sus debilidades y a partir de éstas proponer un proceso enfocado a las líneas de producción de software, con una aplicación de éste a los desarrollos de software ejecutados por la Coordinación Nacional de Tecnología de Información de la UNEXPO.

Fases del Estudio

Fase I: Diagnóstico

Población y Muestra

Según (Balestrini, 2005), población se define como “cualquier conjunto de elementos de los que se quiere conocer o investigar alguna o algunas de sus características”. En este sentido, la población del presente estudio, estará representada

por los sistemas transaccionales que soportan las actividades operativas de la UNEXPO.

La unidad de investigación donde se aplica el estudio lo constituyen la Coordinación Nacional de Tecnología de Información (CNTI) de la UNEXPO, exenta de toda finalidad de lucro, tiene por objetivos, coordinar, supervisar e implantar proyectos que brinden soluciones integradas, para servicios de información, plataforma de voz, datos y seguridad, a fin de garantizar un soporte tecnológico óptimo a la institución.

Esta coordinación gestiona dos sistemas de información: el sistema administrativo integrado, llamado SAI-UNEXPO, el cual esta compuesto por los módulos de Seguridad, Nomina, Contabilidad, Presupuesto, Compra y Almacén, Tesorería, Fondos en Anticipos y Bienes Nacionales, y el sistema académico, encargado de registrar las inscripciones de estudiantes, actualizaciones de notas, horarios, entre otros.

En este orden de ideas, (Ruiz Bolívar, 2007), define la muestra como "un grupo representativo de la población de la cual fue extraída en cuanto a edad, tamaño, número, sexo, condiciones de crecimiento, etc.". Así que, de los sistemas que gestiona la coordinación, se toma como muestra el sistema Administrativo, SAI-UNEXPO, con especial atención sobre los módulos de Seguridad y Nómina de la UNEXPO para diseñar la línea de producción de software para sistemas transaccionales, aplicando el proceso **I**n**D**o**C**a**S** **E**xpandido.

Diseño de la Investigación

En cuanto al diseño de la propuesta de la investigación, se hizo una revisión bibliográfica para el logro de los objetivos.

En primer lugar, se caracterizó el dominio de los sistemas transaccionales, levantando los requisitos funcionales y no funcionales del área en cuestión, obteniendo así un modelo conceptual del dominio.

En segundo lugar, se estudió el enfoque de líneas de producción de software mostrando su marco conceptual, las actividades y activos de software inherentes al mismo, así como sus procesos fundamentales.

En tercer lugar, se estableció seguir el proceso **I**n**D**o**C**a**S** para diseñar una línea de producción de software con calidad, ejecutándose de este modo el proceso de ingeniería de dominio. Se hizo necesaria la expansión de dicho proceso puesto que hasta ahora sólo contaba con el análisis y el diseño del dominio. Se propuso entonces un conjunto de actividades y artefactos para la disciplina de implementación del dominio del proceso de Ingeniería de Dominio y completar el proceso para el diseño de la línea de producción de software.

Finalmente, se aplicó el proceso **I**n**D**o**C**a**S E**xpandido para obtener la línea de producción para sistemas transaccionales, a la Coordinación Nacional de Tecnología de Información de la UNEXPO, como caso de estudio.

Técnicas e Instrumentos de Recolección de Datos

Según (Lee et al, 1992), “Muchos investigadores dedicados a hacer estudios tanto cuantitativos como cualitativos, sugieren combinar más de un método en una misma investigación”. Para esta investigación, se utilizó el método de revisión bibliográfica, el cual permite conocer conceptos, características y funcionamiento de las mejoras de procesos aplicadas a las organizaciones. También se realizaran consultas en Internet dado que es una fuente actualizada en el área de: líneas de producción de software, planificación estratégica, metodologías de mejoras de procesos y de los modelos que se utilizan en el desarrollo de software. Así mismo se emplearán técnicas de resumen, subrayado, presentación de cuadros, ilustraciones, diagramas, que permitirán de forma teórica apoyar la presente investigación.

Fase II: Estudio de Factibilidad

Factibilidad Técnica

Desde el punto de vista tecnológico, la construcción de una línea de producción para sistemas transaccionales, requiere de recursos de personal calificado, software y hardware.

En cuanto al personal calificado, es necesario dos (2) personas graduadas en carreras como Ingeniero o Licenciado en el área de Sistemas o Informática, con una experiencia de por lo menos dos años en el área de desarrollo de sistemas transaccionales, como por ejemplo Comercio electrónico, Sistemas Administrativos, Sistema Bancarios, entre otros. Adicionalmente, se debe tener una persona con conocimientos sobre arquitectura de software, planificación estratégica y metodologías de mejoras de procesos.

Lo antes planteado es factible en la Coordinación Nacional de Tecnología de Información de la UNEXPO (CNTI), ya que tres (3) de las cinco (5) personas de la coordinación, poseen experiencia en el área de desarrollo de sistemas transaccionales, planificación estratégica y metodologías de mejoras de procesos, y una (1) persona conoce sobre el enfoque de líneas de producción de software.

Con respecto al software requerido para el desarrollo de la investigación, consta principalmente de frameworks que soporten los principios enunciados por el enfoque de líneas de producción de software, así como también lenguajes de programación y herramientas que permitan documentar el modelado de los componentes de software. Entre los frameworks manejados por la coordinación, están los siguientes:

- Para el lenguaje de Programación PHP están: Symfony Php, Cakephp y Kumbiaphp.
- La herramienta de programación de Visual Basic 6.0.

A esto se le debe agregar, herramientas de modelado para UML con licencia gpl como Dia, ArgoUML, Eclipse Uml, entre otros, herramienta ACME para modelado de la notación SPEM y herramientas para realizar documentos de oficinas como libreoffice, para documentar el proceso de desarrollo.

En este orden de ideas, en cuanto al software es perfectamente factible el uso de "licencias libres", decretadas por el Gobierno Nacional Venezolano en el año 2001, ya que no representan inversiones onerosas para el presupuesto de la coordinación. Se sugiere entonces, un Sistema Operativo GNU/GPL como por ejemplo Ubuntu en su versión superior a 7.04 o el sistema operativo Canaima, de licencia libre. Aunque podría instalarse en Sistemas Operativos de carácter propietario por ser un sistema portable a cualquier plataforma (WINDOWS, SOLARIS, entre otros). Lo único que se ejecutaría en Software propietario es la herramienta Visual Basic 6.0 y los productos generados por ellas, como algunos módulos del sistema administrativo integrado.

Para finalizar, se debe asignar una computadora a cada persona de la coordinación, con los siguientes requerimientos mínimos de hardware: Procesador Pentium IV, 2 GB de RAM, 120 GB de disco duro, monitor a color, ratón, teclado, unidad de CD/RW/DVD/RW.

Factibilidad Económica

Por otra parte, se dispone de cinco (5) computadoras Pentium IV con 160gb, 4gb de ram, monitor a color, ratón, teclado, unidad de CD/RW/DVD/RW, con casi todas las herramientas necesarias descritas anteriormente, a excepción de la herramienta ACME, pero esta no tiene costo algunos para su adquisión e instalación. Cada computadora cuesta alrededor de Bs. 7.500,00, al multiplicarlos por las 5 que se necesitan da como resultado la cantidad de Bs. 37.500,00. No obstante, la UNEXPO posee las computadoras cumplen tanto la cantidad como las especificaciones señaladas anteriormente.

En cuanto al personal, el costo por cada Ingeniero o Licenciado en el área de sistema es de Bs. 3.000,00 mensual en el ámbito universitario y en la Coordinación Nacional de Tecnología de Información de la UNEXPO, trabajan 5 personas, el costo total mensual es de Bs. 15.000,00. Este costo del personal de la coordinación es pagada por la UNEXPO.

Para finalizar, el costo de las aplicaciones de software a utilizar es de Bs 0,00, debido a estos se encuentran en la red y no poseen costo en cuanto a la adquisición y licencia.

Factibilidad Social

Referente a la factibilidad social, el uso de nuevos productos de software derivados de una línea de producción de software para sistemas transaccionales, permiten el crecimiento y desarrollo de la sociedad académica, industrial, centros de desarrollo de software, entre otros. Beneficiando a los interesados en el desarrollo de aplicaciones bajo el enfoque de líneas de producción de software, con especial atención sobre los sistemas transaccionales, profesionales del área con roles de arquitectos de software, integradores, programadores e incluso los usuarios finales quienes experimentarán beneficios como tiempos cortos en las versiones y mejoras en la calidad de los sistemas y servicios.

Por lo tanto, debido a los beneficios que acarrea socialmente, la ayuda y disposición social para este tipo de trabajos está garantizada por la motivación actual al avance de los sistemas de información.

De lo expresado anteriormente, se puede concluir entonces que el proyecto tiene garantizado su desarrollo, ya que es factible de forma técnica, económica y social.

CAPÍTULO IV

PROPUESTA DEL ESTUDIO

Justificación

El diseño de una línea de producción de software esta relacionada a la construcción de una plataforma reutilizable y la definición de características comunes y variables de una familia. Esta plataforma nos permitirá definir y construir el conjunto de activos de software de la línea de producción de software, que serán de utilidad para el desarrollo de aplicaciones específicas para los clientes.

Dentro de este marco, la propuesta de una línea de producción de software planteado en el presente trabajo de investigación, esta orientada a los sistemas transaccionales, ya que ayudara a mejorar la gestión del desarrollo y la calidad del software que se realiza. Estos beneficios también impactarán positivamente al negocio y dado que los sistemas están relacionados con la parte operativa de la organización (compra, venta, inventario, entre otros), por lo que las actualizaciones de los productos de software se hará en el tiempo previsto, permitiendo obtener los objetivos planteados y mejoras en los servicios prestados.

Para obtener la propuesta de línea de producción de software para sistemas transaccionales para Coordinación Nacional de Tecnología de Información (CNTI) de la UNEXPO, se deben realizar los procesos de la ingeniería del dominio y la ingeniería de la aplicación. Para ejecutar el primer proceso, se utiliza como base el proceso para la Ingeniería del Dominio basado en Calidad de Software (**I**n**D**o**C**a**S**), sin embargo, el método **I**n**D**o**C**a**S**, solo alcanza las disciplinas del análisis y diseño de la Ingeniería del dominio, por lo consiguiente se hace necesario, el desarrollo de las

actividades y artefactos correspondiente a la disciplina de la implementación del dominio, con el objeto de abarcar las tres disciplinas (análisis, diseño e implementación del dominio) de la ingeniería del dominio. Para completar la línea de producción de software faltaría completar las actividades y artefactos relacionados al proceso de la ingeniería de la aplicación, que luego puede ser efectuado en otra investigación.

Asimismo, representa un aporte significativo en la línea de investigación de Ciencias de la Computación mención Ingeniería de Software puesto que podría incentivar nuevas investigaciones en esta área, facilitando el aprendizaje acerca de sistemas transaccionales, calidad de software y líneas de producción de software, incluso se podría utilizar este modelo como marco de referencia e introducirle las adaptaciones necesarias para el logro de otros objetivos en el área.

Objetivos

Objetivo General

Proponer un diseño de una línea de producción de software para sistemas transaccionales a la Coordinación Nacional Tecnología de Información de la Universidad Nacional Experimental Politécnica “Antonio José de Sucre” (UNEXPO).

Objetivos Específicos

- Estudiar el proceso para la **I**ngeniería del **D**ominio basado en **C**alidad de **S**oftware (**I**n**D**o**C**a**S**) para la obtención del análisis del dominio y diseño del dominio.

- Definir y construir las actividades de la implementación del dominio del proceso para la **I**ngeniería del **D**ominio basado en **C**alidad de **S**oftware (**I**n**D**o**C**a**S**).
- Ejecutar la línea de producción de software resultante al proceso de desarrollo de software de la Coordinación Nacional Tecnología de Información de la Universidad Nacional Experimental Politécnica "Antonio José de Sucre" (UNEXPO).

Descripción de la Propuesta

En la presente investigación, se propone un diseño de una línea de producción de software para sistemas transaccionales. Según (Pohl et al, 2005), el paradigma de ingeniería línea de producción de software está dividida en dos actividades macros, llamados la ingeniería de dominio y la ingeniería de la aplicación, en donde cada uno tiene tres sub disciplinas que son: análisis, diseño e implementación. Por tal motivo, al ser un diseño que estamos presentando solo se implementará la actividad de la ingeniería del dominio, utilizando el proceso **I**n**D**o**C**a**S** (proceso para la Ingeniería del Dominio basado en Calidad de Software) que nos ayudará a ejecutar las disciplinas de análisis del dominio y diseño del dominio.(Ver figura 22)

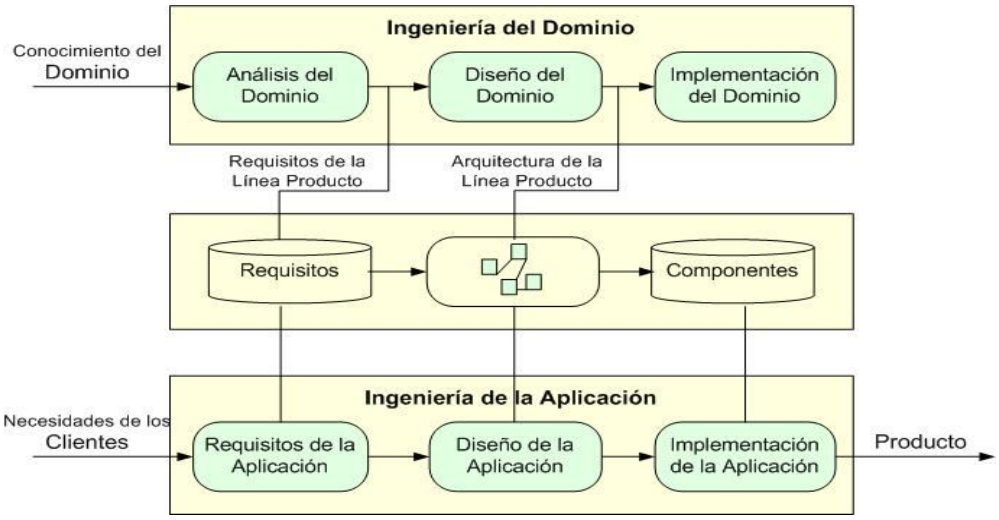


Figura N° 22. Línea de Producción de Software (Pohl et al, 2005)

Con respecto al proceso **I**n**D**o**C**a**S**, al solo cubrir las disciplinas del análisis y diseño del dominio, adoleciendo de la disciplina de la implementación del dominio, por que se hace necesario agregar métodos y técnicas que permitan soportarlas, por lo que se estudia la incorporación del método WATCH – Component, con la finalidad de completar las actividades y los artefactos necesarios para el diseño de la línea de producción de software.

Características del grupo al que va dirigido

Esta investigación está dirigida a la sociedad académica, industrial, centros de desarrollo de software, entre otros, que se interese por el desarrollo de aplicaciones bajo el enfoque de líneas de producción de software, con especial atención a los sistemas transaccionales, profesionales del área con roles de arquitectos de software, integradores, programadores e incluso los usuarios finales.

Estructura de la Propuesta

Para la obtención de la propuesta de una línea de producción de software para sistemas transaccionales se deben cubrir las tres disciplinas que son el análisis, diseño e implementación de la Ingeniería del dominio. Para tal propósito, se utilizara como base el proceso realizado por Canelón (2010), llamado **I**n**D**o**C**a**S**, que abarcara las disciplinas del análisis y diseño del dominio. Seguidamente, se instanciara el método WATCH – Component (Hammar y Montilva, 2003), para incorporar las actividades de *"Especificación del componente"*, *"Aprovisionamiento del componente"*, *"Pruebas del Componente"* y *"Liberación del Componente"* y los artefactos: *"Especificación formal del Componente"*, *"Aceptación de la Plataforma Tecnológica"*, *"Componente Seleccionado"*, *"Componente Probado"*, *"Clasificación del Componente"*, *"Documento de publicación del componente"*, en la Implementación del Dominio.

La combinación del proceso **InDoCaS** con las actividades y artefactos instanciados del método WATCH – Component lo llamaremos **InDoCaS E**xpandido (**InDoCaSE**)(Ver figura 23).

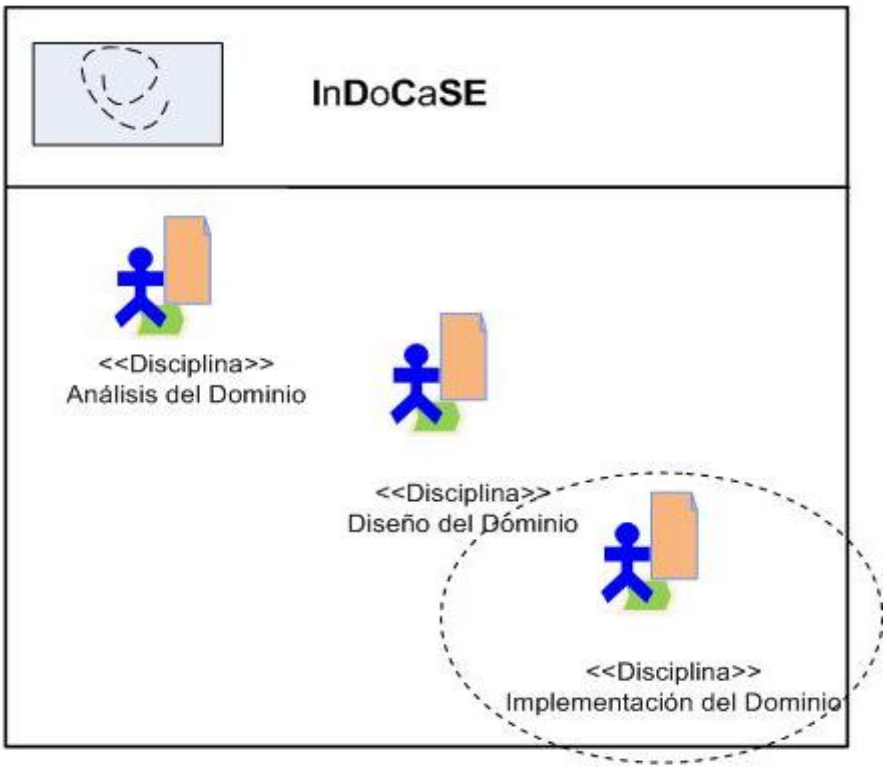


Figura N° 23. InDoCaSE (Autor de la Investigación)

A continuación, en la tabla Nº 5, se presentan las actividades a realizar, utilizando la nomenclatura del método **InDoCaS**:

Tabla N° 5. **InDoCaSE**: Lista de Actividades propuesta para la implementación del dominio

InDoCaS					
ENTRADA	*Modelo conceptual del Dominio*				
DISCIPLINA	**ACTIVIDADES**			**ARTEFACTOS**	
	Nro.	**Tablas**	**Nombre de la actividad**	**Nro.**	**Tablas**
Implementación del Dominio	A_14	8	Especificación del componente	21,22	9,10
	A_15	11	Aprovisionamiento del componente	23	12
	A_16	13	Pruebas del Componente	24	14
	A_17	15	Liberación del Componente	25,26	16,17

(Fuente: Autor de la Investigación)

Definiciones de las actividades y sus artefactos

A continuación, se describirán las actividades utilizadas que provienen del método WATCH – Component creado por (Hammar y Montilva, 2003), con la finalidad de definir las actividades adicionales y cada uno de los artefactos involucrados en la disciplina de la implementación del dominio, ajustándose a la nomenclatura usada por **InDoCaS** para presentar las actividades y artefactos.

Las actividades instanciados a usar del método WATCH – COMPONENT son los siguientes: *"Especificación del componente"*, *"Aprovisionamiento del componente"*, *"Pruebas del componente"* y *"Liberación del componente"* con la finalidad de realizar la disciplina de implementación del dominio y realizar la ingeniería de dominio que es nuestro alcance en esta investigación.

La figura 24 muestra el diagrama de actividades de la disciplina Implementación del Dominio.

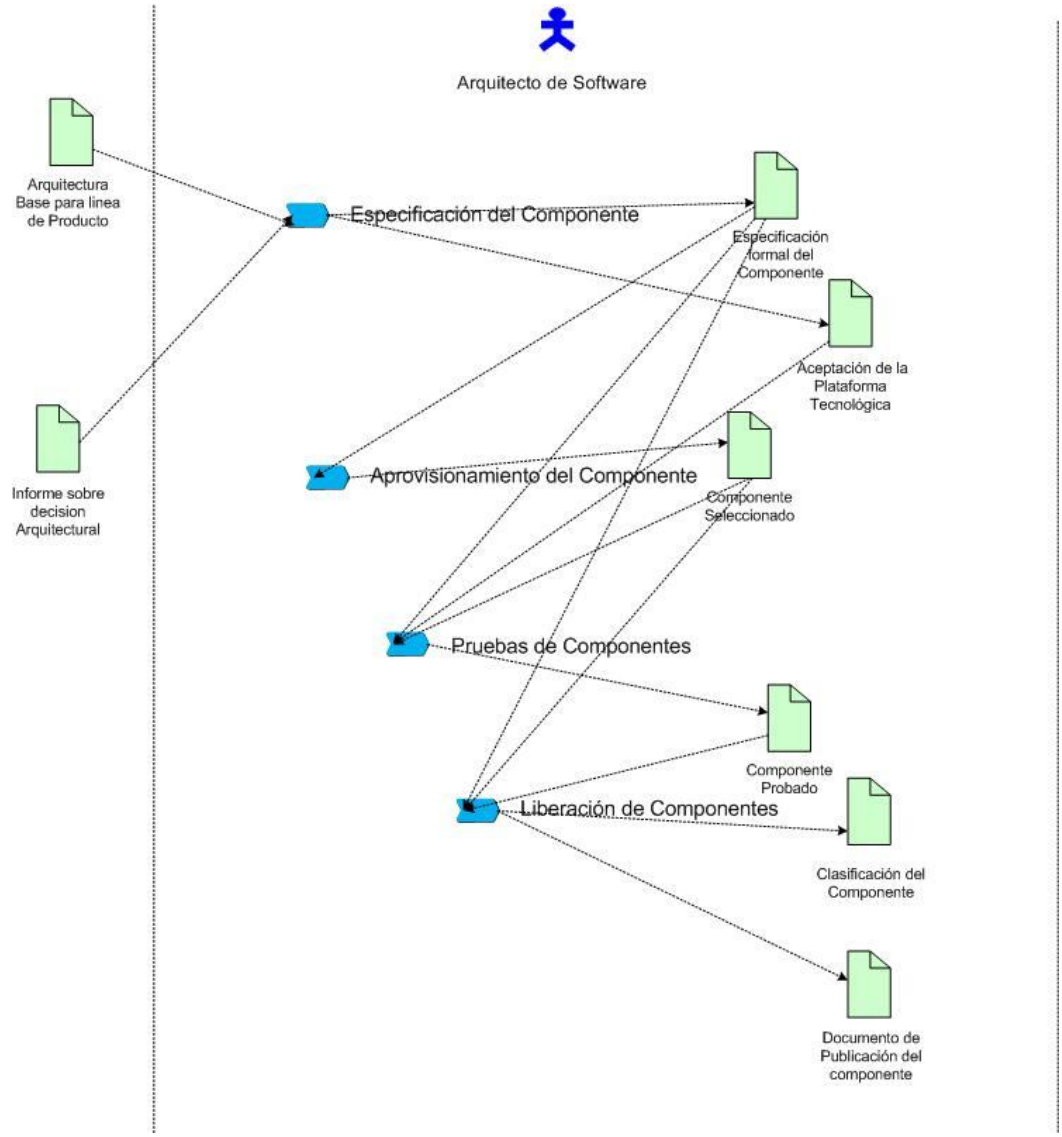


Figura N° 24. InDoCaSE: Diagrama de actividades propuesto para la disciplina de la Implementación del dominio. (Fuente: Autor de la Investigación)

En la tabla Nº 6 presentaremos el resumen de las actividades y artefactos de **InDoCaS E**xpandido, usando la nomenclatura **InDoCaS**.

Tabla N° 6. **InDoCaSE**: Resumen de la lista de actividades propuesta para la ingeniería del dominio

InDoCaS Expandido			
ENTRADA		*Modelo conceptual del Dominio*	
DISCIPLINA		**ACTIVIDADES**	
Análisis del Dominio		**Nro.**	**Nombre de la actividad**
		A_01	Identificación de Requisitos
		A_02	Obtener modelo de similitudes y variabilidad
		A_03	Identificación de propiedades de calidad
		A_04	Obtener modelo de calidad asociado al dominio
		A_05	Creación de escenarios de calidad del dominio
		A_06	Identificar los estilos arquitecturales para el dominio
Diseño del Dominio	**Síntesis**	A_07	Seleccionar los elementos del diseño del dominio que satisfagan el conjunto minimal de requisitos funcionales y no funcionales
		A_08	Escoger patrones arquitecturales candidatos
		A_09	Instanciar elementos arquitecturales para los elementos del diseño del dominio
		A_10	Identificar similitudes entre los elementos arquitecturales instanciados
		A_11	Decidir la selección de la arquitectura como solución arquitectural candidata
	Evaluación	A_12	Validar modelo de calidad del dominio con arquitectura candidatas
		A_13	Escoger arquitectura base para la familia
Implementación del Dominio		**A_14**	**Especificación del componente**
		A_15	**Aprovisionamiento del componente**
		A_16	**Pruebas del Componente**
		A_17	**Liberación del Componente**

(Fuente: Autor de la Investigación)

A continuación, se describen las actividades adicionales y cada uno de los artefactos involucrados en el proceso, ajustada a la nomenclatura que posee **I**n**D**o**C**a**S** (Canelón, 2010) para presentar las actividades y artefactos.

Actividad A_14: Especificación del Componente.

Esta actividad tiene como propósito lograr la especificación del componente que se desea desarrollar, en la misma se describe al componente tanto para los desarrolladores como para los clientes. Utilizaremos los artefactos 19 y 20 denominados *"Arquitectura base para línea de producto"* y *"Informe de decisión arquitectural"* respectivamente, generados en la disciplina del diseño del dominio.

Del artefacto *"Arquitectura base para línea de producto"* se extraerá la información concerniente a las funcionalidades y las interrelaciones del componente con otros componentes, la cuales se utiliza para llenar el contrato de uso y el contrato de realización, propuesta por (Cheesman y Daniels, 2001), que componen el artefacto 21 llamado *"Especificación formal del Componente"*. En el contrato de uso, se señalan las interfaces y las relaciones del componente con otro componente y el contrato de realización, se describen las funcionalidades o métodos que realiza el componente, además se dibujara el diagrama de componente relacionado al componente especificado, usando UML 2.0 y las extensiones que provee.

Al mismo tiempo se utilizará la información necesaria del artefacto *"Informe de decisión arquitectural"* y *"Arquitectura base para línea de producto"*, del primer artefacto se usa las posibles plataforma de desarrollo sugeridos en el informe y del segundo artefacto las funcionalidades que debe realizar el componente, para la tabla de evaluación de plataforma propuesto por (Canelon et al, 2008), que compone el artefacto 22 denominado *"Aceptación de la plataforma tecnológica"*. Este modelo consiste en la construcción de una matriz, en donde en la primera fila se colocar las plataformas de desarrollo sugerido en el informe y en la primera columna las funcionalidades que debe realizar el componente. Dentro de la matriz, se colocan el

puntaje con valores comprendidos entre el 1 al 3, la cual mostramos la descripción en la tabla 7, y la(s) plataforma(s) de mayor puntaje, serán las sugeridas para el desarrollo del componente.

Tabla 7: Escala de Evaluación.

Características	Definición	Puntaje
No Soporta	No apoya el requisito	1
Poco Soporte	De cierta manera se puede satisfacer el requisito o parte de él.	2
Fuerte Soporte	Este requisito aparece explícitamente en la descripción de la plataforma o en su manual.	3

De acuerdo a lo descrito anteriormente, se presenta la continuidad de tablas de actividades y artefactos que vienen del proceso **InDoCaS**:

Tabla 8: Actividad: Especificación del componente.

InDoCaS	
ACTIVIDAD	**DESCRIPCIÓN**
Nombre de la Actividad:	*Especificación del componente*
Responsable:	Diseñador de componentes
Objetivo:	Obtener la especificación del componente que se desea desarrollar.
Nro. Identificación:	A_14
Tipo de documento generado:	Tabla, Documento
Técnica(s) utilizada(s):	UML version 2.0, Método Watch Component
Artefacto(s) de entrada:	Arquitectura base para la línea del producto, Información sobre decisión arquitectural.
Artefacto(s) de salida:	Especificación formal del Componente, Aceptación de la Plataforma Tecnológica artefactos 21 y 22

(Fuente: Autor de la Investigación)

Acorde a lo expresado anteriormente, se presentan en tablas con formato de **InDoCaS**, los dos (2) artefactos derivados de la presente actividad:

El artefacto conocido como *"Especificación formal del Componente"*, contiene funcionalidad esperada por el componente, sus interfaces y operaciones con sus parámetros de entrada, parámetros de salida, restricciones, pre y post condiciones,

todo con ayuda del modelado UML 2.0 y las extensiones que provee. Además, en este documentos se incluyen el contrato de uso y contrato de realización del componente.

En la tabla 9, se presenta el artefacto 21 denominado como *"Especificación formal del Componente"*.

Tabla 9: Artefacto: *"Especificación formal del Componente"*.

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	*Especificación formal del Componente*
Constructor:	Diseñador de componentes
Nro. Identificación ACTIVIDAD:	A_14
Nro. Identificación ARTEFACTO:	21

Formato Asociado:

Contrato de Uso	
Nombre del Componente	Nombre del componente
Interfaces Soportadas	Los nombres de las interfaces que contiene el componente.
Signatura	El nombre y parámetros de una operación, mensaje o evento.
Pre-condición	Los datos de entradas necesarias para que el componente realice las funciones
Post-condición	Las posibles salidas que pueda generar el componente.
Restricciones	Las reglas sobre qué se puede hacer y qué no se puede.
Comportamiento	La descripción de cómo debería proceder el componente
Invariantes	Las invariantes especifica qué será verdadero durante el tiempo en que opere el componente

En el contrato de realización se colocan el nombre, la descripción y las condiciones que deben realizar y cumplir todas las funciones que contenga el componente.

Nro de Identificación de la funcion	**Función Exportada del Componente**	**Pre-condición**	**Post-condición**
El nro del método de componente	Nombre del Método o función que realiza el componente	El valor de los datos que debe entrar a la función	La descripción de la función a ejecutar y los tipos de datos que

			deben salir.
..	..	..	..
n	Función n (Datosentrada): Datosalida	Ej. Not null	Ej. Arreglo1 de tipo string

(Fuente: Autor de la Investigación)

Para el artefacto 22, *"Aceptación de la Plataforma Tecnológica"* se deben especificar la plataforma donde puede desarrollarse el componente. El resumen del presente artefacto se muestra en la tabla 10.

Tabla 10: Artefacto: *"Aceptación de la Plataforma tecnológica"*.

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	*Aceptación de la Plataforma Tecnológica*
Constructor:	Experiencia del Diseñador de componentes
Nro. Identificación ACTIVIDAD:	A_14
Nro. Identificación ARTEFACTO:	22

Formato asociado: se presentan el cuadro de evaluación para extraer la plataforma que más se ajusta a las funcionalidades de los componentes.

Criterios de evaluación (Funcionalidades del Componente)	**Plataforma 1**	**..**	**Plataforma n**
..	..	..	..
Total de Puntaje			

(Fuente: Autor de la Investigación)

Actividad A_15: Aprovisionamiento del Componente.

En esta actividad, de acuerdo a la información aportada por los artefactos 21 y 22 llamados "*Especificación formal del Componente*" y "*Aceptación de la Plataforma tecnológica*" respectivamente, generados en la actividad de "*Especificación del Componente"*, se buscara el componente en los repositorios externos, internos o libres. Luego se seleccionara el tipo de aprovisionamiento a saber son: Adquirir el componente, Suscribir el componente, Adaptar el componente, Envolver el componente y Desarrollar un nuevo componente. Definimos a continuación cada uno de ellos, según (Hammar y Montilva, 2003):

- Adquirir el componente: los componentes pueden ser adquiridos de varias fuentes como repositorios internos, repositorios externos, vendidos por otra empresa y de fuentes abiertas. Si se escoge esta vía la selección del componente debe hacerse de acuerdo a las especificaciones de la arquitectura, el vendedor, las veces que ha sido reutilizadas, plataforma de desarrollo, etc. Esta selección puede arrojar como resultado la selección de componentes que necesiten ser adaptados y luego integrados en la aplicación o que puedan ser inmediatamente integrados.

- Suscribir el componente: se refiere a la funcionalidad o servicio de un componente que es ejecutada por una tercera parte. La interface es especificada por un desarrollador y/o empresa que se dedique a fabricar componente. Para esto es necesario que las interfaces se encuentren bien especificadas en el contrato de especificación, ya que ayudara a su verificación.

- Adaptar el componente: se hace generalmente con el uso de patrones, ya que estos definen para qué pueden ser utilizados y como utilizarlos. El termino patrón, que es adoptado por los diseñadores de software, nos describen abstracciones de software utilizadas por desarrolladores y programadores, es decir, representan soluciones a problemas comunes en un contexto particular.

- Desarrollar un nuevo componente: cuando el componente no se encuentra, no cumple con los requisitos de calidad, es muy costoso de adquirir, el proveedor no

cumple con las exigencias, es necesario implementar el componente. Esto generalmente se hace dentro de la misma empresa con el fin de utilizarlo en el proyecto y de reutilizarlo a futuro.

Después se escogerá el componente o se desarrollará para que cumpla con la información de los contratos, para luego documentar las tareas del aprovisionamiento del componente.

El resumen de esta actividad, siguiendo con el estándar del proceso **InDoCaS**, se presenta en la tabla 11.

Tabla 11: Actividad: Aprovisionamiento del componente.

InDoCaS	
ACTIVIDAD	**DESCRIPCIÓN**
Nombre de la Actividad:	*Aprovisionamiento del componente*
Responsable:	Diseñador de componentes
Objetivo:	Describir el proceso de aprovisionamiento de un componente previamente especificado
Nro. Identificación:	A_15
Tipo de documento generado:	Documento. Diagrama de Componente
Técnica(s) utilizada(s):	UML versión 2.0, Método Watch Component,
Artefacto(s) de entrada:	Especificación formal del Componente y Aceptación de la Plataforma tecnológica
Artefacto(s) de salida:	Componente Seleccionado artefacto 23

(Fuente: Autor de la Investigación)

A continuación se presenta el artefacto de la actividad aprovisionamiento del componente.

En la tabla 12, se muestra el artefacto 23 llamado *"Componente seleccionado"*, este documento debe contener todo lo relacionado al aprovisionamiento del componente, y variara, dependiendo de la forma del aprovisionamiento, si se desarrolla uno nuevo por ejemplo, debe estar incluido el diseño del componente.

Tabla 12: Artefacto: Componente Seleccionado.

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	*Componente Seleccionado*
Constructor:	Diseñador de componentes
Nro. Identificación ACTIVIDAD:	A_15
Nro. Identificación ARTEFACTO:	23
Formato Asociado: se muestra el tipo de aprovisionamiento seleccionado, la relación con la plataforma tecnológica seleccionada previamente y las funcionalidades del componente. Estas funcionalidades se muestran a través de un diagrama de componentes.	

(Fuente: Autor de la Investigación)

Actividad A_16: Pruebas del Componente.

El proceso de pruebas corresponde al proceso de encontrar las diferencias en el comportamiento de los componentes de software con respecto a la manera que se espera que estos se comporten, y verificar además que todos los requerimientos han sido implementados correctamente. Si se encontraran errores en la implementación, estos deben ser corregidos, antes de que el componente sea liberado.

Se revisan las funcionalidades del componentes que provienen del artefacto *"Especificación formal del Componente"* y *"Aceptación de la Plataforma Tecnológicas"* con las funciones registradas en el artefacto *"Componente Seleccionado*", utilizando parcialmente la técnica de certificación del componente usado en WATCH-Component, con el fin de verificar que: el componente sea consistente en su comportamiento, las dependencias del componente estén bien especificadas y la configuración del componente puede ser realizada según lo indicado en la documentación. Con la técnica de certificación, la información de las funcionalidades se le realiza cuatro (4) tareas de verificación llamadas: verificación de consistencias, verificación de dependencias, verificación de configuración y

compatibilidad entre las versiones. Se describe a continuación, cada una de las verificaciones:

- Verificación de Consistencia. Se compara los contratos de uso y contrato de realización provenientes del artefacto *"Especificación formal del Componente"* con la información proveniente del artefacto *"Componente Seleccionado"*.

- Verificación de dependencias. Se compara los diagrama de componentes del artefacto *"Especificación formal del Componente"* con el diagrama derivado del artefacto *"Componente Seleccionado"*.

- Verificación de configuración. Se compara los datos provenientes del artefacto *"Especificación formal del Componente"* con la información derivada del artefacto *"Aceptación de la Plataforma Tecnológicas"*

- Compatibilidad entre las versiones. Se compara, en caso de existir la versión anterior del componente, la información del artefacto *"Componente Seleccionado"* con la versión del componente que se posea en el repositorio.

Seguidamente, se diseña el plan de pruebas del componente con las pruebas a realizar y los resultados esperados, usando la información de los contratos de uso y realización del artefacto "*Especificación formal del Componente*", utilizando la técnica de planificación de pruebas realizada por (Hammar y Montilva, 2003) en WATCH-Component, en donde se agrupa en tres tipos de pruebas, llamadas pruebas funcionales, pruebas de comportamiento y pruebas de aceptación.

El cuadro de verificación del componente y plan de pruebas mencionados anteriormente, constituye el *"Componente Probado"*.

A continuación, manteniendo la nomenclatura del proceso **I**n**D**o**C**a**S**, se presenta el resumen de la actividad:

Tabla 13: Actividad: Prueba del componente

InDoCaS	
ACTIVIDAD	**DESCRIPCIÓN**
Nombre de la Actividad:	*Prueba del componente*
Responsable:	Diseñador de componentes
Objetivo:	Verificar que todos los requerimientos han sido implementados correctamente.
Nro. Identificación:	A_16
Tipo de documento generado:	Documento. Diagrama de Componente
Técnica(s) utilizada(s):	UML versión 2.0, Método Watch Component
Artefacto(s) de entrada:	Especificación formal del Componente, Componente Seleccionado, Aceptación de la Plataforma Tecnológicas
Artefacto(s) de salida:	Componente Probado 24

(Fuente: Autor de la Investigación)

En consecuencia, se muestran a continuación el artefacto 24 generado en la actividad Prueba del componente, denominado *"Componente Probado"*, donde se describe las funcionalidades y características, ya verificado, asociado con el plan de pruebas a realizar a la hora de desarrollado del componente y el diseño en UML del componente.

Tabla 14: Artefacto: Componente Probado.

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	*Componente Probado*
Constructor:	Diseñador de componentes
Nro. Identificación ACTIVIDAD:	A_16
Nro. Identificación ARTEFACTO:	24

Formato Asociado: se muestra características y las funcionalidades del componente. Diagrama de Componentes y el diseño del plan de prueba.

Verificación de Componente

Caso de Pruebas	Detalles	Cumple (Si o No)
Verificar Consistencia		
Verificar dependencias		

Verificar configuración		
Compatibilidad entre las versiones		

El plan de Prueba a realizar el componente es el siguiente:

Pruebas funcionales

Caso de Pruebas	Pruebas a realizar	Resultados Esperado
Prueba de las Funciones	Ej.Entrada por teclado Verificación de Usuario Valido Verificación de Usuario No Valido	Letra escrita por pantalla Entrada del usuario al sistema Mensaje de Acceso denegado
Prueba de interfaces	Verificación de Usuario Valido	

Pruebas De Comportamiento

Caso de Pruebas	Pruebas a realizar	Resultados Esperado
Requerimientos no funcionales	Ej. Usuarios y clave vacía	Mensaje de Acceso denegado
Prueba Confiabilidad	Verificación de Usuario Valido Verificación de Usuario No Valido	Generación de Identificación de la sesión Entrada del usuario al sistema Mensaje de Acceso denegado
Pruebas de configuración	Pruebas de ejecución en plataforma Linux y Windows	El mismo comportamiento en los dos sistemas
Pruebas de Recuperación	Como el componente no guarda datos, no debe haber la posibilidad de recuperar datos.	Ninguno en especifico

Pruebas de Aceptación

Caso de Pruebas	Pruebas a realizar	Resultados Esperado
Prueba de Aceptación por el grupos	Ej. Puesta a prueba en conjunción con los componentes con los cuales debe operar.	Aceptación del Componente

(Fuente: Autor de la Investigación)

Actividad A_17: Liberación del Componente.

En este momento, después que el componente ha sido probado, se utiliza las informaciones provenientes de los artefactos: *"Especificación formal del componente"*, *"Componente Seleccionado"* y *"Componente Probado"*, usando el método Reusable Asset Specification (RAS), para la generar el artefacto 25 *"Clasificación del Componente"*. Este método RAS, secciona (4) cuatro grupos mayores, llamados: sección Clasificación, sección Soluciones, sección de Usos y sección de Relaciones con los Componentes. A continuación se muestra la descripción de cada una de las secciones:

- Clasificación: se presenta una descripción del contexto donde es el activo el componente.
- Soluciones: señala y describe los artefactos (requisitos, diagramas, código, entre otros), que posee el componente.
- Usos: contiene las normas para instalar, personalizar y utilizar el componente.
- Relaciones con los Componentes: describe la relación de este componente con otros componentes.

Luego se procede a la publicación de la especificación del componente, conjuntamente con los resultados de las pruebas del componente una vez utilizados por los usuarios externos. Para obtener el artefacto 26 *"Documento de publicación del Componente"*, se utilizar el método RAS para la publicación de componente, en donde se selecciona la forma de cómo se va a publicar la información relacionada al artefacto 25 *"Clasificacion del Componente"*. A continuación, se muestra los tipos de publicación del componente:

- Paquete incluido todo como un archivo.(Ej. Archivos.zip)
- Paquete dispersado en varios archivos, se tiene dos sub-divisiones.

- Los artefactos están en una sola ubicación física.
- Los artefactos están distribuidos en diferentes ubicaciones.

En la tabla 15, que se muestra a continuación, se resumen la definición de la actividad.

Tabla 15: Actividad: Liberación del Componente.

InDoCaS	
ACTIVIDAD	**DESCRIPCIÓN**
Nombre de la Actividad:	*Liberación del componente*
Responsable:	Diseñador de componentes
Objetivo:	Publicar el componente para se colocado en un repositorio de componentes y estar disponible al desarrollador.
Nro. Identificación:	A_17
Tipo de documento generado:	Documentos
Técnica(s) utilizada(s):	Método Watch Component, Reusable Asset Specification versión 2.2(RAS)
Artefacto(s) de entrada:	Especificación formal del componente, Componente Seleccionado, Componente Probado.
Artefacto(s) de salida:	Clasificación del componente y Documento de publicación del componente artefacto 25 y 26

(Fuente: Autor de la Investigación)

Los artefactos generados por esta actividad son los siguientes:

Tabla 16: Artefacto: Clasificación del componente.

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	*Clasificación del componente*
Constructor:	Diseñador de componentes
Nro. Identificación ACTIVIDAD:	A_17
Nro. Identificación ARTEFACTO:	25
Formato Asociado En este documento se almacenan los datos correspondientes a la clasificación del componente, el acceso que va a tener el componente, la especificación de cada una de sus formas. Se muestra a continuación las secciones a llenar. <u>Sección Clasificación</u>	

Sección Soluciones Sección de Usos Sección de Relaciones con los Componentes

(Fuente: Autor de la Investigación)

Tabla 17: Artefacto: Documento de Publicación del componente.

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	*Documento de publicación del componente*
Constructor:	Diseñador de componentes
Nro. Identificación ACTIVIDAD:	A_17
Nro. Identificación ARTEFACTO:	26
Formato Asociado En este documento corresponde a las especificaciones de las formas de componente, los resultados ante de la publicación y la información de los resultados obtenidos en la utilización que se obtenga después de su publicación.	

(Fuente: Autor de la Investigación)

Caso de Estudio: Aplicación del Proceso InDoCaSE al Dominio de Sistemas Transaccionales para Coordinación Nacional de Tecnología de Información.

A continuación se presenta la ejecución del proceso **InDoCaS E**xpandido con el objeto de obtener la propuesta de la línea de producción de software para la Coordinación Nacional de Tecnología de Información (CNTI) de la UNEXPO, la cual, es una coordinación adscrita a la Oficina Central de Tecnología y Servicios de Información (OCTSI) y tiene por objetivos, coordinar, supervisar e implantar proyectos que brinden soluciones integradas, para servicios de información, plataforma de voz, datos y seguridad, a fin de garantizar el soporte tecnológico óptimo a la institución. (Resolución C.U. Nº 2005-E09-05). En esta coordinación se ejecutan proyectos que buscan integrar las diferentes transacciones originados por los procesos administrativos y académicos con el fin de mejorar los servicios que se

prestan a los usuarios. Adicionalmente, se mantienen los sistemas que ayudan a manejar a la UNEXPO con sus operaciones cotidianas tanto en la parte administrativa, (Seguridad, Nomina, Contabilidad, Presupuesto, Compra y Almacén, Tesorería, Fondos en Anticipos, Bienes Nacionales), como en la parte académica, (módulos de inscripciones de estudiantes, actualizaciones de notas, entre otros).

Por tal motivo, usaremos la Coordinación Nacional de Tecnología de Información y los procesos administrativos de nómina de la UNEXPO para obtener una línea de producción de software para sistemas transaccionales, aplicando el proceso **I**n**D**o**C**a**SE**.

Del Proyecto del Sistema Administrativo Integrado de la UNEXPO (SAI-UNEXPO), se presentan a continuación los requisitos levantados para el desarrollo del sistema de nóminas:

- Manejo de nóminas por vicerrectorados y núcleos.
    - Generar reportes de listado por vicerrectorados y/o núcleos.
    - Generar reportes nóminas por vicerrectorados y/o núcleos.
- Gestionar de cargos de la institución con sus dedicaciones y remuneraciones ordinarias y prestaciones sociales asociadas.
    - Generar reportes de remuneración ordinaria por cargo.
    - Generar listado de conceptos de las remuneraciones.
    - Gestionar Cargos y el tipo de dedicación.
- Gestionar la información relacionados a los empleados como datos personales, cargos y escalas dentro de la institución, cuenta de banco para el pago de la nómina, prestaciones sociales generadas y el historial de cargos.
    - Generar reportes sobre el empleado y sus datos básicos.

- Listado de funcionarios asignados a fondos de pensionados y jubilados.
- Consulta de Recibos de pagos.
- Gestionar Cargos y el tipo de dedicación de los funcionarios.
- Listado de funcionarios asignados por bancos.

A continuación se aplicara el proceso **InDoCaSE** al dominio de los sistemas transaccionales del CNTI, en la tabla 18 se muestra el grupo de actividades que se realizaran en las disciplinas del análisis, diseño e implementación del dominio. En la especificación se utiliza la nomenclatura establecido en el proceso.

Tabla 18. Actividades del modelo de procesos **InDoCaSE** aplicados al dominio

	DOMINIO	**SISTEMAS TRANSACCIONALES**	
Disciplina	**Actividad**	**Nombre de la actividad**	**Tablas Resultantes**
Análisis del Dominio	A_01	Identificación de requisitos	19, 20
	A_02	Obtener modelo de similitudes y variabilidad	21,22,23
	A_03	Identificación de propiedades de Calidad	24,25
	A_04	Obtener modelo de calidad asociado al dominio	26
	A_05	Creación de escenarios de calidad del dominio	27
	A_06	Identificar los estilos arquitecturales para el dominio	28
Diseño del Dominio	A_07	Seleccionar elementos del diseño del dominio que satisfagan el conjunto minimal de requisitos funcionales y no funcionales	29
	A_08	Escoger patrones arquitecturales candidatos	30
	A_09	Instanciar elementos arquitecturales para los elementos del diseño del dominio	31
	A_10	Identificar similitudes entre los elementos arquitecturales instanciados	32
	A_11	Decidir la selección de la arquitectura como solución arquitectural candidata	33,34
	A_12	Validar modelo de calidad del dominio con arquitecturas candidatas	35,36
	A_13	Escoger arquitectura base para la familia	37,38

Implementación del dominio	A_14	Especificación del componente	39,40
	A_15	Aprovisionamiento del Componente	41
	A_16	Pruebas del Componente	42
	A_17	Liberación de Componente	43,44

(Fuente: Autor de la Investigación)

Actividades del Análisis del Dominio

Actividad 1: Identificación de Requisitos.

En la presente actividad se desarrollan dos artefactos que son: listas de requisitos funcionales y no funcionales del dominio, en la tabla 19, se muestran los requisitos funcionales de los sistemas transaccionales en la Coordinación Nacional de Tecnología de Información.

Tabla 19: 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	Manejo de nóminas por vicerrectorados y núcleos.
2	Generar reportes de listado por vicerrectorados y/o núcleos.
3	Generar reportes nóminas por vicerrectorados y/o núcleos.
4	Gestionar los cargos de la institución con sus dedicaciones, remuneraciones ordinarias y prestaciones sociales asociadas.
5	Generar reportes de remuneración ordinaria por cargo.
6	Generar listado de conceptos de las remuneraciones.
7	Gestionar Cargos y el tipo de dedicación.
8	Gestionar la información relacionados a los empleados como datos personales, cargo y escalas dentro la institución, cuenta de banco para el pago de la nómina, prestaciones sociales generadas y el historial de cargo.
9	Generar reportes sobre el empleado y sus datos básicos.

10	Listado de funcionarios asignados a fondos de pensionados y jubilados.
11	Consulta de Recibos de pagos.
12	Asignar Cargos y el tipo de dedicación de los funcionarios.
13	Listado de funcionarios asignados por bancos.

(Fuente: Autor de la Investigación)

Tabla 20. 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

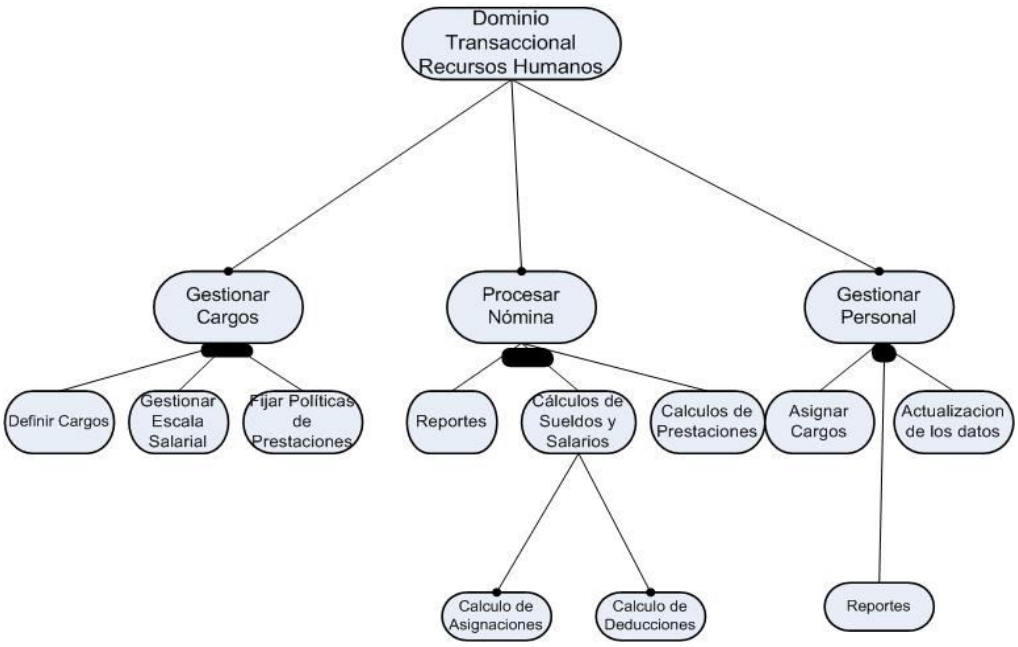
Nro. Identificación	Reglas del Negocio asociadas al dominio	Requisitos no funcionales derivados de las reglas del negocio
	Políticas	
1	Los servicios funcionan en una Intranet.	-Debe funcionar en maquinas con sistemas operativo Canaima. -Cumplimientos de estándares y normativas de la institución con el fin de garantizar que el sistema se ejecute.
	Procesamiento	
2	Integridad de la información	- La transacción se debe se guardada completa y correctamente en la Base de datos y cumpliendo con las reglas de negocio de la institución. - El sistema debe garantizar que una vez que transacción se confirma, su efecto no se pierde en la base de datos, a pesar de fallos posteriores.
3	El servicio debe ser garantizado en la intranet	-Hacer posible el acceso a los servicios, lo cual implica, ancho de banda apropiado y garantizar las conexiones, teniendo que solventar problema de redes. -Tiempo de transmisión apropiado, tiempo de repuesta adecuados dentro de rango establecido.
4	Solución centralizada con respecto a los datos	- Acceso centralizado a los datos mediante de interfaces conectadas a la base de datos.
	Implementación	
5	Fiabilidad	-El sistema en caso de fallas, debe tener mecanismos de recuperación y de respaldo de datos.
6	Restricción de miembros del grupo	- Se permite el acceso únicamente a personal autorizado por la Institución.

(Fuente: Autor de la Investigación)

Actividad 2: Obtener modelo de similitudes y variabilidad.

En esta actividad, se define el conjunto de requisitos mínimos y obligatorios en los sistemas transaccionales realizados por la Coordinación Nacional de Tecnología de Información de la UNEXPO, usando FODA, consiguiendo tres artefactos: conjunto de características, conjunto de puntos de variación y conjunto minimal de requisitos funcionales y no funcionales. El modelo de características se presenta a continuación en la tabla 21.

Tabla 21. Conjunto de características

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

(Fuente: Autor de la Investigación)

A continuación se obtienen los puntos de variación, los cuales describen dónde existen diferencias en las aplicaciones, expresan la variabilidad en las características y se presenta en la tabla 22.

Tabla 22. 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
Gestionar Escala Salarial	Esta característica depende en gran medida al tabulador universitario que exige la OPSU. En las empresas comerciales se rigen por la ley Orgánica del Trabajo.
Fijar Política de Prestaciones	Esta característica depende en gran medida a la formula que exige la OPSU. En las empresas comerciales se rigen por la ley Orgánica del Trabajo.

(Fuente: Autor de la Investigación)

Seguidamente, se presenta el conjunto de requisitos mínimos y obligatorios de la familia de sistemas transaccionales para el SAI-UNEXPO, descrito por el artefacto "*conjunto minimal de requisitos funcionales y no funcionales*" que se muestra en la tabla 23.

Tabla 23. 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	Manejo de nóminas por vicerrectorados y núcleos.
2	Generar reportes de listado por vicerrectorados y/o núcleos.
3	Generar reportes nóminas por vicerrectorados y/o núcleos.
4	Gestionar los cargos de la institución con sus dedicaciones, remuneraciones ordinarias y prestaciones sociales asociadas.
5	Generar reportes de remuneración ordinaria por cargo.
6	Generar listado de conceptos de las remuneraciones.
7	Gestionar Cargos y el tipo de dedicación.
8	Gestionar la información relacionados a los empleados como datos personales, cargo y escalas dentro la institución, cuenta de banco para el

	pago de la nómina, prestaciones sociales generadas y el historial de cargo.
9	Generar reportes sobre el empleado y sus datos básicos.
10	Listado de funcionarios asignados a fondos de pensionados y jubilados.
11	Consulta de Recibos de pagos.
12	Asignar Cargos y el tipo de dedicación.
13	Listado de funcionarios asignados por bancos.

Nro. Identificación	Descripción del requisito no funcional
1	-Cumplimientos de estándares y normativas de la institución con el fin de garantizar que el sistema se ejecute.
2	- La transacción se debe se guardada completa y correctamente en la Base de datos y cumpliendo con las reglas de negocio de la institución.
3	- El sistema debe garantizar que una vez que transacción se confirma, su efecto no se pierde en la base de datos, a pesar de fallos posteriores.
4	-Hacer posible el acceso a los servicios, lo cual implica, ancho de banda apropiado y garantizar las conexiones, teniendo que solventar problema de redes.
5	-Tiempo de transmisión apropiado, tiempo de repuesta adecuados dentro de rango establecido.
6	- Acceso centralizado a los datos mediante de interfaces conectadas a la base de datos.
7	-El sistema en caso de fallas, debe tener mecanismos de recuperación y de respaldo de datos.
8	- Se permite el acceso únicamente a personal autorizado por la Institución.

(Fuente: Autor de la Investigación)

Actividad 3: Identificación de propiedades de Calidad.

La actividad identificación de propiedades de calidad permite identificar las propiedades de calidad asociadas al artefacto conjunto minimal de requisitos funcionales y no funcionales presentados en la tabla 19, ajustadas a las definiciones establecidas en la terminología estándar ISO/IEC 2010, en esta actividad se obtienen dos artefactos: lista de requisitos funcionales y no funcionales con sus propiedades de calidad asociadas. En las tablas 24 y 25, se muestran las listas de requisitos funcionales y no funcionales con sus propiedades de calidad asociadas, incorporando las propiedades y sus atributos de calidad.

Tabla 24. 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

Nro. Identificación	**Descripción del requisito funcional**	**Características de calidad ISO/IEC 25010**	**Atributos (Características)**
1	Manejo de nóminas por vicerrectorados y núcleos.	**Confiabilidad** -Disponibilidad	-Atributo: tiempo de espera. Métrica: un número en el rango [1..5]
		Eficiencia -Comportamiento en el tiempo	-Atributo: Ejecución con respecto a la utilización de recurso: memoria, ancho de banda,etc.
		Funcionalidad -Preciso	-Atributo: tiempo de completitud de tareas. Métrica: un número en el rango [1..10]
		Seguridad -Integridad -Autenticidad	-Presencia de mecanismo: Un mecanismo debe ser provisto. Métrica: Booleano -Protección para el origen de datos y Protección para autentificación de acceso. Métrica: un número en el rango [0..1]
2	Generar reportes de listado por vicerrectorados y/o núcleos.	**Confiabilidad** -Disponibilidad	-Atributo: tiempo de espera. Métrica: un número en el rango [1..5]
		Eficiencia -Comportamiento en el tiempo	-Atributo: Ejecución con respecto a la utilización de recurso: memoria, ancho de banda,etc.

		Funcionalidad -Preciso	-Atributo: tiempo de completitud de tareas. Métrica: un número en el rango [1..10]
		Usabilidad -Operabilidad	-Proporción de las funcionalidades que son entendidas correctamente por el usuario.
3	Generar reportes nóminas por vicerrectorados y/o núcleos.	**Confiabilidad** -Disponibilidad	-Atributo: tiempo de espera. Métrica: un número en el rango [1..5]
		Eficiencia -Comportamiento en el tiempo	-Atributo: Ejecución con respecto a la utilización de recurso: memoria, ancho de banda,etc.
		Funcionalidad -Preciso	-Atributo: tiempo de completitud de tareas. Métrica: un número en el rango [1..10]
		Usabilidad -Operabilidad	-Proporción de las funcionalidades que son entendidas correctamente por el usuario.
		Seguridad -Integridad -Autenticidad	-Presencia de mecanismo: Un mecanismo debe ser provisto. Métrica: Booleano -Protección para el origen de datos y Protección para autentificación de acceso. Métrica: un número en el rango [0..1]
4	Gestionar los cargos de la institución con sus dedicaciones, remuneraciones ordinarias y prestaciones sociales asociadas.	**Usabilidad** -Reconocido como adecuado -Operabilidad	-Proporción de las funcionalidades que son entendidas correctamente por el usuario.
		Funcionalidad -Preciso	-Atributo: tiempo de completitud de tareas. Métrica: un número en el rango [1..10]
		Seguridad -Integridad	-Presencia de mecanismo: Un mecanismo debe ser

		-Autenticidad	provisto. Métrica: Booleano -Protección para el origen de datos y Protección para autentificación de acceso. Métrica: un número en el rango [0..1]
5	Generar reportes de remuneración ordinaria por cargo.	**Confiabilidad** -Disponibilidad **Eficiencia** -Comportamiento en el tiempo **Funcionalidad** -Preciso **Usabilidad** -Operabilidad	-Atributo: tiempo de espera. Métrica: un número en el rango [1..5] -Atributo: Ejecución con respecto a la utilización de recurso: memoria, ancho de banda,etc. -Atributo: tiempo de completitud de tareas. Métrica: un número en el rango [1..10] -Proporción de las funcionalidades que son entendidas correctamente por el usuario.
6	Generar listado de conceptos de las remuneraciones.	**Confiabilidad** -Disponibilidad **Eficiencia** -Comportamiento en el tiempo **Funcionalidad** -Preciso **Usabilidad** -Operabilidad	-Atributo: tiempo de espera. Métrica: un número en el rango [1..5] -Atributo: Ejecución con respecto a la utilización de recurso: memoria, ancho de banda,etc. -Atributo: tiempo de completitud de tareas. Métrica: un número en el rango [1..10] -Proporción de las funcionalidades que son entendidas correctamente por el usuario.
7	Gestionar Cargos y el tipo de dedicación.	**Usabilidad** -Reconocido como adecuado	-Proporción de las funcionalidades que son entendidas correctamente

		-Operabilidad **Seguridad** -Integridad -Autenticidad -Confidencialidad	por el usuario. -Presencia de mecanismo: Un mecanismo debe ser provisto. Métrica: Booleano -Protección para el origen de datos y Protección para autentificación de acceso. Métrica: un número en el rango [0..1]
8	Gestionar la información relacionados a los empleados como datos personales, cargo y escalas dentro la institución, cuenta de banco para el pago de la nómina, prestaciones sociales generadas y el historial de cargo.	**Usabilidad** -Reconocido como adecuado -Operabilidad **Seguridad** -Integridad -Autenticidad -Confidencialidad	-Proporción de las funcionalidades que son entendidas correctamente por el usuario. -Presencia de mecanismo: Un mecanismo debe ser provisto. Métrica: Booleano -Protección para el origen de datos y Protección para autentificación de acceso. Métrica: un número en el rango [0..1]
9	Generar reportes sobre el empleado y sus datos básicos.	**Usabilidad** -Reconocido como adecuado -Operabilidad **Funcionalidad** -Preciso **Seguridad** -Integridad -Autenticidad	-Proporción de las funcionalidades que son entendidas correctamente por el usuario. -Atributo: tiempo de completitud de tareas. Métrica: un número en el rango [1..10] -Presencia de mecanismo: Un mecanismo debe ser provisto. Métrica: Booleano -Protección para el origen de datos y Protección para autentificación de acceso. Métrica: un número en el rango [0..1]
10	Listado de funcionarios asignados a fondos de pensionados y jubilados.	**Usabilidad** -Reconocido como adecuado -Operabilidad **Funcionalidad** -Preciso	-Proporción de las funcionalidades que son entendidas correctamente por el usuario. -Atributo: tiempo de completitud de tareas. Métrica: un número en el

			rango [1..10]
		Seguridad -Integridad -Autenticidad	-Presencia de mecanismo: Un mecanismo debe ser provisto. Métrica: Booleano -Protección para el origen de datos y Protección para autentificación de acceso. Métrica: un número en el rango [0..1]
11	Consulta de Recibos de pagos.	**Confiabilidad** -Disponibilidad	-Atributo: tiempo de espera. Métrica: un número en el rango [1..5]
		Eficiencia -Comportamiento en el tiempo	-Atributo: Ejecución con respecto a la utilización de recurso: memoria, ancho de banda,etc.
		Funcionalidad -Preciso	-Atributo: tiempo de completitud de tareas. Métrica: un número en el rango [1..10]
		Usabilidad -Operabilidad	-Proporción de las funcionalidades que son entendidas correctamente por el usuario.
		Seguridad -Integridad -Autenticidad	-Presencia de mecanismo: Un mecanismo debe ser provisto. Métrica: Booleano -Protección para el origen de datos y Protección para autentificación de acceso. Métrica: un número en el rango [0..1]
12	Asignar Cargos y el tipo de dedicación.	**Usabilidad** -Reconocido como adecuado -Operabilidad	-Proporción de las funcionalidades que son entendidas correctamente por el usuario.
		Seguridad -Integridad -Autenticidad -Confidencialidad	-Presencia de mecanismo: Un mecanismo debe ser provisto. Métrica: Booleano -Protección para el origen de datos y Protección para autentificación de acceso. Métrica: un número en el

			rango [0..1]
13	Listado de funcionarios asignados por bancos.	**Confiabilidad** -Disponibilidad	-Atributo: tiempo de espera. Métrica: un número en el rango [1..5]
		Eficiencia -Comportamiento en el tiempo	-Atributo: Ejecución con respecto a la utilización de recurso: memoria, ancho de banda,etc.
		Funcionalidad -Preciso	-Atributo: tiempo de completitud de tareas. Métrica: un número en el rango [1..10]
		Usabilidad -Operabilidad	-Proporción de las funcionalidades que son entendidas correctamente por el usuario.
		Seguridad -Integridad -Autenticidad	-Presencia de mecanismo: Un mecanismo debe ser provisto. Métrica: Booleano -Protección para el origen de datos y Protección para autentificación de acceso. Métrica: un número en el rango [0..1]

(Fuente: Autor de la Investigación)

Tabla 25. Lista de requisitos no funcionales con sus propiedades de calidad asociada

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

Nro. Identificación	Reglas del Negocio asociadas al dominio	Requisitos no funcionales derivados de las reglas del negocio	Características de calidad ISO/IEC 25010	Atributos (Características)
	Políticas			
1	Los servicios funcionan en una Intranet.	-Debe funcionar en maquinas con sistemas operativo Canaima. -Cumplimientos de estándares y normativas de la institución con el fin de garantizar que el sistema se ejecute.	**Portabilidad** -Instalabilidad **Portabilidad** - Adaptabilidad **Confiabilidad** - Disponibilidad	-Proporción de las funcionalidades que pueden ser adoptadas por el usuario -Atributo: Presencia de mecanismo. -Métrica booleano. Proporción de las funcionalidades que pueden ser adoptadas por el usuario -Atributo: Presencia de mecanismo. -Métrica booleano. Atributo: Tiempo de servicio agregado. La proporción del tiempo de servicio agregado que la aplicación está disponible. Métrica: Un número en rango de [0..1]
	Procesamiento			
2	Integridad de la información	- La transacción se debe se guardada completa y correctamente en la Base de datos y cumpliendo con las reglas de negocio de la institución. - El sistema debe garantizar que una vez que transacción se confirma, su efecto no se pierde en la base de datos, a pesar de fallos posteriores.	**Funcionalidad** -Preciso **Confiabilidad** - Recuperabilidad -Tolerancia a fallas	-Atributo: tiempo de completitud de tareas. Métrica: un número en el rango [1..10] -Presencia de mecanismo: Un mecanismo debe ser provisto. Métrica: Booleano
3	El servicio debe ser garantizado en la intranet	-Hacer posible el acceso a los servicios, lo cual implica, ancho de banda apropiado y garantizar las conexiones, teniendo	**Confiabilidad** - Disponibilidad	Atributo: Tiempo de servicio agregado. La proporción del tiempo de servicio agregado que la

		que solventar problema de redes. -Tiempo de transmisión apropiado, tiempo de repuesta adecuados dentro de rango establecido.	**Eficiencia** - Comportamiento en el tiempo	aplicación está disponible. Métrica: Un número en rango de [0..1] Atributo: Tiempo de espera. Segundos de espera. Métrica: un número en el rango de [1..5]
4	Solución centralizada con respecto a los datos	- Acceso centralizado a los datos mediante de interfaces conectadas a la base de datos.	**Funcionalidad** -Apropiado	-Atributo: tiempo de completitud de tareas. Métrica: un número en el rango [1..10]
	Implementación			
5	Fiabilidad	-El sistema en caso de fallas, debe tener mecanismos de recuperación y de respaldo de datos.	**Confiabilidad** - Recuperabilidad -Tolerancia a fallas	-Presencia de mecanismo: Un mecanismo debe ser provisto. Métrica: Booleano
6	Restricción de miembros del grupo	- Se permite el acceso únicamente a personal autorizado por la Institución.	**Seguridad** -Autenticidad - Confidencialidad	-Presencia de mecanismo: Un mecanismo debe ser provisto. Métrica: Booleano -Protección para el origen de datos y Protección para autentificación de acceso. Métrica: un número en el rango [0..1]

(Fuente: Autor de la Investigación)

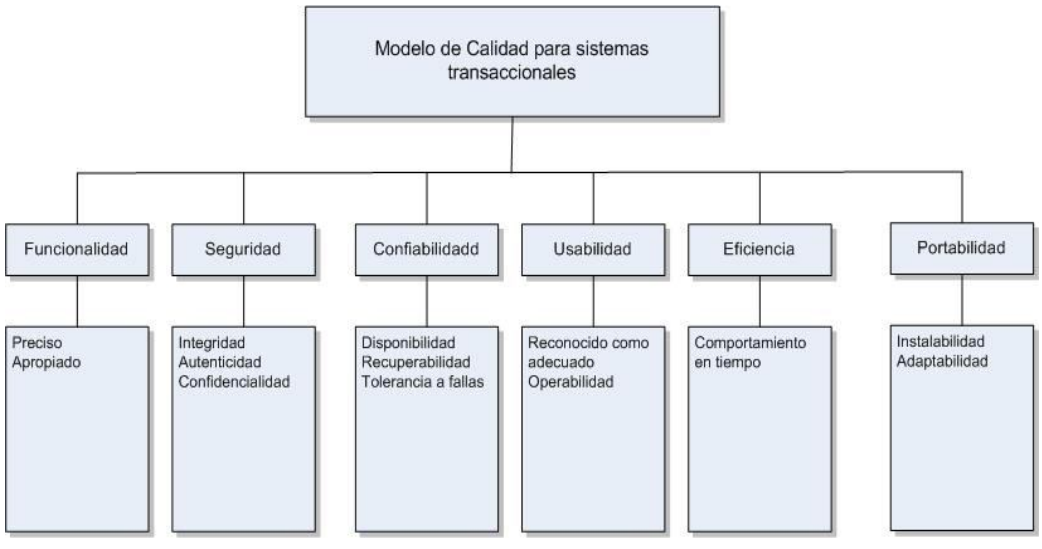
En la próxima sección se utiliza la información presentadas en la tabla 23 y 24 para definir el modelo de calidad asociado a los sistemas transaccionales.

Actividad 4: Obtener modelo de calidad asociado al dominio.

En la actividad obtener modelo de calidad asociado al dominio se obtiene el artefacto de modelo de calidad del dominio el cual representa la calidad de un producto de software del dominio como una expresión a cerca de la capacidad del software de ejecutar y mantener el nivel de servicio especificado. Las propiedades de calidad muestran el grado con el cual el software es capaz de proporcionar y mantener dichos servicios. En conclusión el modelo de calidad para el dominio de las aplicaciones de sistemas transaccionales viene dado por la tabla 26:

Tabla 26. Lista de requisitos no funcionales con sus propiedades de calidad asociada

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



(Fuente: Autor de la Investigación)

Actividad 5: Creación de escenarios de calidad del dominio.

En la actividad creación de escenarios de calidad del dominio se propone crear escenarios de calidad para representar los intereses de los grupos de evaluación y confirmar que se han evaluado los requisitos deseados de calidad. En la actividad se obtiene el artefacto escenarios de calidad conseguida dentro de las diversas vistas de los requisitos a través del modelo RECLAMO y se adiciona a cada sub-características de calidad la definición del escenario. Los escenarios de calidad se muestran a continuación tabla 27:

Tabla 27. Escenarios de Calidad

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	*Lista de requisitos no funcionales con sus propiedades de calidad asociada*
Constructor:	Analista de Requisitos
Nro. Identificación ACTIVIDAD:	A_05
Nro. Identificación ARTEFACTO:	9

Escenario de Calidad	Requisitos arquitecturales	Características de calidad ISO/IEC 25010	Atributos (Características)
1	-Debe funcionar en maquinas con sistemas operativo Canaima.	**Portabilidad** -Instalabilidad -Adaptabilidad	-Proporción de las funcionalidades que pueden ser adoptadas por el usuario -Atributo: Presencia de mecanismo. -Métrica booleano.
	-Cumplimientos de estándares y normativas de la institución con el fin de garantizar que el sistema se ejecute.	**Confiabilidad** -Disponibilidad	Atributo: Tiempo de servicio agregado. La proporción del tiempo de servicio agregado que la aplicación está disponible. Métrica: Un número en rango de [0..1]

2	- La transacción se debe se guardada completa y correctamente en la Base de datos y cumpliendo con las reglas de negocio de la institución. - El sistema debe garantizar que una vez que transacción se confirma, su efecto no se pierde en la base de datos, a pesar de fallos posteriores.	**Funcionalidad** -Preciso **Confiabilidad** -Recuperabilidad -Tolerancia a fallas	-Atributo: tiempo de completitud de tareas. Métrica: un número en el rango [1..10] -Presencia de mecanismo: Un mecanismo debe ser provisto. Métrica: Booleano
3	-Hacer posible el acceso a los servicios, lo cual implica, ancho de banda apropiado y garantizar las conexiones, teniendo que solventar problema de redes. -Tiempo de transmisión apropiado, tiempo de repuesta adecuados dentro de rango establecido.	**Confiabilidad** -Disponibilidad **Eficiencia** -Comportamiento en el tiempo	Atributo: Tiempo de servicio agregado. La proporción del tiempo de servicio agregado que la aplicación está disponible. Métrica: Un número en rango de [0..1] Atributo: Tiempo de espera. Segundos de espera. Métrica: un número en el rango de [1..5]
4	- Acceso centralizado a los datos mediante de interfaces conectadas a la base de datos.	**Funcionalidad** -Apropiado	-Atributo: tiempo de completitud de tareas. Métrica: un número en el rango [1..10]
5	-El sistema en caso de fallas, debe tener mecanismos de recuperación y de respaldo de datos.	**Confiabilidad** -Recuperabilidad -Tolerancia a fallas	-Presencia de mecanismo: Un mecanismo debe ser provisto. Métrica: Booleano
6	- Se permite el acceso únicamente a personal autorizado por la Institución.	**Seguridad** -Autenticidad -Confidencialidad -Integridad	-Presencia de mecanismo: Un mecanismo debe ser provisto. Métrica: Booleano -Protección para el origen de datos y Protección para autentificación de acceso. Métrica: un número en el rango [0..1]

(Fuente: Autor de la Investigación)

Actividad 6: Identificar los estilos arquitecturales para el dominio.

En la actividad identificar los estilos arquitecturales para el dominio se produce el artefacto de estilos arquitecturales que soporta el diseño arquitectural de las aplicaciones de sistemas transaccionales en el cual se representan los componentes y las relaciones entre ellos con las restricciones de su aplicación y las asociaciones y reglas del diseño para su construcción. Es importante señalar que en esta actividad es relevante la experticia del arquitecto en el estudio de los estilos a ser incorporados al artefacto resultante atendiendo al conjunto minimal de requisitos funcionales y no funcionales ajustados al modelo de calidad del dominio. Previo a la identificación de los estilos arquitecturales que conforman el artefacto estilos arquitecturales de los sistemas transaccionales, debemos considerar los artefactos conjunto minimal de requisitos funcionales y no funcionales y modelo de calidad del dominio en donde existen características de calidad que deben ser garantizadas por los estilos arquitecturales, en particular requisitos como:

- La transacción se debe se guardada completa y correctamente en la Base de datos y cumpliendo con las reglas de negocio de la institución. **Funcionalidad** (Preciso).
- Acceso centralizado a los datos mediante de interfaces conectadas a la base de datos. . **Funcionalidad** (Apropiado).
- Hacer posible el acceso a los servicios, lo cual implica, ancho de banda apropiado y garantizar las conexiones, teniendo que solventar problema de redes. **Confiabilidad** (Disponibilidad).
- Tiempo de transmisión apropiado, tiempo de repuesta adecuados dentro de rango establecido. **Eficiencia** (Comportamiento en el tiempo).

Por los dichos anteriormente, nos da muestra que estilos como Arquitecturas cliente-servidor 3-Capas y arquitectura Modelo-Vista-Controlador, es por ello que a continuación se describen las características básicas y referencias de experiencias en

el uso de estos estilos como posibles elementos del artefacto estilos arquitecturales de esta actividad:

- **Arquitecturas cliente-servidor:** es un modelo de sistema en el que dicho sistema se organiza como un conjunto de servicios y servidores asociadas, más unos de los clientes que acceden y usan los servicios (Sommerville, 2005). Los componentes de este modelo son:
  - Un conjunto de servidores que ofrecen servicios a otros subsistemas.
  - Un conjunto de clientes que llaman a los servicios ofrecidos por los servidores.
  - Una red que permite a los clientes acceder a estos servicios. Esto no es estrictamente necesario ya que los clientes y los servidores podría ejecutarse sobre la única maquina. Sin embargo, la mayoría de los sistemas cliente-servidor se implementan como sistemas distribuidos.

La arquitectura cliente-servidor, se divide en cliente-servidor dos capas y cliente-servidor tres capas. A continuación se describe cada una de ella:

- Cliente-servidor dos capas: una aplicación se organiza como un servidor (o múltiple servidores idénticos) y un conjunto de clientes. Un ejemplo de este tipo de arquitectura es la de los sistemas bancarios ATM, en donde cada ATM es un cliente y el servidor es un mainframe que procesa la cuenta del cliente en la base de datos.
- Cliente-servidor tres capas: en esta arquitectura la aplicación esta constituida por procesos lógicamente separados en presentación, el procesamiento de la aplicación y la gestión de los datos que se ejecutan sobre procesadores diferentes. Es recomendable usarlo cuando las aplicaciones posean cientos o miles de clientes, en donde tanto los datos como la aplicación son volátiles y se integran datos de múltiples fuentes. Un ejemplo es un sistema bancario por Internet, en el cual la base de datos de clientes del banco proporciona servicios de gestión de datos, un servidor web proporciona los servicios de aplicación (Transferir efectivo, generar estados de cuenta, entre otros) y la computadora del usuario con un navegador de Internet es el cliente.

o**Arquitectura Modelo-Vista-Controlador:** es un patrón de diseño de arquitectura de software usado principalmente en aplicaciones que manejan gran cantidad de datos y transacciones complejas donde se requiere una mejor separación de conceptos para que el desarrollo este estructurado de una mejor manera facilitando la programación en diferentes capas de manera paralela e independiente (Rivera, 2008). MVC sugiere la separación del software en tres (3) estratos:

- Modelo: Es la representación de la información que maneja la aplicación. El modelo en si son los datos puros en contexto del sistema proveen de información al usuario o a la aplicación misma.
- Vista: Es la representación del modelo en forma grafica disponible para la interacción con el usuario.
- Controlador: Es la capa encargada de manejar y responde las solicitudes del usuario, procesando la información necesaria y modificando el Modelo en caso de ser necesario.

El ciclo de vida de la arquitectura MVC es representado por tres capas mencionadas anteriormente y el usuario. En la figura 25, se muestra el diagrama representa el ciclo de vida:

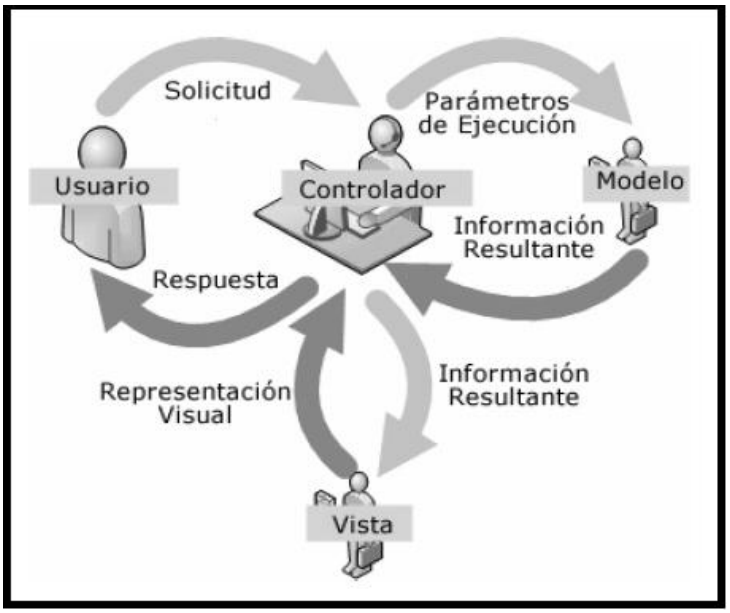


Figura 25. Diagrama de Ciclo de Vida de la Arquitectura MVC.

(Fuente: Rivera, 2008)

El primer paso en el ciclo de vida empieza cuando el usuario hace una solicitud al Controlador con información sobre lo que el usuario desea realizar, entonces el Controlador decide a quien debe delegar la tarea y es aquí donde el Modelo empieza su trabajo. En esta etapa, el Modelo se encarga de realizar operaciones sobre la información que maneja para cumplir con lo que le solicita el Controlador. Una vez que termina su labor, le regresa al Controlador la información resultante de sus operaciones, el cuál a su vez redirige a la Vista. La Vista se encarga de transformar los datos en información visualmente entendible para el usuario. Finalmente, la representación grafica es transmitida de regreso al Controlador y éste se encarga de transmitírsela al usuario. El ciclo entero puede empezar nuevamente si el usuario así lo requiere.

El artefacto los estilos arquitecturales, se muestran a continuación tabla 28:

Tabla 28. 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
Arquitectura cliente-servidor (Tres Capas) Modelo-Vista-Controlador(MVC)	

(Fuente: Autor de la Investigación)

Actividades del Diseño del Dominio.

Actividad 7: Seleccionar elementos del diseño del dominio que satisfagan el conjunto minimal de requisitos funcionales y no funcionales.

Mediante esta actividad se procede a escoger los elementos del diseño (sistema, subsistema conceptual, componente conceptual) a descomponer. Todas las entradas para este elemento del diseño deben estar disponibles (limitaciones, requisitos

funcionales y de calidad). En esta actividad se produce el artefacto denominado elementos de diseño del dominio para sistemas transaccionales, mostrado en la tabla 29, que a continuación se presenta:

Tabla 29. Elementos de diseño del dominio.

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

Nro. Identificación	**Descripción del requisito funcional**	**Importancia**	**Dificultad**
1	Manejo de nóminas por vicerrectorados y núcleos.	Alta	Media
2	Generar reportes de listado por vicerrectorados y/o núcleos.	Alta	Bajo
3	Generar reportes nóminas por vicerrectorados y/o núcleos.	Alta	Alta
4	Gestionar los cargos de la institución con sus dedicaciones, remuneraciones ordinarias y prestaciones sociales asociadas.	Alta	Alta
5	Generar reportes de remuneración ordinaria por cargo.	Media	Media
6	Generar listado de conceptos de las remuneraciones.	Media	Baja
7	Gestionar Cargos y el tipo de dedicación.	Alta	Media
8	Gestionar la información relacionados a los empleados como datos personales, cargo y escalas dentro la institución, cuenta de banco para el pago de la nómina, prestaciones sociales generadas y el historial de cargo.	Alta	Media
9	Generar reportes sobre el empleado y sus datos básicos.	Alta	Alta
10	Listado de funcionarios asignados a fondos de pensionados y jubilados.	Alta	Media
11	Consulta de Recibos de pagos.	Alta	Alta
12	Asignar Cargos y el tipo de dedicación.	Alta	Media
13	Listado de funcionarios asignados por bancos.	Alta	Media
14	Cumplimientos de estándares y normativas de la institución con el fin de garantizar que el sistema se ejecute.	Alta	Alta
15	La transacción se debe se guardada completa y	Alta	Media

	correctamente en la Base de datos y cumpliendo con las reglas de negocio de la institución.		
16	El sistema debe garantizar que una vez que transacción se confirma, su efecto no se pierde en la base de datos, a pesar de fallos posteriores.	Alta	Media
17	Hacer posible el acceso a los servicios, lo cual implica, ancho de banda apropiado y garantizar las conexiones, teniendo que solventar problema de redes.	Media	Baja
18	Tiempo de transmisión apropiado, tiempo de repuesta adecuados dentro de rango establecido.	Alta	Media
19	Acceso centralizado a los datos mediante de interfaces conectadas a la base de datos.	Media	Media
20	El sistema en caso de fallas, debe tener mecanismos de recuperación y de respaldo de datos.	Alta	Media
21	Se permite el acceso únicamente a personal autorizado por la Institución.	Alta	Alta

(Fuente: Autor de la Investigación)

Actividad 8: Escoger patrones arquitecturales candidatos.

En la actividad escoger patrones arquitecturales candidatos se produce el artefacto patrones arquitecturales candidatos, que consiste de un conjunto de patrones presentados como una jerarquía que son estudiados y revisados como posible solución arquitectural a los sistemas transaccionales, el cual se muestra en la tabla 30:

Tabla 30. Patrones arquitecturales candidatos.

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

Nro. Patron	**Nombre**	**Descripción**	**Favorece**	**Componentes Conectores**	**Ejemplo**

1	Proxy	Proporciona un sustituto de otro objeto para controlar su acceso	**Seguridad**	Componentes	ProxyWeb
2	Wrapper Facade	Encapsula las funciones y los datos proporcionados por las APIs no orientada a objetos, en clases de interfaces mas concisas, robustas, portables y mantenibles	**Portabilidad** **Confiabilidad** (Disponibilidad) **Mantenibilidad**	**Componentes** Aplicación Componente Funciones API **Conectores** Mecanismo de comunicación de las clases	Librerias de Java, Microsoft, Php.
3	Network Communication Protocol	Ancho de banda, medida de dirección durante la ejecución.			
4	Multi-database	Permite la manipulación de los datos centrales de las bases de datos, a través de componentes externa. Organiza las bases de datos distribuidas sobre un modelo cliente/servidor donde mediadores aceptan queries de los usuarios, los reconducen a las BD disponibles y devuelven las respuestas adecuadas.	**Portabilidad** (Adaptabilidad)	**Componentes** Repositorio de datos centrales y agentes. **Conectores** Mecanismos y protocolos de comunicación entre componentes	Implementaciones de ODBC,JDBC, Oracle
5	Facade	Proporcionar una interfaz unificada para un conjunto de interfaces en un subsistema. Define una interfaz de alto nivel que hace que el subsistema sea más fácil de usar.	**Portabilidad** **Mantenibilidad**		
6	Data Transfer Object (Fowler, 2003)	Contenedores de datos para el transporte entre los subsistemas de aplicación de software. DTO se utilizan a menudo en relación con los objetos de acceso a datos para recuperar datos de una base de datos.	**Eficiencia** Comportamiento en tiempo **Usabilidad** Operabilidad	**Componentes** Datos vinculados al resultado de una transacción **Conectores** Mecanismos y protocolos de comunicación entre componentes	Objetos simples (POJOs) en java
7	Abstract Factory(Gamma et al.,1995)	Interfaz para crear familias de objetos sin especificar sus clases.	**Portabilidad** (Adaptabilidad)		

8	Adapter	Traductor que adapta la interfaz de un servidor a un cliente	**Portabilidad** (Adaptabilidad)		
9	Transaction Script(Fowler, 2003)	Organiza la lógica de negocio mediante procedimientos en los que cada procedimiento se ocupa de una sola solicitud de la presentación, haciendo las llamadas directamente a la base de datos. Cada transacción tendrá su propia secuencia de comandos de transacción, a pesar de subtareas comunes que se pueden dividir en sub procedimientos.	**Usabilidad** Reconocido como adecuado Operabilidad **Funcionalidad** Preciso Apropiado		
10	Server Session State(Fowler, 2003)	Mantiene el estado de sesión en un sistema de servidor de una forma serializada	**Seguridad** Autenticidad Confidencialidad Integridad		
11	Unit of Work(Fowler, 2003)	Mantiene una lista de los objetos afectados por una transacción comercial a cabo y coordina la redacción de los cambios y la resolución de los problemas de concurrencia.	**Funcionalidad** Preciso Apropiado **Confiabilidad** Recuperabilidad Tolerancia a fallas		
12	Optimistic Offline Lock	Evita los conflictos entre las transacciones de negocios simultáneas mediante la detección de un conflicto y hacer retroceder la transacción.	**Confiabilidad** Recuperabilidad Tolerancia a fallas **Seguridad** Integridad		

(Fuente: Autor de la Investigación)

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

En la actividad instanciar elementos arquitecturales para los elementos del diseño del dominio se obtiene el artefacto elementos arquitecturales (Componentes y conectores). La actividad consiste en seleccionar patrones arquitecturales desde el artefacto patrones arquitecturales candidatos que satisfacen el artefacto elementos del diseño del dominio. Los patrones arquitecturales escogidos presentan diferentes niveles de calidad, por lo que se escoge la solución más apropiada para la más alta prioridad:

Tabla 31. Elementos arquitecturales (componentes y conectores).

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

Nro. Identificación	**Descripción del requisito funcional**	**Nombre del patrón escogido**	**Numero del patrón escogido**
1	Manejo de nóminas por vicerrectorados y núcleos.	-Proxy -Wrapper Facade -Network Communication Protocol -Multidatabase - Facade -Data Transfer Object - Adapter - Unit of Work	1,2,3,4,5,6, 8,11
2	Generar reportes de listado por vicerrectorados y/o núcleos.	-Wrapper Facade -Network Communication Protocol -Multidatabase	2,3,4,5,6,8, 9

		- Facade -Data Transfer Object - Adapter -Transaction Script	
3	Generar reportes nóminas por vicerrectorados y/o núcleos.	-Wrapper Facade -Network Communication Protocol -Multidatabase - Facade -Data Transfer Object - Adapter -Transaction Script - Unit of Work	2,3,4,5,6,8, 9,11
4	Gestionar los cargos de la institución con sus dedicaciones, remuneraciones ordinarias y prestaciones sociales asociadas.	-Proxy -Wrapper Facade -Network Communication Protocol -Multidatabase - Facade -Data Transfer Object - Unit of Work	1,2,3,4,5,6, 11
5	Generar reportes de remuneración ordinaria por cargo.	-Wrapper Facade -Network Communication Protocol -Multidatabase - Facade -Data Transfer Object - Adapter -Transaction Script	2,3,4,5,6,8, 9
6	Generar listado de conceptos de las remuneraciones.	-Wrapper Facade -Network Communication Protocol -Multidatabase - Facade -Data Transfer Object - Adapter	2,3,4,5,6,8, 9

		-Transaction Script	
7	Gestionar Cargos y el tipo de dedicación.	-Proxy -Wrapper Facade -Network Communication Protocol -Multidatabase - Facade -Data Transfer Object - Unit of Work	1,2,3,4,5,6, 11
8	Gestionar la información relacionados a los empleados como datos personales, cargo y escalas dentro la institución, cuenta de banco para el pago de la nómina, prestaciones sociales generadas y el historial de cargo.	-Proxy -Wrapper Facade -Network Communication Protocol -Multidatabase - Facade -Data Transfer Object - Unit of Work	1,2,3,4,5,6, 11
9	Generar reportes sobre el empleado y sus datos básicos.	-Wrapper Facade -Network Communication Protocol -Multidatabase - Facade -Data Transfer Object - Adapter -Transaction Script	2,3,4,5,6,8, 9
10	Listado de funcionarios asignados a fondos de pensionados y jubilados.	-Wrapper Facade -Network Communication Protocol -Multidatabase - Facade -Data Transfer Object - Adapter -Transaction Script	2,3,4,5,6,8, 9
11	Consulta de Recibos de pagos.	-Proxy -Wrapper Facade -Network Communicatio	1,2,3,4,5,6, 11

		n Protocol -Multidatabase - Facade -Data Transfer Object - Unit of Work	
12	Asignar Cargos y el tipo de dedicación.	-Proxy -Wrapper Facade -Network Communicatio n Protocol -Multidatabase - Facade -Data Transfer Object - Unit of Work	1,2,3,4,5,6, 11
13	Listado de funcionarios asignados por bancos.	-Wrapper Facade -Network Communicatio n Protocol -Multidatabase - Facade -Data Transfer Object - Adapter -Transaction Script	2,3,4,5,6,8, 9
14	Cumplimientos de estándares y normativas de la institución con el fin de garantizar que el sistema se ejecute.	- Abstract Factory - Unit of Work	7,11
15	La transacción se debe se guardada completa y correctamente en la Base de datos y cumpliendo con las reglas de negocio de la institución.	-Transaction Script - Unit of Work -Optimistic Offline Lock	9,11,12
16	El sistema debe garantizar que una vez que transacción se confirma, su efecto no se pierde en la base de datos, a pesar de fallos posteriores.	-Optimistic Offline Lock	12
17	Hacer posible el acceso a los servicios, lo cual implica, ancho de banda apropiado y garantizar las conexiones, teniendo que solventar problema de redes.	-Network Communicatio n Protocol	3
18	Tiempo de transmisión apropiado, tiempo de repuesta adecuados dentro de rango establecido.	-Transaction Script - Unit of Work	9,11
19	Acceso centralizado a los datos mediante de interfaces conectadas a la base de datos.	-Proxy -Wrapper Facade - Facade	1,2,5,7,10

		- Abstract Factory -Server Session State	
20	El sistema en caso de fallas, debe tener mecanismos de recuperación y de respaldo de datos.	- Unit of Work -Optimistic Offline Lock	11,12
21	Se permite el acceso únicamente a personal autorizado por la Institución.	-Proxy -Multidatabase	1,4

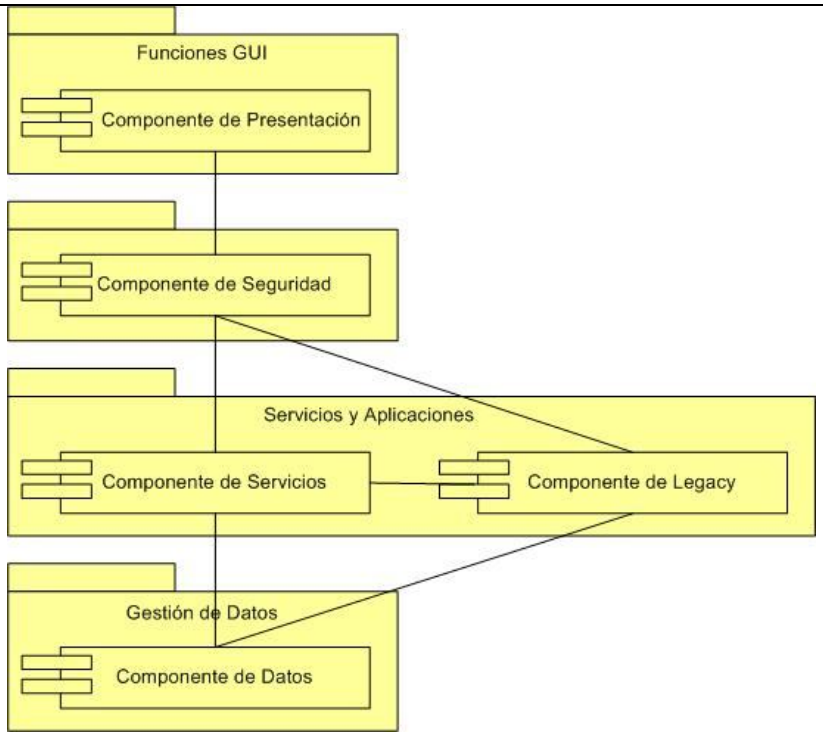
(Fuente: Autor de la Investigación)

Actividad 10: Identificar similitudes entre elementos arquitecturales instanciados.

Esta actividad denominada Identificar similitudes entre elementos arquitecturales instanciados posee la generación de un artefacto llamado elementos arquitecturales similares, en el cual se obtiene los elementos arquitecturales similares obtenidos mediante la agrupación de las similitudes entre los componentes instanciados como parte de los catálogos de patrones arquitecturales incorporados en los activos arquitecturales. La estructura de la arquitectura se visualiza mediante un diagrama de componentes. A continuación se presentan los componentes principales y se incorpora el diagrama de componente en la tabla 32:

Tabla 32. Elementos Arquitecturales Similares.

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



- Componente de presentación: Esta es la capa que interactúa con el usuario final, es la encargada de presentar la información y recolectarla para hacer uso de los servicios expuestos por la capa de servicio, para satisfacer los casos de uso de la aplicación.

-Componente de Seguridad (SSL, autenticación y autorización, y sesión): Se encarga de identificar los usuarios, y permitirles utilizar y manipular la información adecuada. Este componente esta subdividido en tres módulos, el primero se encarga del cifrado y descifrado de los datos (SSL), el segundo del proceso de autenticación y autorización de los usuarios para revisar si poseen permiso o no para entrar a la aplicación y el tercero el proceso verificación de sesiones.

-Componente de datos: Este componente esta conformado por la Base de Datos y el componente administrador de la base de datos. La primera contiene los datos relacionados a las transacciones realizadas. El componente Administrador de la Base de Datos es el encargado de realizar las consultas y modificaciones a la información almacenada en la base de datos. La base de datos utilizada será una base de datos relacional.

-Componente de servicios: se dividen en dos tipo: servicios de utilidad y servicios del sistema transaccional de Recurso Humano. Entre los servicios de utilidad esta la administración del sistema e integración con sistemas externos. Los servicios del Sistema Transaccional implementan comportamientos del negocio soportado por la aplicación y se apoya sobre los servicios de utilidad, de esta forma las personalizaciones del modelo de negocio consistirían en modificar o establecer parámetros para estos servicios.

-Componente Legacy: permite la importación y exportación de datos de la aplicación, para su integración con otros sistemas como sistema de presupuesto es un sistema donde asocian los códigos presupuestarios dados por el gobierno con los gastos del personal de

	la institución con la finalidad de llevar detalladamente los gastos realizados por la institución.

(Fuente: Autor de la Investigación)

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

La actividad decidir la selección de la arquitectura como solución arquitectural candidata incorpora las soluciones alternativas o parciales que satisfacen el modelo de calidad del dominio, agregadas en el artefacto conjunto de soluciones arquitecturales candidatas obtenidas en esta actividad y presentadas en la tabla 33:

Tabla 33. Conjunto de soluciones candidatas.

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

Nro. Identificación	**Descripción del requisito funcional**	**Nombre del patrón escogido**	**Numero del patrón escogido**
1	Manejo de nóminas por vicerrectorados y núcleos.	-Proxy -Network Communication Protocol -Multidatabase - Facade -Data Transfer Object - Adapter - Unit of Work	1,3,4,5,6,8, 11
2	Generar reportes de listado por vicerrectorados y/o núcleos.	-Network Communication Protocol -Multidatabase - Facade -Data Transfer	3,4,5,6,8,9

		Object - Adapter -Transaction Script	
3	Generar reportes nóminas por vicerrectorados y/o núcleos.	-Network Communication Protocol -Multidatabase - Facade -Data Transfer Object - Adapter - Unit of Work	3,4,5,6,8,11
4	Gestionar los cargos de la institución con sus dedicaciones, remuneraciones ordinarias y prestaciones sociales asociadas.	-Proxy -Network Communication Protocol -Multidatabase - Facade -Data Transfer Object - Unit of Work	1,3,4,5,6,11
5	Generar reportes de remuneración ordinaria por cargo.	-Network Communication Protocol -Multidatabase - Facade -Data Transfer Object - Adapter -Transaction Script	3,4,5,6,8,9
6	Generar listado de conceptos de las remuneraciones.	-Network Communication Protocol -Multidatabase - Facade -Data Transfer Object - Adapter -Transaction Script	3,4,5,6,8,9
7	Gestionar Cargos y el tipo de dedicación.	-Proxy -Network Communication Protocol -Multidatabase - Facade -Data Transfer Object - Unit of Work	1,3,4,5,6,11
8	Gestionar la información relacionados a los	-Proxy	1,3,4,5,6,11

	empleados como datos personales, cargo y escalas dentro la institución, cuenta de banco para el pago de la nómina, prestaciones sociales generadas y el historial de cargo.	-Network Communication Protocol -Multidatabase - Facade -Data Transfer Object - Unit of Work	
9	Generar reportes sobre el empleado y sus datos básicos.	-Network Communication Protocol -Multidatabase - Facade -Data Transfer Object - Adapter -Transaction Script	3,4,5,6,8,9
10	Listado de funcionarios asignados a fondos de pensionados y jubilados.	-Network Communication Protocol -Multidatabase - Facade -Data Transfer Object - Adapter -Transaction Script	3,4,5,6,8,9
11	Consulta de Recibos de pagos.	-Proxy -Network Communication Protocol -Multidatabase - Facade -Data Transfer Object - Unit of Work	1,3,4,5,6,11
12	Asignar Cargos y el tipo de dedicación.	-Proxy -Network Communication Protocol -Multidatabase - Facade -Data Transfer Object - Unit of Work	1,3,4,5,6,11
13	Listado de funcionarios asignados por bancos.	-Network Communication Protocol -Multidatabase - Facade -Data Transfer Object	3,4,5,6,8,9

		- Adapter -Transaction Script	
14	Cumplimientos de estándares y normativas de la institución con el fin de garantizar que el sistema se ejecute.	- Unit of Work	11
15	La transacción se debe se guardada completa y correctamente en la Base de datos y cumpliendo con las reglas de negocio de la institución.	- Unit of Work -Optimistic Offline Lock	11,12
16	El sistema debe garantizar que una vez que transacción se confirma, su efecto no se pierde en la base de datos, a pesar de fallos posteriores.	-Optimistic Offline Lock	12
17	Hacer posible el acceso a los servicios, lo cual implica, ancho de banda apropiado y garantizar las conexiones, teniendo que solventar problema de redes.	-Network Communicatio n Protocol	3
18	Tiempo de transmisión apropiado, tiempo de repuesta adecuados dentro de rango establecido.	- Unit of Work	11
19	Acceso centralizado a los datos mediante de interfaces conectadas a la base de datos.	-Proxy - Facade -Server Session State	1,5,10
20	El sistema en caso de fallas, debe tener mecanismos de recuperación y de respaldo de datos.	- Unit of Work -Optimistic Offline Lock	11,12
21	Se permite el acceso únicamente a personal autorizado por la Institución.	-Proxy -Multidatabase	1,4

(Fuente: Autor de la Investigación)

Al mismo tiempo, se incorpora la información del proceso de decisión en el otro producto artefacto de esta actividad denominado soporte de decisión arquitectural conformado por: comentarios de la escogencia, decisiones consideradas y rechazadas, trazabilidad de la decisión. Este artefacto soporte de decisión arquitectural previamente denotado y expuesto en la tabla 34 será de ayuda en el proceso de evaluación a llevar a cabo en la siguiente fase.

Tabla 34. Soporte de decisión arquitectural.

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	*Soporte de decisión arquitectural*
Constructor:	Arquitecto de software
Nro. Identificación ACTIVIDAD:	A_11
Nro. Identificación ARTEFACTO:	16
En la lógica del diseño se ha separado en componente de presentación, componente de servicios y componente de datos, con el objeto de ser escalable y más segura ante ataques. Esta propiedad permite separar físicamente la capa de presentación de la capa de servicio (en un Servidor de Presentación y otro Servidor de Aplicaciones). En estos casos, puede utilizarse un firewall para poner la limitación de que sólo el Servidor de Presentación tenga acceso al Servidor de Aplicaciones. Del mismo modo, si se separa físicamente la componente de servicios del componente de datos (en un Servidor de Aplicaciones y otro Servidor de Datos) puede limitarse con un firewall el acceso al Servidor de Datos y permitirle acceder solo al Servidor de Aplicaciones. De esta forma se dispone de un único punto de acceso a la base de datos, de forma que todas las operaciones contra ellas se ejecutaran desde un único punto. Adicionalmente, la Componente de Seguridad se encarga de identificar los usuarios, y permitirles utilizar y manipular la información adecuada y esta subdividido en tres módulos, el primero se encarga del cifrado y descifrado de los datos (SSL), el segundo de los procesos de firmado y verificación de firmas. La Componente de servicios: se dividen en dos tipo: servicios de utilidad y servicios del sistema transaccional de Recurso Humano. Entre los servicios de utilidad esta la administración del sistema e integración con sistemas externos. Los servicios del Sistema Transaccional implementan comportamientos del negocio soportado por la aplicación y se apoya sobre los servicios de utilidad, de esta forma las personalizaciones del modelo de negocio consistirían en modificar o establecer parámetros para estos servicios.	

(Fuente: Autor de la Investigación)

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

Esta actividad denominada validar modelo de calidad del dominio con arquitecturas candidatas procederá a revisar el cumplimiento de las características de calidad del modelo de calidad del dominio por parte de las arquitecturas candidatas. Una vez, que se chequean las características de calidad de la solución arquitectural y las mismas cumplen el modelo de calidad, se incorpora como arquitectura validada en

el artefacto arquitectura validada a la familia producto de esta actividad y mostrado a continuación:

Tabla 35. Arquitectura validada a la familia producto.

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

Nro. Identificación	**Requisitos no funcionales derivados de las reglas del negocio**	**Propiedades de Calidad asociadas a los requisitos no funcionales (ISO/IEC 25010)**	**Nombre del patrón escogido**
1	Cumplimientos de estándares y normativas de la institución con el fin de garantizar que el sistema se ejecute.	**Portabilidad** Instalabilidad Adaptabilidad **Confiabilidad** Disponibilidad	- Unit of Work
2	La transacción se debe se guardada completa y correctamente en la Base de datos y cumpliendo con las reglas de negocio de la institución.	**Funcionalidad** -Preciso	- Unit of Work -Optimistic Offline Lock
3	El sistema debe garantizar que una vez que transacción se confirma, su efecto no se pierde en la base de datos, a pesar de fallos posteriores.	**Confiabilidad** Recuperabilidad Tolerancia a fallas	-Optimistic Offline Lock
4	Hacer posible el acceso a los servicios, lo cual implica, ancho de banda apropiado y garantizar las conexiones, teniendo que solventar problema de redes.	**Confiabilidad** Disponibilidad	-Network Communication Protocol
5	Tiempo de transmisión apropiado, tiempo de repuesta adecuados dentro de rango establecido.	**Eficiencia** Comportamiento en el tiempo	- Unit of Work

6	Acceso centralizado a los datos mediante de interfaces conectadas a la base de datos.	**Funcionalidad** Apropiado	-Proxy - Facade -Server Session State
7	El sistema en caso de fallas, debe tener mecanismos de recuperación y de respaldo de datos.	**Confiabilidad** Recuperabilidad Tolerancia a fallas	- Unit of Work -Optimistic Offline Lock
8	Se permite el acceso únicamente a personal autorizado por la Institución.	**Seguridad** Autenticidad Confidencialidad Integridad	-Proxy - Multidatabase

(Fuente: Autor de la Investigación)

Así mismo, el razonamiento de la decisión es soportado en el artefacto denominado documento de razonamiento arquitectural:

Tabla 36. Documento de razonamiento arquitectural.

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

En la arquitectura cliente-servidor tres capas la aplicación esta constituida por procesos lógicamente separados en presentación, el procesamiento de la aplicación y la gestión de los datos que se ejecutan sobre procesadores diferentes. Es recomendable usarlo cuando las aplicaciones posean cientos o miles de clientes, en donde tanto los datos como la aplicación son volátiles y se integran datos de múltiples fuentes. Adicionalmente, la arquitectura Modelo-Vista-Controlador es un patrón de diseño de arquitectura de software usado principalmente en aplicaciones que manejan gran cantidad de datos y transacciones complejas donde se requiere una mejor separación de conceptos para que el desarrollo este estructurado de una mejor manera facilitando la programación en diferentes capas de manera paralela.

Por esta razón, la lógica del diseño se ha separado en componente de presentación, componente de servicios y componente de datos, con el objeto de ser escalable y más segura ante ataques. Esta propiedad permite separar físicamente la capa de presentación de la capa de servicio (en un Servidor de Presentación y otro Servidor de Aplicaciones). En estos casos, puede utilizarse un firewall para poner la limitación de que sólo el Servidor de Presentación tenga acceso al Servidor de Aplicaciones. Del mismo modo, si se separa físicamente la componente de servicios del

componente de datos (en un Servidor de Aplicaciones y otro Servidor de Datos) puede limitarse con un firewall el acceso al Servidor de Datos y permitirle acceder solo al Servidor de Aplicaciones. De esta forma se dispone de un único punto de acceso a la base de datos, de forma que todas las operaciones contra ellas se ejecutaran desde un único punto.

INDICADORES DE RAZONAMIENTO

Riesgos:

En la propiedad Confiabilidad (Recuperabilidad, Tolerancia a fallas) y Funcionalidad (Preciso) para garantizar que una transacción se mantenga en la base de datos, a pesar de que pueda presentarse fallos posteriores se considero los patrones Unit of Work y Optimistic Offline Lock con el riesgo de que el tiempo de respuestas de la aplicacion se degrade.

No Riesgos:

En la propiedad Confiabilidad (disponibilidad) utilizando patrones como proxy y el uso apropiado de protocolos de redes existe garantía de la propiedad.

Puntos Sensitivos

En la propiedad Seguridad (autencidad, confidencialidad) al estar presente la confidencialidad amerita mejorar el nivel de encriptamiento de la técnica usada.

En la propiedad Confiabilidad para mantener la integridad de la transacción y permitir la recuperación del sistema en posible falla amerita mas procesamiento afectándose la eficiencia.

Puntos de Compensación

En la propiedad Seguridad (autenticidad, confidencialidad) presente al manejar el nivel de encriptamiento puede tener un impacto en la seguridad y eficiencia en la ejecución.

Nro. Identificación	**Requisitos no funcionales derivados de las reglas del negocio**	**Propiedades de Calidad asociadas a los requisitos no funcionales (ISO/IEC 25010)**	**Nombre del patrón escogido**	**Comentarios Decisiones consideradas, rechazadas**
1	Cumplimientos de estándares y normativas de la institución con el fin de garantizar que el sistema se ejecute.	**Portabilidad** Instalabilidad Adaptabilidad **Confiabilidad** Disponibilidad	- Unit of Work	Mantiene el seguimiento de todo lo que ocurren en una transacción de negocio.
2	La transacción se debe se guardada completa y correctamente en la Base de datos y cumpliendo con las reglas de negocio de la institución.	**Funcionalidad** -Preciso	- Unit of Work	Mantiene el seguimiento de todo lo que ocurren en una transacción de negocio.

			-Optimistic Offline Lock	Previene conflictos de concurrencia en transacciones de negocio, detectando un conflicto y realizando un rollback de la transacción.
3	El sistema debe garantizar que una vez que transacción se confirma, su efecto no se pierde en la base de datos, a pesar de fallos posteriores.	**Confiabilidad** Recuperabilidad Tolerancia a fallas	-Optimistic Offline Lock	Previene conflictos de concurrencia en transacciones de negocio, detectando un conflicto y realizando un rollback de la transacción.
4	Hacer posible el acceso a los servicios, lo cual implica, ancho de banda apropiado y garantizar las conexiones, teniendo que solventar problema de redes.	**Confiabilidad** Disponibilidad	-Network Communication Protocol	Manejo apropiado de redes y optimizacion en el uso del dispositivo
5	Tiempo de transmisión apropiado, tiempo de repuesta adecuados dentro de rango establecido.	**Eficiencia** Comportamiento en el tiempo	- Unit of Work	Mantiene el seguimiento de todo lo que ocurren en una transacción de negocio.
6	Acceso centralizado a los datos mediante de interfaces conectadas a la base de datos.	**Funcionalidad** Apropiado	-Proxy - Facade -Server Session State	Se considero WrapperFacade pero es muy compleja su

				implementación.
7	El sistema en caso de fallas, debe tener mecanismos de recuperación y de respaldo de datos.	**Confiabilidad** Recuperabilidad Tolerancia a fallas	- Unit of Work -Optimistic Offline Lock	Mantiene el seguimiento de todo lo que ocurren en una transacción de negocio. Previene conflictos de concurrencia en transacciones de negocio, detectando un conflicto y realizando un rollback de la transacción.
8	Se permite el acceso únicamente a personal autorizado por la Institución.	**Seguridad** Autenticidad Confidencialidad Integridad	-Proxy - Multidatabase	Con un mecanismo de seguridad se aporta la solución

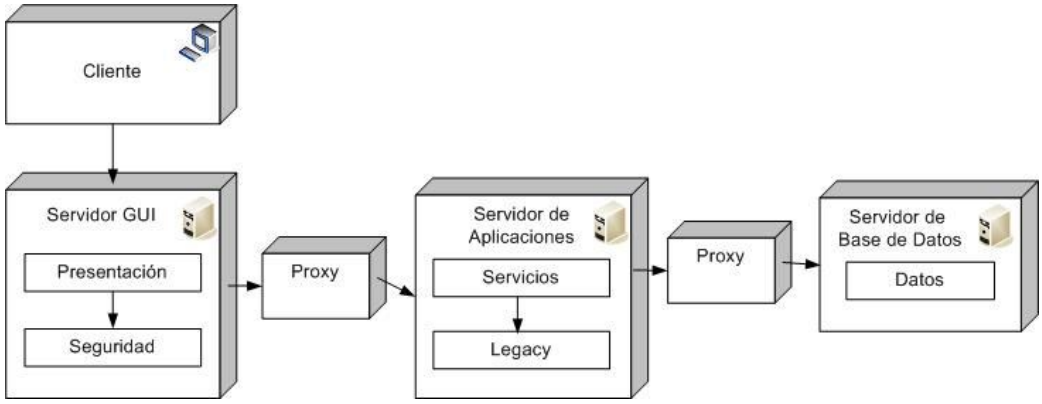
(Fuente: Autor de la Investigación)

Actividad 13: Escoger arquitectura base para la familia.

En esta actividad ya se tiene una evaluación de todas las arquitecturas y se consigue un consenso en cuanto a las opciones que existen entre arquitectura cliente-servidor tres capas y arquitectura Modelo-Vista-Controlador y podemos concluir que existe una arquitectura adecuada a los requisitos de calidad establecidos representada en el artefacto arquitectura base para la línea de producto compuesto de diagramas que representan los componentes y conectores, mostrados en la tabla 37:

Tabla 37. Arquitectura base para la línea de producto.

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



Componente de la Arquitectura

- Componente de presentación: Esta es la capa que interactúa con el usuario final, es la encargada de presentar la información y recolectarla para hacer uso de los servicios expuestos por la capa de servicio, para satisfacer los casos de uso de la aplicación.

-Componente de Seguridad (SSL, autenticación y autorización, y sesión): Se encarga de identificar los usuarios, y permitirles utilizar y manipular la información adecuada. Este componente esta subdividido en tres módulos, el primero se encarga del cifrado y descifrado de los datos (SSL), el segundo del proceso de autenticación y autorización de los usuarios para revisar si poseen permiso o no para entrar a la aplicación y el tercero el proceso verificación de sesiones.

-Componente de datos: Este componente esta conformado por la Base de Datos y el componente administrador de la base de datos. La primera contiene los datos relacionados a las transacciones realizadas. El componente Administrador de la Base de Datos es el encargado de realizar las consultas y modificaciones a la información almacenada en la base de datos. La base de datos utilizada será una base de datos relacional.

-Componente de servicios: se dividen en dos tipo: servicios de utilidad y servicios del sistema transaccional de Recurso Humano. Entre los servicios de utilidad esta la administración del sistema e integración con sistemas externos. Los servicios del Sistema Transaccional implementan comportamientos del negocio soportado por la aplicación y se apoya sobre los servicios de utilidad, de esta forma las personalizaciones del modelo de negocio consistirían en modificar o establecer parámetros para estos servicios.

-Componente Legacy: permite la importación y exportación de datos de la aplicación, para su integración con otros sistemas como sistema de presupuesto es un sistema donde asocian los códigos presupuestarios dados por el gobierno con los gastos del personal de la institución con la finalidad de llevar detalladamente los gastos realizados por la institución.

(Fuente: Autor de la Investigación)

Finalmente, presenta los resultados de la escogencia de la arquitectura base para la familia, generando el artefacto informe sobre decisión arquitectural, definido en la tabla 38:

Tabla 38. Informe sobre decisión arquitectural

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	*Informe sobre decisión arquitectural*
Constructor:	Arquitecto de software
Nro. Identificación ACTIVIDAD:	A_13
Nro. Identificación ARTEFACTO:	20
La ventaja principal de aplicar esta arquitectura es que el desarrollo se lleva a cabo en varios niveles y en caso que sobrevenga un cambio se ataca el nivel requerido sin tener que revisar entre código mezclado y siendo transparente al usuario final. Otro beneficio es que al estar separado en capas la lógica del diseño es más escalable y segura a posibles ataques, ya que se puede colocar firewall entre las diferentes capas y solo poseen una sola vía de comunicación entre los componentes. De esta forma se dispone de un único punto de acceso a la base de datos, de forma que todas las operaciones contra ellas se ejecutaran desde un único punto, la cual se puede controlar mejor las transacciones realizadas. Se presentan algunas plataformas que soportan la implementación de la arquitectura escogida: Framework MVC para php tales como Symfony, Cakephp, entre otros. Para Java / J2ee esta Struts, Spring, y las soluciones como los framework de Microsoft basado .NET.	

(Fuente: Autor de la Investigación)

Actividades de la Implementación del Dominio.

Actividad 14: Especificación del componente.

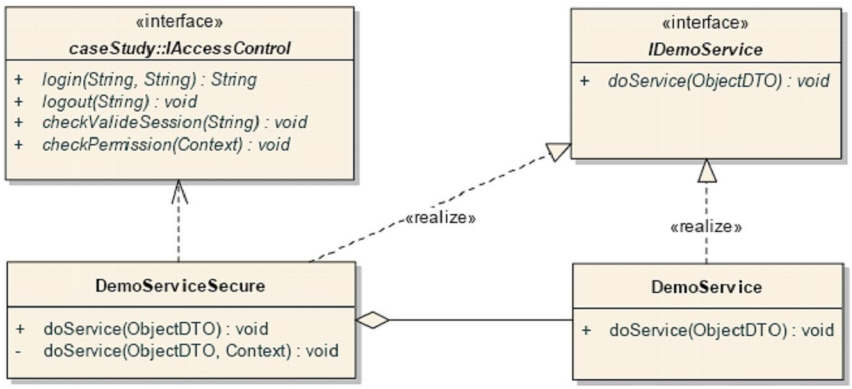
Utilizaremos el Componente de Autorización y Autenticación del Paquete de Seguridad para la actividad especificación del componente se obtiene el artefacto de

Documento de Especificación formal del Componente en donde se realiza de manera técnica la funcionalidad esperada por el componente, sus interfaces y operaciones con sus parámetros de entrada, parámetros de salida, restricciones, pre y post condiciones, todo con ayuda del modelado UML 2.0 y las extensiones que provee. Además, en este documentos se incluyen el contrato de uso y contrato de realización del componente. En conclusión la Especificación formal del Componente viene dado por la tabla 39:

Tabla 39. Especificación formal del Componente.

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	*Especificación formal del Componente*
Constructor:	Diseñador de componentes
Nro. Identificación ACTIVIDAD:	A_14
Nro. Identificación ARTEFACTO:	21

Formato Asociado:
Diagrama del Componente de Autorización y Autenticación y sus conexiones:



Contrato de Uso	
Nombre del Componente	CAccessControl
Interfaces Soportadas	IAccessControl
Signatura	Login(usuario:string,clave:string) Logout(idsesion:string) checkValidateSession(idsesion:string) checkPermission(idusuario:string)
Pre-condición	Usuario: cualquier combinación de números y

	letras Clave: cualquier combinación de números y letras
Post-condición	Emitirá un valor combinado de número y letras que identifica el Id de la sesión, si no emitirá un mensaje determinado de Acceso Denegado
Restricciones	Para poder acceder debe colocarse la combinación correcta del usuario y la clave
Comportamiento	Espera la ejecución del usuario y devuelve el resultado según los valores de entradas.
Invariantes	El componente presentara los siguientes estados: tomando los datos, validando los datos y permitiendo el acceso.

En el contrato de realización se colocan el nombre, la descripción y las condiciones que deben realizar y cumplir todas las funciones que contenga el componente.

Nro de Identificación de la funcion	Función Exportada del Componente	Pre-condición	Post-condición
1	Login(usuario:string,clave: string)	NOT NULL	Validara si el usuario con la clave es correcta o no. Emitirá un valor combinado de número y letras que identifica el Id de la sesión, si no emitirá un mensaje determinado de Acceso Denegado
2	Logout(idsesion:string)	NOT NULL	Eliminara el valor del Identificador de la Sesión Devuelve Vacio y redirecciona a la pantalla de validación
3	checkValidateSession(idsesion:string)	NOT NULL	Valida si el valor del Identificador de la Sesión esta activo. Devuelve Vacio y redirecciona a la pantalla de validación
4	checkPermission(idusuario:string)	NOT NULL	Revisa si el usuario tiene permiso o no al servicio que esta solicitando Devuelve Vacio y redirecciona a la

			servicio principal

(Fuente: Autor de la Investigación)

Para el artefacto 22, Aceptación de la Plataforma Tecnológica se deben especificar los ambientes de corridas del componente, paquetes necesarios, etc., es decir, la plataforma donde puede ejecutarse el componente, tomando en consideración las recomendaciones realizadas en el Informe sobre decisión arquitectural. El resumen del presente artefacto se muestra en la tabla 40.

Tabla 40: Aceptación de la Plataforma tecnológica.

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	*Aceptación de la Plataforma Tecnológica*
Constructor:	Experiencia del Diseñador de componentes
Nro. Identificación ACTIVIDAD:	A_14
Nro. Identificación ARTEFACTO:	22

Formato asociado: se presentan el cuadro de evaluación para extraer la plataforma que más se ajusta a las funcionalidades de los componentes.

Características	**Definición**	**Puntaje**
No Soporta	No apoya el requisito	1
Poco Soporte	De cierta manera se puede satisfacer el requisito o parte de él.	2
Fuerte Soporte	Este requisito aparece explícitamente en la descripción de la plataforma o en su manual.	3

Criterios de evaluación (Funcionalidades del Componente)	**Framework PHP (Cakephp)**	**Framework JAVA (Struts)**	**Framework .NET**
Login(usuario:string,clave:string)	3	3	3
Logout(idsesion:string)	3	3	3
checkValidateSession(idsesio	3	3	3

n:string)			
checkPermission(idusuario:string)	3	3	3
Total	12	12	12

Como vemos en la tabla de comparación, las plataformas sugeridas pueden cumplir con el desarrollo y puesta en producción del componente. Se descartara la plataforma .NET, ya que no cumple con una de las reglas de negocio, que el desarrollo debe realizarle bajo Software libre. Se seleccionara el Framework Cakephp desarrollado bajo lenguaje php, debido a que la mayoría de los programadores en tienen conocimiento en este lenguaje de programación.

(Fuente: Autor de la Investigación)

Actividad 15: Aprovisionamiento del Componente.

En esta actividad aprovisionamiento del componente se buscara el componente en los repositorios externos, internos o libres. Luego se seleccionara el tipo de aprovisionamiento según los nombrados por (Hammar y Montilva, 2003), que son: Adquirir el componente, Suscribir el componente, Adaptar el componente, Envolver el componente y Desarrollar un nuevo componente.

En la tabla 41, se muestra el artefacto Componente Seleccionado, este documento debe contener todo lo relacionado al aprovisionamiento del componente, y variara, dependiendo de la forma del aprovisionamiento, si se desarrolla uno nuevo por ejemplo, debe estar incluido el diseño del componente.

Tabla 41: Componente Seleccionado.

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	*Componente Seleccionado*
Constructor:	Diseñador de componentes
Nro. Identificación ACTIVIDAD:	A_15
Nro. Identificación ARTEFACTO:	23
Se busco, siguiendo la información la especificación del Documento de Especificación formal del Componente y el Documento de Aceptación de la Plataforma Tecnológica, en diferentes fuentes como: • Sistemas legados: Existen componente de seguridad para Visual Basic, Delphi, jsp,etc.	

Pero no cumple con el requisito de que pueda ejecutarse en el lenguaje de programación php.

- Suscribir el componente: En este caso la suscripción no es una opción debido a que no se maneja recursos financieros.
- Repositorio Interno: Este caso no es factible debido a que no se tiene ningún componente de este tipo en el repositorio interno.
- Repositorio Externo: En este caso se tomo como repositorio externo Internet, siendo de especial interés aquellos sitios de fuente abierta y que se especialicen en componente programados en php. Con la realización de esta búsqueda se lograron encontrar componentes factibles en las siguientes direcciones:
  - PhpClases: http://www.phpclasses.org/browse/class/78.html
  - Cakephp: http://book.cakephp.org/view/1296/Security-Component

Se selecciono el componente de cakephp, debido a que de fuente abierta y esta desarrollado de forma modular y es más fácil de entender. Aunque posee algunas funcionalidades que no son requeridas, se procederá a adaptarlo a las necesidades particulares.

(Fuente: Autor de la Investigación)

Actividad 16: Pruebas del Componente.

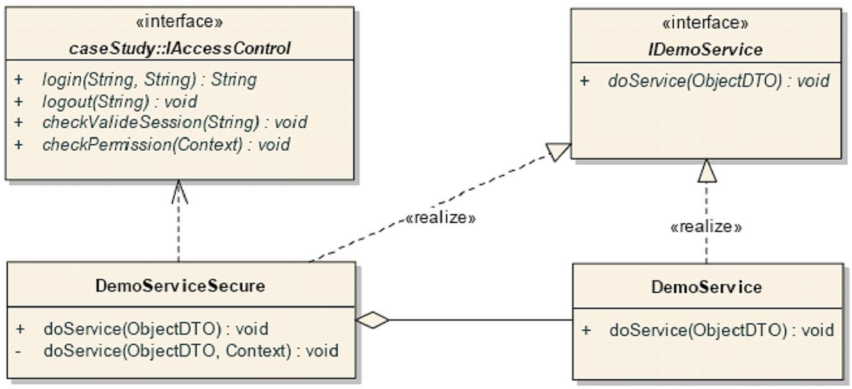
En esta actividad Pruebas del Componente se realiza la revisión de las funcionalidades del componente que provienen del artefacto *Especificación formal del Componente* y *Aceptación de la Plataforma Tecnológicas* con las funciones registrada en el artefacto *Componente Seleccionado* con el fin de verificar que: el componente sea consistente en su comportamiento, las dependencias del componente estén bien especificadas y la configuración del componente puede ser realizada según lo indicado en la documentación. Seguidamente, se diseña el plan de pruebas del componente con las pruebas a realizar y los resultados esperados. El cuadro de verificación del componente y plan de pruebas mencionados anteriormente, constituye el *Componente Probado*.

En consecuencia, se muestra a continuación el artefacto generado de la actividad *Prueba del Componente*. El artefacto *Componente Probado,* es el documento donde se describe las funcionalidades y características, ya verificado, asociado con el plan de pruebas a realizar a la hora de desarrollado del componente y el diseño en UML del componente.

Tabla 42: Componente Probado.

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	*Componente Probado*
Constructor:	Experiencia del Diseñador de componentes
Nro. Identificación ACTIVIDAD:	A_16
Nro. Identificación ARTEFACTO:	24

Descripción del componente ya verificado:



Verificación de Componente

Caso de Pruebas	Detalles	Cumple (Si o No)
Verificar Consistencia	El componente es consistente con los contratos de uso y realización del artefacto Especificación formal del Componente.	Si
Verificar dependencias	Las dependencia con el componente de servicios está presente.	Si
Verificar configuración	El componente se instala con facilidad	Si, necesita estar instalado el php y el apache.
Compatibilidad entre las versiones	No existe versiones anteriores	Si, por ser la primera versión.

El plan de Prueba a realizar el componente es el siguiente:
Pruebas funcionales

Caso de Pruebas	Pruebas a realizar	Resultados Esperado
Prueba de las Funciones	Entrada por teclado Verificación de Usuario Valido Verificación de Usuario No Valido	Letra escrita por pantalla Entrada del usuario al sistema Mensaje de Acceso denegado
Prueba de interfaces: como es una sola interfaz se utilizan los mismos casos		

Pruebas De Comportamiento

Caso de Pruebas	Pruebas a realizar	Resultados Esperado
Requerimientos no funcionales	Usuarios y clave vacía	Mensaje de Acceso denegado
Prueba Confiabilidad	Verificación de Usuario Valido Verificación de Usuario No Valido	Generación de Identificación de la sesión Entrada del usuario al sistema Mensaje de Acceso denegado
Pruebas de configuración	Pruebas de ejecución en plataforma Linux y Windows	El mismo comportamiento en los dos sistemas
Pruebas de Recuperación	Como el componente no guarda datos, no debe haber la posibilidad de recuperar datos.	Ninguno en especifico

Pruebas de Aceptación

Caso de Pruebas	Pruebas a realizar	Resultados Esperado
Prueba de Aceptación por el grupos	Puesta a prueba en conjunción con los componentes con los cuales debe operar.	Aceptación del Componente

(Fuente: Autor de la Investigación)

Actividad 17: Liberación del Componente.

Para finalizar, se procede a realizar la actividad liberación del componente, en donde se cataloga el componente, con la finalidad de ser colocado en repositorio interno u externo para que puedan ser accedidos por los desarrolladores. Luego se procede a la publicación de la especificación del componente, conjuntamente con los resultados de las pruebas del componente y, a medida que se utiliza el componente, los resultados del uso obtenidos por los usuarios externos.

Tabla 43: Clasificación del componente.

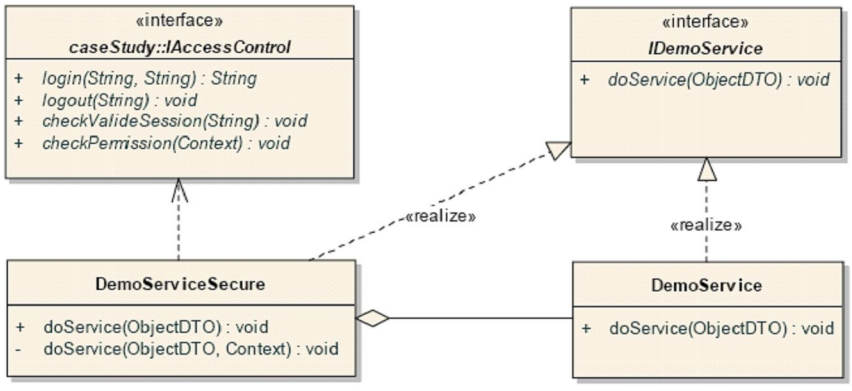
InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	*Clasificación del componente*
Constructor:	Diseñador de componentes
Nro. Identificación ACTIVIDAD:	A_17
Nro. Identificación ARTEFACTO:	25

Sección Clasificación : Versión 1

Se encarga de identificar los usuarios, y permitirles utilizar y manipular la información adecuada. El proceso de autenticación y autorización de los usuarios revisa si los usuarios poseen permiso o no para entrar a la aplicación y el tercero el proceso verificación de sesiones.

Sección Soluciones

Diagrama de Componente



Contrato de Uso	
Nombre	CAccessControl
Interfaces Soportadas	IAccessControl
Signatura	Login(usuario:string,clave:string) Logout(idsesion:string) checkValidateSession(idsesion:string) checkPermission(idusuario:string)
Pre-condición	Usuario: cualquier combinación de números y letras Clave: cualquier combinación de números y letras
Post-condición	Emitirá un valor combinado de número y letras que identifica el Id de la sesión, si no emitirá un mensaje determinado de Acceso Denegado
Restricciones	Para poder acceder debe colocarse la combinación correcta del usuario y la clave
Comportamiento	Espera la ejecución del usuario y devuelve el resultado según los valores de entradas.
Invariantes	El componente presentara los siguientes estados: tomando los datos, validando los datos y permitiendo el acceso.

En el contrato de realización se colocan el nombre, la descripción y las condiciones que deben realizar y cumplir todas las funciones que contenga el componente.

Nro de Identificación de la funcion	Función Exportada del Componente	Pre-condición	Post-condición
1	Login(usuario:string,clave:string)	NOT NULL	Validara si el usuario con la clave es correcta o no. Emitirá un valor combinado de número y letras que identifica el Id de la sesión, si no emitirá un mensaje determinado de Acceso Denegado
2	Logout(idsesion:string)	NOT NULL	Eliminara el valor del Identificador de la Sesión Devuelve Vacio y redirecciona a la pantalla de validación
3	checkValidateSession(idsesion:string)	NOT NULL	Valida si el valor del Identificador de la Sesión esta activo.

			Devuelve Vacio y redirecciona a la pantalla de validación
4	checkPermission(idusuario :string)	NOT NULL	Revisa si el usuario tiene permiso o no al servicio que esta solicitando Devuelve Vacio y redirecciona a la servicio principal

Pruebas realizadas

Nro Revisión	Resultado de revisión
	Según la funcionalidad deseada, descrito en el artefacto Arquitectura Base para línea de Producto, en el cual el componente de autenticación y verificación debe revisar si el usuario posee autorización o no para entrar a las aplicaciones o servicios. Por otra parte en el artefacto Documento de Aceptación de la Plataforma Tecnológicas, se decidió que la plataforma a utilizar el framework Cakephp desarrollado en el lenguaje php. Revisando el contrato de realización y lo antes planteado, podemos concluir que el componente satisface los requerimientos iniciales planteados en el artefacto Arquitectura Base para línea de Producto.

El plan de Prueba a realizar el componente es el siguiente:

Pruebas funcionales

Caso de Pruebas	Pruebas a realizar	Resultados Esperado
Prueba de las Funciones	Entrada por teclado Verificación de Usuario Valido Verificación de Usuario No Valido	Letra escrita por pantalla Entrada del usuario al sistema Mensaje de Acceso denegado
Prueba de interfaces: como es una sola interfaz se utilizan los mismos casos		

Pruebas De Comportamiento

Caso de Pruebas	Pruebas a realizar	Resultados Esperado
Requerimientos no funcionales	Usuarios y clave vacía	Mensaje de Acceso denegado

Prueba Confiabilidad	Verificación de Usuario Valido Verificación de Usuario No Valido	Generación de Identificación de la sesión Entrada del usuario al sistema Mensaje de Acceso denegado
Pruebas de configuración	Pruebas de ejecución en plataforma Linux y Windows	El mismo comportamiento en los dos sistemas
Pruebas de Recuperación	Como el componente no guarda datos, no debe haber la posibilidad de recuperar datos.	Ninguno en especifico

Pruebas de Aceptación

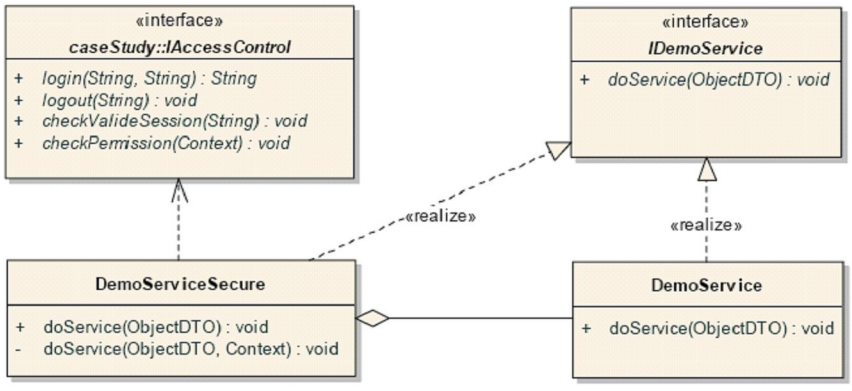
Caso de Pruebas	Pruebas a realizar	Resultados Esperado
Prueba de Aceptación por el grupos	Puesta a prueba en conjunción con los componentes con los cuales debe operar.	Aceptación del Componente

Sección de Usos

Según el artefacto Aceptación de Plataforma tecnológica, el componente debe estar instalado el lenguaje de programación Php y el servidor en Apache 2.0. Estas herramientas pueden ejecutarse tanto en sistemas operativo Linux y Windows.

Sección de Relaciones con los Componentes

Diagrama de Componente



Según el diagrama de componente esta enlazado con el componente de servicios.

(Fuente: Autor de la Investigación)

Tabla 44: Documento de Publicación del componente.

InDoCaS	
ARTEFACTO	**DESCRIPCIÓN**
Nombre:	*Documento de publicación del componente (DPC)*
Constructor:	Diseñador de componentes
Nro. Identificación ACTIVIDAD:	A_17
Nro. Identificación ARTEFACTO:	26
Formato Asociado La selección de la forma de publicación del componente, se realizara cuando la Coordinacion Nacional de Tecnología de Información de la UNEXPO, crea su repositorio.	

(Fuente: Autor de la Investigación)

Se ha presentado una propuesta de línea de producción de software para sistemas transaccionales para la Coordinación Nacional Tecnología de Información de la UNEXPO y se completa las disciplinas de análisis, diseño e implementación del dominio del proceso de Ingeniería de dominio propuesto por el **I**n**D**o**C**a**S** **E**xpandido.

La línea de producción de software para sistemas transaccionales propuesto puede ser utilizada como base para el diseño y construcción de aplicaciones transaccionales en las universidades públicas venezolanas. Adicionalmente, el proceso de **I**n**D**o**C**a**S** **E**xpandido, podría ser utilizado para el diseño y construcción de líneas de producción de software en otro dominio, tanto en el área académica como el área industrial.

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

Después de realizar la implementación de una línea de producción de software para sistemas transaccionales y realizar un caso de estudio a la Coordinación Nacional de Tecnología de Información de la UNEXPO, se llegó a las conclusiones siguientes:

1. El presente estudio se centró en la disciplina ingeniería de dominio, así mismo, se utilizó el proceso general denominado **I**n**D**o**C**a**S**, el cual se expandió incorporando la disciplina de implementación del dominio. El proceso **I**n**D**o**C**a**S** **E**xtendido se utilizó para llevar a cabo las actividades que componen la línea de producción de software para sistemas transaccionales utilizando como caso de estudio la Coordinación Nacional de Tecnología de información de la UNEXPO.

2. Se hizo necesaria la incorporación de las actividades correspondiente a la implementación del dominio, instanciando el método WATCH – Component, para crear el proceso **I**n**D**o**C**a**S** **E**xtendido, siendo este un aporte importante. Las actividades agregadas son: "Especificación del componente", "Aprovisionamiento del componente", "Pruebas del Componente" y "Liberación del Componente" y los artefactos son: "Especificación formal del Componente", "Aceptación de la Plataforma Tecnológica", "Componente Seleccionado", "Componente Probado", "Clasificación del Componente" y "Documento de publicación del componente".

3. Los componentes básicos obtenidos de la arquitectura de software para los sistemas transaccionales son los siguientes: Componente de Presentación,

Componente de Seguridad, Componente de Datos, Componentes de Servicios y Componente Legacy.

4. El modelo de proceso **I**n**D**o**C**a**S** permite construir el modelo de calidad desde la disciplina de análisis del dominio, es decir, se formaliza la especificación de los requisitos de calidad y determina las características no funcionales. En los modelos existente no se incorpora en forma implícita el modelo de calidad asociado al dominio, por esta razón ha sido tomado como proceso base para la construcción del proceso **I**n**D**o**C**a**S E**xpandido.

5. El modelo de proceso de **I**n**D**o**C**a**S E**xpandido, podría ser utilizado para el diseño y construcción de líneas de producción de software en otro dominio, tanto en el área académica como el área industrial e instituciones universitarias.

6. Para que un sistema puede ser catalogado como sistema transaccional debe sus operaciones cumplir con las propiedades de las transacciones ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad). Por otra parte, son los primeros sistemas que se implementan en las organizaciones, ya que operan sobre las base de la operaciones de una empresa, por tal razón, estos sistemas deber ser rápidos, fiables e inflexibles con la información que captan.

7. Los sistemas transaccionales operan sobre la parte operativa de una empresa, ya que se encargan de llevar los datos relacionados a inventarios, nominas, fabricación, contabilidad, entre otros. Debido a la gran cantidad de datos e importancia en la organización, un fallo de estas aplicaciones, pueden influir negativamente en el rendimiento de la empresa. Por tal motivo, es importante que un mantenimiento o desarrollo de nuevas funcionalidades deben realizarse bajo proceso de desarrollo de software como el enfoque de líneas de producción de software.

8. Los beneficios asociados a las líneas de producción como reducción en tiempo de entrega del producto, reducción de costos, crecimiento controlado de las aplicaciones, reducción de las tasas de defectos y mejoras en calidad de los productos, pueden ser trasladado a los sistemas transaccionales.

Recomendaciones

Basándose en la investigación realizada, observaciones y datos recabados, se presentan las siguientes recomendaciones:

1. Este enfoque puede ser generalizado para generar y/o proponer líneas de producción de software orientado a sistemas transaccionales tanto en el área académica como en el mercado comercial.

2. Requiere incorporar al proceso **I**n**D**o**C**a**S** **E**xpandido las actividades de la ingeniería de la aplicación, desarrollando las actividades y artefactos necesarios incorporándolo bajo la nomenclatura del proceso **I**n**D**o**C**a**SE**.

3. Soportar a **I**n**D**o**C**a**S** **E**xpandido como en todo modelo de procesos, de herramientas que faciliten la trazabilidad y guía de las diversas actividades y artefactos que componen el mismo.

REFERENCIAS BIBLIOGRÁFICAS

Altintas I., Surav M., Keskin O., Cetin S. (2005). Aurora Software Product Line. Cybersoft Information Technologies Co. Turquia.

Balestrini, M. (2005). Cómo se elabora el proyecto de investigación. 3ª edición. Caracas: B.L Consultores.

Bass L., Clements P., Kazman R.(2003). "Software Architecture in Practice". SEI Series in Software Engineering. Pearson Education. Boston.

Bernstein P.,Newcomer E. (2009). Principles of Transaction Processing. Segunda Edición. Morgan Kaufman.

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

Canelón R., Losavio F., Matteo A., Chirinos L.(2009).Modelo conceptual para modelación de aplicaciones móviles sensibles al contexto. Rev. Fac. Ing. UCV v.24 n.2 Caracas. http://www.scielo.org.ve/scielo.php?pid=S0798-40652009000200010&script=sci_arttext (Consultada:Noviembre 22, 2010).

Canelón R., Gómez E., Rivero L. (2008). Interfaces Dinámicas en Dispositivos Móviles. Publicaciones en Ciencias y Tecnología, UCLA, Volumen II. Barquisimeto.

Clements P., Northrop L.(2001). "Software Product Lines. Practices and Patterns". SEI Series in Software Engineering. Addison Wesley. Segunda Edición. Boston, USA. pp. 29-31.

Chirinos, L., Losavio, F., Matteo, A. (2004). Identifying Quality-Based Requirements. Information Systems Management. Editorial: Auerbach Publications, January.

Deming, W. (1989). Calidad, Productividad y competitividad. La salida de la crisis. Diaz de Santos. Madrid.

Díaz Pérez M., Liz Contreras, Y., Rivero S. (2009). Características de los sistemas de información que permiten la gestión oportuna de la información y el conocimiento institucional. ACIMED. Pag 66-71.

Diccionario de Informática. http://www.alegsa.com.ar/Dic/sistema%20transaccional.php (Consultada:Julio 16, 2010).

Domínguez K., Pérez M., Mendoza L. y Grimán A.(2005). HACIA UNA ONTOLOGÍA PARA FÁBRICAS DE SOFTWARE. V Workshop de Ingeniería del Software - Jornadas Chilenas de Computación 2005, Valdivia, Chile.

Dorman, D., Boyd L. (1997). Object Models for Business Transaction Processing Systems. OOPSLA Workshop1997. http://st-www. cs.illinois.edu /users/Johnson /business-transactions/dorman.html. Consultada(Julio 20,2010)

Durán H., Meda M., Sapien A., Piñón L. (2008). Un marco de trabajo de una fábrica de software para el reuso del diseño arquitectónico y de componentes de software. Tecnociencias. Volumen II Nro 1 Enero-Abril 2008. Chihuahua. México.

Fábregas, J. y Bauza, J. (1991). Administración de Proyectos. Editorial Miro C.A. Venezuela.

García Peñalvo F., Barras J., Laguna Serrano M. y Marqués Corral J. (2002). Líneas de Productos, Componentes, Frameworks y Mecanos. Departamento de Informática y Automática. Universidad de Salamanca.

Goldsmith, R. (2007). REAL CHAOS, Two Wrongs May Make a Right. Computer Aid., Inc. Estados Unidos.

Hamar V. y Montilva J. (2003). Aspectos Metodológicos del Desarrollo y reutilización de Componentes de Software. Tesis de la Universidad de Los Andes. Mérida.

Hetrick W., Krueger C. y Moore J. (2006). Incremental Return on Incremental Investment: Engenio's Transition to Software Product Line Practice. BigLever Software & Engenio Information Technologies.

Hernández R., C. Baptista P. (2003). Metodología de la Investigación. Tercera Edición. México: McGraw Hill Interamericana de México, S.A. de C.V.

ISO/IEC 25010 (2007). Software Engineering–Software Product Quality Requirements and Evaluation (SQuaRE) – Quality model and guide.

ISO/IEC 9126-1 (2001). Software Engineering – product quality part 1.Quality model.

Juran J. (1990). Juran y el liderazgo para la calidad. Díaz de Santos.

Krueger, C. (2006). Introduction to the Emerging Practice Software Product Line Development. METHODS & TOOLS. Fall 2006 (Volume 14 - number 3).

Krueger, C. (2006). New Methods Behind a New Generation of Software Product Line Successes. BigLever Software, Inc.

Krueger, C. (2006). Introduction to Software Product Lines. BigLever Software, Inc. http://www.softwareproductlines.com/introduction/introduction.html (Consultada: Julio 17, 2010)

Laudon, J. y K. (2006). Sistemas de Información gerencial – Administración de la empresa dígital. Pearson Educación- Prentice Hall.

Lee A.S., Baskerville R.L. y Davies, L A. (1992). Workshop on two techniques for qualitative data analysis: Actino Research and Ethography. Proceeding of The Thirteen International Conference on Information Systems.

Montaldo, F. (2005). "Patrones de Diseño de Arquitecturas de Software Enterprise". Tesis de Grado. Departamento de Computación, Facultad de Ingeniería, Universidad de Buenos Aires. Argentina.

Montilva, Jonas (2006). Desarrollo de Software Basado en Líneas de Producción de Software. IEEE Computer Society Región 9 Capítulo Argentina Programa DVP.

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

Montilva J., Hazam K. y Gharawi J.(2000). The Watch Model for development business software in small and midsize organization. In IV World Multiconference on Systemics, Cybernetics and Informatics (SCI`2000). Orlando, Florida (USA).

Montilva J., Barrios J. y Rivero M. (2008). GRAY WATCH MÉTODO DE DESARROLLO DE SOFTWARE PARA APLICACIONES EMPRESARIALES. Proyecto METHODIUS. FONACIT 2005000165. Mérida.

Moreno Quinteros J., Varese Isern I., Joyanes Aguilar, L. (2003). Modelo transaccional para aplicaciones de workflow. II Simposio Internacional de Sistemas de Información e Ingeniería de Software en la Sociedad del Conocimiento. (SISOFT 2003), Madrid, España.

Morisawa Y., Torii K.. (2001[a]). A Practical Method to Select an Architectural Style of Product Lines for Distributed Processing Systems. The 2001 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'2001), Las Vegas.

Morisawa Y., Torii K. (2001[b]). Architectural Styles for Distributed Processing Systems and Practical Selection Method. Proceedings of the 8th European software engineering conference held jointly with 9th ACM SIGSOFT international symposium on Foundations of software engineering. Viena. Austria.

Northrop L., Clements P. (2007). A Framework for Software Product Line Practice, Version 5.0. Software Engineering Institute y Carnegie Mellon University. Pittsburgh.

Northrop, L. (2008). Software Product Lines Essentials. Software Engineering Institute y Carnegie Mellon University. Pittsburgh.

Papazoglou M. (2003). Web Services and Business Transactions. Springer Netherlands. Volume 6, Number 1. marzo de 2003.

Pohl K., Böckle G. y Van Der Linden F. (2005). Software Product Line Engineering. Springer-Verlag. Berlín.

Rivera, A. (2008). Sistema asistente para la generación de horarios de cursos. Universidad de las Américas Puebla. México.

Rivero D., Montilva J., Granados G., Barrios J., Besembel I. y Sandia B. (2007) La Industria de Software en Venezuela: Una Caracterización de su Recurso Humano. Primer Encuentro Venezolano sobre Tecnologías de la Información e Ingeniería del Software.

Ruiz Bolívar, C. (2007). Instrumentos de Investigación Educativa. Procedimientos para su diseño y validación. 3era reimpresión. Barquisimeto: Ediciones CIDEG,c.a.

Shapiro J. (2008). Planificación estratégica. http://www.civicus.org/recursos-y-publicaciones#Herramientas. (Consultada: Julio 17, 2010)

Sommerville I. (2006). Ingeniería del Software. Séptima Edición. Pearson Educación S.A. Madrid.(P.06)

Universidad Nacional Experimental Politécnica “Antonio José de Sucre” (UNEXPO), (2005). Resolución C.U. Nº 2005-E09-05. P. 26.

Universidad Pedagógica Experimental Libertador (2006). Manual de Trabajos de Grados de Especialización, Maestría y Tesis Doctorales. Caracas.

Urbina A. (2009). SISTEMA DE INVENTARIO DE OBRAS Y ACTIVIDADES DEL PROYECTO NACIONAL DE GASIFICACIÓN POR METANO Y GLP PARA LA GERENCIA DE DESARROLLO SOCIAL DE PDVSA GAS COMUNAL. Universidad Nacional Experimental del Táchira.

Wiegers, K. (1999). Process Improvement that Works, Software Development Magazine, 7(10).

A. CURRÍCULO VITAE DEL AUTOR

Leiban Alberty Rivero Colmenarez

Cursante del Postgrado en Ciencias de la Computación Mención Ingeniería de software de la Uiversidad Centroccidental "Lisandro Alvarado". Nació en Barquisimeto, Estado Lara, el 04 de Abril de 1982. Realizó estudios de Educación primaria en la Escuela Básica "San Ignacio de Loyola", Barquisimeto-Venezuela. Seguidamente cursó sus estudios de Educación Secundaria y Diversificada en el Liceo "José María Domínguez", Barquisimeto-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: **PHP Avanzado** avalado por Formar Venezuela en May. 2007, **Ingles** avalado por Fundación Universidad de Carabobo (FUNDAUC) en Ene. 2004. **Introducción a Oracle 8i** avalado por Universidad Centroccidental Lisandro Alvarado (UCLA) en Jun. 2004, **Mantenimiento de Equipos de Computación** avalado por Universidad Centroccidental Lisandro Alvarado (UCLA). Jun. 2004, **Mantenimiento y Reparación de PC** avalado por Instituto: Centro de Computación I.L.C.A. en (2004), **Análisis, Programación y Diseño de Sistema (Nivel I)** avalado por Instituto: Instrucción Especializada en Computación (INESCO), **Análisis, Programación y Diseño de Sistema(Nivel II)** avalado por Instituto Instrucción Especializada en Computación (INESCO)**, Lenguaje de Programación de Visual Basic 6.0** avalado por Centro de Computación de la Universidad Centro Occidental Lisandro Alvarado (UCLA) 2004**.**
Ha obtenido reconocimientos por la excelencia académica, ***Cuadro de Honor UCLA.*** Universidad Centroccidental Lisandro Alvarado (UCLA).Decanato de Ciencias y Tecnología (DCYT). Barquisimeto Estado Lara ***Lapso 2003-II, Lapso 2002-I,Lapso 2001-I, Lapso 2000-II, Lapso 2000-I.*** Por haber obtenido la calificación sobresaliente de 19 puntos en la asignatura estructuras Discretas I en la carrera Ingenieria en Informática en el Lapso 00-I. Por haber ganado el primer como mejor proyecto en el decanato de ciencias y tecnología y como mejor en el área de conocimiento en las II Jornadas Estudiantiles de Investigación y Extensión de la UCLA. Barquisimeto Noviembre de 2004. Por la destacada Participación en las II Jornadas Estudiantiles de Investigación y Extensión de la UCLA. Febrero – 2005, por la Presentación de Proyectos Estudiantiles del Área de Sistemas.Universidad Centroccidental Lisandro Alvarado (UCLA).Decanato de Ciencias y Tecnología (DCYT). Posee experiencia laboral como Preparador de Sistemas en el Centro de Computación del Decanato de Ciencias y Tecnología. Universidad Centrooccidental Lisandro Alvarado (UCLA) desde Feb. 2004 a Dic. 2005, como Analista Programador en Sistema en AUDICONSULT S.A. desde Mar. 2006 al May. 2006, actualmente Analista Programador en Sistema en la Coordinación Nacional de Tecnología de Información (CNTI) de la Universidad Nacional Experimental Politécnica "Antonio José de Sucre" (UNEXPO), desde Jun. 2006 a la actualidad.
.