UNIVERSIDAD CENTROCCIDENTAL

"LISANDRO ALVARADO".

EXTENSION DEL METODO GRAY WATCH BASADO EN EL ESTANDAR DE CALIDAD ISO/IEC 25010

ILIANA CRISTINA SANCHEZ

Barquisimeto, 2011.

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

EXTENSION DEL METODO GRAY WATCH BASADO EN EL ESTANDAR DE CALIDAD ISO/IEC 25010

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

Por: ILIANA CRISTINA SANCHEZ

Barquisimeto, 2011.

EXTENSION DEL METODO GRAY WATCH BASADO EN EL ESTANDAR DE CALIDAD ISO/IEC 25010

Por: ILIANA CRISTINA SANCHEZ

Trabajo de grado aprobado

Jorge Luis Pérez Medina	**Rodolfo Canelón Osal**
Tutor	

Edgar González

Barquisimeto, 18 de Noviembre de 2011.

DEDICATORIA

Al Dios Todopoderoso y supremo, a ti sea toda la gloria, el honor y la honra por los siglos de los siglos.

AGRADECIMIENTO

A mi Padre, a mi Señor Jesucristo y al Espíritu Santo de Dios por darme la gracia, la sabiduría, el poder y la gloria para culminar esta meta con gran éxito.

A mi amado esposo Javier, por su paciencia, comprensión y amor incondicional que siempre me ha entregado. Te amo amor!

A mi tía Carmen, a mi tía Giomar y a mi suegra Lucia por siempre impulsarme a culminar esta etapa de mi vida profesional, Dios las bendiga siempre! Las amo!

A mi tutor el Prof. Jorge Luis Pérez por su apoyo incondicional en el momento que más lo requería, por su atención y dedicación al trabajo, mil gracias!!

A mi amiga Carmen Julia por ser siempre una guía y consejera en mi vida profesional, te quiero mucho!

A mi amiga María José, eres especial! Siempre estás en los momentos precisos te quiero amiga!

A mis hermanas Nataly y Adriana, mucho que agradecerles, mis amigas, confidentes, compañeras de oración, siempre unidas! Gracias por todo! Las amo!

A los profesores Rodolfo Canelón, Edgar González y Edison Sira por el apoyo dado en todo momento.

INDICE GENERAL

INDICE DE FIGURAS

INDICE DE CUADROS

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

EXTENSION DEL METODO GRAY WATCH BASADO EN EL ESTANDAR DE CALIDAD ISO/IEC 25010

Autora: Iliana Cristina Sánchez

Tutor: Dr**.** Jorge Luis Pérez Medina

RESUMEN

Los métodos más representativos de la Industria de Software han logrado que la construcción de sistemas grandes y complejos sea más sencilla cada día. Asimismo, la Industria del Software ha buscado alcanzar el aseguramiento de la calidad de estos sistemas para lograr un producto competitivo que cumpla con los requisitos de calidad, y a su vez satisfaga la necesidad de los clientes. No obstante, hablar de calidad del software implica la necesidad de contar con parámetros que permitan establecer los niveles mínimos que un producto de este tipo debe alcanzar para que se considere de calidad. La presente investigación tuvo como finalidad diseñar la extensión del método GRAY WATCH en su más reciente versión propuesta por Montilva et al (2008), específicamente en los procesos técnicos de Análisis y Diseño, a su vez los productos generados en los Procesos de Análisis y Diseño se acoplaron en el Proceso de Implementación. Se utilizó el estándar de calidad del producto ISO/IEC 25010, que establece criterios para la especificación de requisitos de calidad de productos software, sus métricas y su evaluación, e incluye un modelo de calidad que está compuesto por características y sub-características. Para la extensión del método se tomaron el proceso para la Ingeniería de Dominio basado en Calidad de Software denominado **I**n**D**o**C**a**S** (Canelón, 2010) y el proceso **I**n**D**o**C**a**SE** (Rivero, 2011) como metodología para la definición de actividades y productos en los procesos de Análisis, Diseño e Implementación de la Aplicación. Finalmente, cabe destacar que de acuerdo a la naturaleza del estudio, esta investigación se enmarcó en la modalidad de Proyectos Especiales, y fue validado a través de su aplicación en un caso de estudio denominado Programa de Investigación UPEL-IPB.

Palabras Claves: Método GRAY WATCH, Calidad del Software**,** ISO/IEC 25010, **I**n**D**o**C**a**S, I**n**D**o**C**a**SE**

INTRODUCCIÓN

Los actuales métodos y técnicas de Ingeniería de Software han logrado que la construcción de sistemas grandes y complejos sea más sencilla cada día. Asimismo, la industria del software ha buscado alcanzar el aseguramiento de la calidad de estos sistemas, para lograr un producto competitivo que cumpla con los requisitos de calidad, y a su vez satisfacer las necesidades de los clientes con respecto al producto de software. No obstante, hablar de calidad del software implica la necesidad de contar con parámetros que permitan establecer los niveles mínimos que un producto de este tipo debe alcanzar para que se considere de calidad.

Según Veenendaaal, Hendriks, y Vonderen (2007) en la práctica, los parámetros de calidad de los productos de software son frecuentemente descritos en términos generales, en consecuencia es difícil para los ingenieros de prueba verificar la calidad del producto software contra los requisitos, es por ello, que varios autores han creado diversos modelos que permitan validar la calidad del mismo.

En este propósito, Scalone (2006) menciona algunos modelos, el primero es el de McCall, en el año 1977 hizo la propuesta de romper el concepto de calidad en una serie de factores. Esta idea fue seguida por otros muchos autores como Barry Boehm en 1978, quienes intentaron capturar la calidad del producto de software en una colección de características y subcaracterísticas dependientes, las cuales a su vez son conectadas a indicadores y métricas. Así surgen los modelos de calidad constituidos por un conjunto de características y subcaracterísticas dependientes que permiten evaluar el producto software mediante las métricas establecidas.

En este mismo orden de ideas, la Organización Internacional para estándares - ISO (por sus siglas en inglés Internacional Organization for Standardization) y la Comisión Electrónica Internacional - IEC (por sus siglas en inglés, International Electrotechnical Commission) recientemente publicaron un estándar para la evaluación de la calidad del producto software, la norma ISO/IEC 25000, que proporciona una guía para el uso de las nuevas series de estándares internacionales,

llamados Requisitos y Evaluación de Calidad de Productos de Software (SQuaRE). Esta serie establece criterios para la especificación de requisitos de calidad de productos de software, sus métricas y su evaluación. (ISO/IEC 25000, 2005). SQUARE está formada por las siguientes divisiones: División de gestión de calidad (25000), División del modelo de calidad (25010), División de mediciones de calidad (25020), División de requisitos de calidad (25030), División de evaluación de calidad (25040).

Esta norma recomienda el uso de un modelo de calidad, que según Calero et al. (2010), es un conjunto de características de calidad y sus relaciones, acompañado de las técnicas para evaluar tales características. La nueva norma propone ocho características de calidad tales como: funcionalidad, Rendimiento, seguridad, compatibilidad, usabilidad, portabilidad, fiabilidad y mantenibilidad.

Dado lo anterior, la presente investigación tuvo como finalidad incorporar elementos de calidad basados en la norma ISO/IEC 25010 a los procesos de Análisis y Diseño del método GRAY WATCH, a su vez se acoplo el proceso de implementación, con la intención de optimizar este método de desarrollo de software y producir aplicaciones con calidad del producto software.

Estructura del Trabajo

El trabajo de investigación está estructurado en cinco capítulos los cuales se especifican a continuación:

Capítulo I. El problema. La industria del software tiene dos perspectivas de calidad: la calidad del producto en sí y la calidad del proceso para obtenerlo (planes, actividades, tareas, entre otros para desarrollar y mantener el software). Este estudio se centra en la calidad del producto y su aplicación en fases tempranas del desarrollo del software. Se utiliza el método GRAY WATCH como modelo de desarrollo de software y el modelo de Calidad del producto de la ISO.

Capítulo II. Se presenta un marco conceptual de la norma de calidad ISO/IEC 25010 para el producto Software que aplican medidas internas y externas para evaluar la calidad del producto, por otro lado se presentan trabajos que aplican modelos de calidad desde diferentes perspectivas y que permiten evidenciar la importancia de considerar la calidad del producto en las primeras fases de desarrollo.

Capítulo III. En el diseño de la investigación se utilizaron los principios de la ingeniería de métodos, los aspectos metodológicos propuestos por Montilva et al. (2004), el proceso **I**n**D**o**C**a**S** basado en calidad de software propuesto por Canelón (2010) y el proceso **I**n**D**o**C**a**SE** propuesto por Rivero (2011).

Capítulo IV. Presenta la propuesta del estudio, el diseño del método GRAY WATCH extendido basado en el estándar de calidad ISO/IEC 25010, presentando las actividades y artefactos incorporados en los procesos de análisis y diseño de la aplicación.

Capitulo V. Se presenta un conjunto de conclusiones obtenidas en el estudio realizado y algunas recomendaciones para futuras investigaciones. Finalmente se presentan las Referencias Bibliográficas.

CAPITULO I

EL PROBLEMA

Contexto General

Actualmente el término calidad se usa con mucha frecuencia en cualquiera de los ámbitos de la sociedad, principalmente en organizaciones de producción de alimentos, bienes y servicios, no obstante en la industria del software se ha acrecentado recientemente su uso (Rodríguez, 2010). La misión de las compañías de software según O'Regan (2002) es proporcionar productos o servicios novedosos a un precio competitivo a sus clientes, esto requiere de una clara visión del negocio, una cultura de innovación, conocimiento detallado del dominio del negocio y una estrategia de desarrollo del producto. Esto también requiere de un enfoque en la satisfacción del cliente y calidad de software para garantizar que la calidad deseada se basa en el producto de software.

En lo que respecta a calidad, García (2001) expone que las organizaciones desarrolladoras de software quieren ser capaces de producir y entregar software confiable, a tiempo y apegado al presupuesto acordado con el cliente, es por ello que crean planes de gestión de calidad que miden y controlan los procesos de desarrollo.

En este orden de ideas, se puede definir la calidad del software según Pressman (2002) como la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo documentados y con las características implícitas que se esperan de todo software desarrollado profesionalmente.

Esto concuerda con Crosby (1979) que define calidad como conformidad con los requisitos, esto implica que los requisitos deberán expresarse claramente tal que

no puedan ser mal interpretados, y aunado a ello, en el proceso de producción y desarrollo se deben tomar medidas para determinar la conformidad con dichos requisitos. De otro modo Juran uno de los principales y primeros pensadores modernos de calidad (1982) define calidad como idoneidad o adecuación al uso, esta definición además de considerar los requerimientos del cliente incluye sus expectativas, es decir aquellas características que el usuario reconoce que son beneficiosas para él.

Por su parte, Fitzpatrick (1996) establece que la calidad del software es el grado en que un conjunto definido de características deseables son incorporadas a un producto con el fin de mejorar su rendimiento.

De los planteamientos anteriores se deduce que las características deseadas del software son la base de las medidas de calidad, teniendo que la falta de concordancia con los requisitos es una falta de calidad. A su vez se destaca la importancia de que estas características se incorporen al producto desde el principio a través de los requisitos del sistema y por otra parte se evalúen las expectativas del cliente en cuanto al uso del producto (calidad en uso).

Desde otra perspectiva, Canelón (2010) define Calidad de Software como los requisitos que debe cumplir un sistema de software desde el punto de vista estructural o arquitectónico. Para Losavio et al (2009) la calidad del producto se basa en satisfacer las características deseadas del producto, y esto se puede validar a través del diseño de la arquitectura del sistema, que debe responder a los requisitos exigidos por el sistema: funcionales y no funcionales.

Por otra parte Scalone (2006) plantea que a la hora de definir la calidad del software se debe diferenciar entre la calidad del Producto de software y la calidad del Proceso de desarrollo. Actualmente las principales iniciativas de calidad software se orientan a los procesos de desarrollo, es decir a las actividades de administración y gestión de proyectos de desarrollo de software, tales como el CMMI, modelo para la mejora y evaluación de los procesos de desarrollo de productos software, el IEEE 1092-1998 estándar que define los procesos para la validación y verificación del

software, el ISO/IEC 15504 norma para la evaluación del proceso de software, entre otros. (Rodríguez et al. 2010).

En lo que se refiere a la calidad del producto, Chirinos et al (2004) menciona que se han hecho diversos esfuerzos para la especificación de requisitos de calidad, hace referencia al modelo de calidad ISO/IEC 9126-1, el enfoque SQUID (Software Quality in the Development Process y el estándar ISO/IEC 14598-3.

A los efectos de este trabajo, cabe mencionar que la Organización Internacional para Estándares ISO, publicó un nuevo estándar que se centra en el lado de la calidad del producto denominado SQUARE (Software Product Quality Requirements and Evaluations, ISO/IEC 25000). Esta norma establece criterios para la especificación de requisitos de calidad de productos software, sus métricas y su evaluación. Propone un conjunto de ocho características y subcaracterísticas que generan un modelo de calidad que apoya la especificación y evaluación del software desde diferentes perspectivas (ISO/IEC 25010, 2011).

Situación Problemática

Las metodologías definen un conjunto de criterios de desarrollo que guían la forma en que se aplica la Ingeniería del software, si no se sigue ninguna metodología siempre habrá falta de calidad. Para el desarrollo de software es necesario aplicar un enfoque disciplinado y sistemático, esto se logra a través de métodos (metodologías) que son un conjunto de pasos y procedimientos que guían la construcción de un software. Cortes et al. (2006), al respecto define una metodología de desarrollo como

> "una recopilación de técnicas y procedimientos estructurados en fases para la producción de productos de software de manera eficaz y englobando todo el ciclo del mismo. Indica que debe hacer, como, cuando y quien debe hacerlo. Determina las etapas y controles a aplicar. " (p. 76)

Así, los métodos deben ajustarse al ciclo de vida del proceso de desarrollo de software, apoyando las distintas actividades, incluyendo la gestión de calidad.

La mayoría de los métodos de desarrollo de software tienen un único fin que es producir software de alta calidad con el menor uso de recursos y tiempo, sin embargo muchos no tienen un método especifico que detalle los pasos para asegurar la calidad del producto.

El siguiente trabajo hace referencia al método GRAY WATCH diseñado por Montilva et al (2000), que describe el ciclo de vida para el desarrollo de aplicaciones Web. El método GRAY WATCH en su más reciente versión Montilva et al (2008) se encuentra enmarcado en el desarrollo de software empresarial bajo el paradigma de reutilización de componentes, con la finalidad de garantizar que la aplicación se entregue a tiempo y dentro del presupuesto acordado con el cliente.

GRAY WATCH define tres grupos de procesos: técnicos, de gestión y de soporte. Los procesos técnicos se relacionan con las actividades de análisis, diseño, implementación y pruebas de las aplicaciones. Los procesos de gestión se encargan de gerenciar el desarrollo de cada aplicación como un proyecto de ingeniería; involucran, por lo tanto, actividades de planificación, organización, administración, dirección y control del proyecto.

Por su parte, los procesos de soporte complementan los procesos técnicos y gerenciales con actividades, tales como: el aseguramiento de la calidad, la gestión de la configuración y la gestión de riesgos del proyecto. Montilva et al (2008).

En este orden de ideas, al examinar los procesos técnicos del método GRAY WATCH, se pudo conocer que la gestión de la calidad que usa este método está orientada mas a la calidad del proceso que a la calidad del producto, y se lleva a cabo a través de actividades de Aseguramiento de Calidad, Verificación y Validación del software mediante el seguimiento de un plan que guía el proceso y permite tomar acciones correctivas.

Es por ello, que con respecto a la calidad del producto se observa un vacío, puesto que el método GRAY WATCH no tiene un proceso que considere los aspectos de evaluación y control de calidad en etapas tempranas de desarrollo,

particularmente, en las primeras fases de desarrollo del software como es el Diseño arquitectónico, que se encarga de producir la estructura de la aplicación representada a través de una arquitectura de software con sus componentes, conectores y restricciones arquitectónicas (Montilva et al, 2008), este proceso no detalla las actividades para obtener requisitos de calidad que permitan evaluar la arquitectura.

Un requisito de calidad, según Losavio et al (2009) son propiedades que caracterizan una solución arquitectónica y esto coincide con la perspectiva de Canelón et al (2009) que considera que los requisitos no funcionales expresados en términos de requisitos de calidad son fundamentales para el diseño arquitectural, por esta razón, en el diseño de la arquitectura se hace necesario un proceso que describa los pasos para la especificación de requisitos de calidad, diseño de modelos de calidad y arquitecturas basadas en atributos de calidad.

De lo anterior se desprende la necesidad de esta investigación que tuvo como finalidad diseñar la extensión del método GRAY WATCH en los procesos técnicos de Análisis y Diseño, de la Aplicación, incorporando actividades que generan productos de calidad, aplicando el estándar de calidad ISO/IEC 25010.

La calidad del producto que se incorpora al método GRAY WATCH en los procesos de Análisis, se centró en la especificación de los requisitos, para ello, se identificaron los requisitos funcionales y no funcionales y las propiedades de calidad asociadas y se construyó el modelo de calidad según el estándar de calidad del producto ISO/IEC 25010. De igual manera, la calidad del producto que se aplica en los Procesos de Diseño se enfocó en la identificación de estilos arquitecturales y en la selección de una arquitectura que satisfaga los atributos de calidad. Finalmente los productos generados en el Procesos de Análisis y Diseño se acoplan a lo largo de la cadena de procesos WATCH.

Para ilustrar la propuesta de solución, se presenta a continuación la figura 1.1:

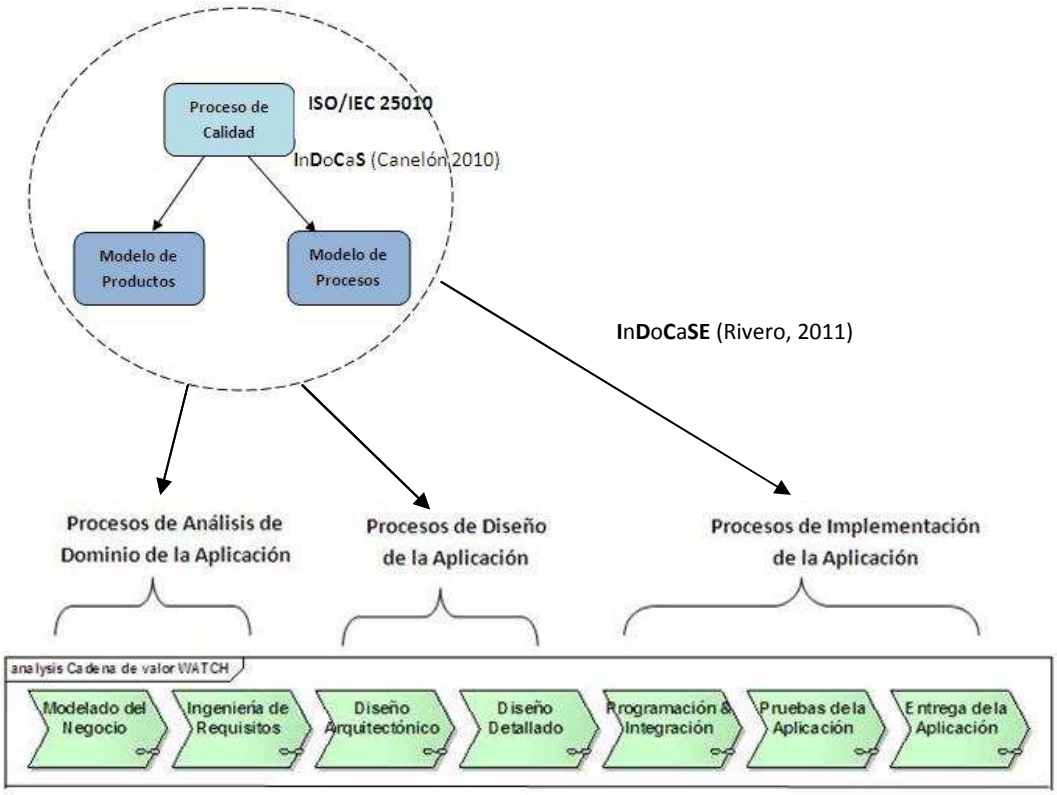


Figura 1.1: Propuesta de Extensión de los procesos Técnicos de GRAY WATCH
Fuente: el autor

Luego de la ilustración anterior cabe destacar que para la extensión del método se tomaron los principios metodológicos de la Ingeniería de Métodos definido por Odell (1996), se creó el Modelo de Productos: para describir los productos que se desarrollaron y el Modelo de Procesos: que explica los pasos a seguir para obtener unos requisitos de calidad y un diseño arquitectónico de calidad. Además se utilizó el proceso para la Ingeniería de Dominio basado en Calidad de Software denominado **InDoCaS** (Canelón, 2010) como metodología para la definición de actividades y productos en los procesos de Análisis y Diseño basados en calidad de software, y el proceso **InDoCaSE** (Rivero, 2011) para acoplar las actividades y productos en el proceso de Implementación.

Perspectivas de la Investigación

En atención a lo expuesto, el autor del presente proyecto consideró pertinente profundizar acerca del modelo de productos más adecuado que describa los artefactos generados en el proceso de calidad basado en la norma ISO/IEC 25010. Adicionalmente se hizo necesario conocer acerca de los pasos a seguir para obtener los requisitos de calidad, modelos de calidad y el diseño arquitectónico. Los resultados de estas acciones permitieron extender el método GRAY WATCH incorporando actividades que generaron productos de calidad aplicando el estándar ISO/IEC 25010, dichos productos se adaptaron al resto de los procesos de la Cadena del método GRAY WATCH. Finalmente se validó la extensión del método GRAY WATCH, para ello se aplicó en un caso de estudio, específicamente el Programa de Investigación de la UPEL-IPB.

Dentro de este orden de ideas, a continuación se presenta el objeto de estudio.

Objetivo General

Diseñar la extensión del método GRAY WATCH basado en el estándar de calidad ISO/IEC 25010.

Objetivos Específicos

1. Definir las actividades y los productos que permitirán expandir los procesos de Análisis y Diseño del método GRAY WATCH aplicando el estándar de calidad ISO/IEC 25010.
2. Incorporar las actividades y los productos en los procesos de Análisis y Diseño del método GRAY WATCH aplicando el estándar de Calidad ISO/IEC 25010.

3. Acoplar el proceso de implementación del método GRAY WATCH con los artefactos generados en los procesos extendidos de análisis y diseño.
4. Aplicar el método GRAY WATCH extendido al caso de estudio: Programa de Investigación UPEL-IPB.

Justificación e Importancia

El método GRAY WATCH está fundamentado en las mejores prácticas de la Ingeniería de Software y la Gestión de Proyectos, comprende todo el ciclo de vida del software; desde el modelado del dominio, especificación de los requisitos, hasta la puesta en operación de la aplicación.

La presente investigación se justificó en la necesidad de mejorar el método GRAY WATCH en lo que respecta a calidad del producto, utilizando el estándar de calidad publicado recientemente, el estándar ISO/IEC 25000 SQUARE, específicamente la división del modelo de calidad ISO/IEC 25010. Con la extensión de este método en sus etapas tempranas de desarrollo se pretende optimizar la gestión de la calidad del producto, asegurando que la aplicación desarrollada cumpla con los estándares de calidad y con las especificaciones solicitadas por el cliente.

Cabe agregar, que esta investigación centró el interés en los aspectos de diseño y evaluación de calidad en los procesos de Análisis y Diseño de la Aplicación, lo que permitirá garantizar un diseño arquitectónico que satisfaga los requisitos de calidad asociados al estándar ISO/IEC 25010, garantizando así la selección de la arquitectura más adecuada y con atributos de calidad.

Esta investigación ofrecerá aportes a la industria desarrolladora de software empresarial, específicamente a los equipos de desarrollo, ya que orientará acerca de que deben hacer y cómo deben desarrollar una aplicación con calidad de software basado en el nuevo estándar ISO/IEC 25010.

Alcance y Limitaciones

El alcance de este estudio se enmarcó en la extensión de los procesos de Análisis y Diseño del método GRAY WATCH, mediante la aplicación del estándar de calidad ISO/IEC 25010, esto se realizó bajo los principios del enfoque de la Ingeniería de Métodos que utiliza el método GRAY WATCH para la creación de un modelo de productos y un modelo de proceso que permitirá integrar actividades y productos basados en dicho estándar. En este propósito, se acoplo los productos de calidad generados al proceso de Implementación del método GRAY WATCH.

Esta investigación se limita a la creación y/o adaptación de actividades que especifican requisitos funcionales y no funcionales, identifican propiedades de calidad, construyen modelos de calidad basados en el estándar ISO/IEC 25010 y permiten evaluar y seleccionar arquitecturas que satisfagan los requisitos de calidad del sistema. Estas actividades se integran al método de desarrollo GRAY WATCH en sus primeras etapas, dentro de sus procesos de desarrollo, específicamente en la Ingeniería de Requisitos y en el Diseño Arquitectónico, para ello se tomó como base el proceso **I**n**D**o**C**a**S** propuesto por Canelón (2010) que tiene como objetivo obtener una arquitectura base para las familias aplicando las actividades correspondientes a las disciplinas Análisis y Diseño del Dominio, para la disciplina de Implementación se utilizará el proceso **I**n**D**o**C**a**SE** propuesto por Rivero (2011).

Cabe destacar que el presente trabajo se delimita solo a la Ingeniería de Aplicación que es la base del método GRAY WATCH, la construcción de componentes basados en la reutilización de activos de software.

Finalmente, para trabajos futuros queda el diseño de métricas que permitan evaluar la calidad de uso del producto software, es decir que permiten evaluar la calidad de la puesta en marcha de la aplicación.

CAPITULO II

MARCO TEÓRICO

Antecedentes

A continuación se hace una exploración de las referencias impresas y electrónicas, con el propósito de enunciar la concepción teórica que sustenta la investigación, apoyada principalmente, en trabajos anteriores que se relacionan con la presente investigación. Para la comprensión de cada antecedente y de su relación con este trabajo se presenta un cuadro de análisis que muestra el enfoque de la investigación. Para ello se utiliza el cuadro referencial propuesto por Rolland et al. (1998) que establece cuatro vistas de un escenario o situación, cada una de ellas permite capturar un aspecto relevante del escenario. En el caso de este trabajo se acordaron las siguientes vistas, preguntas y criterios:

Tabla 2.1 Cuadro de referencia para los antecedentes

Vista	Pregunta	Criterio
Que?	Cual fue el motivo de la investigación?	Objeto de Estudio
Como?	Que procedimiento se llevo a cabo? Que modelos de calidad se uso?	Metodología
	Que herramientas se usaron?	Tecnología
Cuando?	A quienes está dirigida la investigación	Caso de Estudio
Para que?	Cuáles fueron las conclusiones del estudio?	Resultados
	En que sustenta la presente investigación?	Aporte

Fuente: El autor

Estas preguntas permiten abordar cada trabajo de una forma organizada, clara y precisa, determinando cual es la intención de la investigación (objeto de estudio), cuales son los procedimientos que se llevaron a cabo, que metodología (métodos de ingeniería de software, criterios de calidad usados) y herramientas (tecnologías) usadas, cuáles fueron los clientes (caso de estudio donde se aplico), finalmente los resultados obtenidos y el fundamento teórico y metodológico que ofrece a esta investigación.

Las investigaciones relacionadas con el tema tratado en el presente estudio, se presentan en un orden cronológico comenzando con la más antigua.

Se inicia con el trabajo de Palomo et al. (2007), quienes en su artículo "Sistema de Gestión de la Investigación en la Universidad de Talca de Chile" (SGI), evidenciaron los resultados y avances de la implantación del sistema. La Universidad de Talca creó una aplicación con la finalidad de satisfacer requisitos tanto de nivel operativo, táctico como estratégico de las autoridades institucionales, sus investigadores y de los diferentes actores interesados en el trabajo investigativo y sus resultados, en particular el empresarial y gubernativo.

En cuanto a la metodología de desarrollo del software se aplicó el proceso unificado RUP (por sus siglas en ingles Rational Unified Process) (Kruchten y Kroll, 2004). En lo que se refiere a la ingeniería de requisitos, se empleó el Sistema de Administración de Requerimientos (SAR) (Palomo et al., 2007) herramienta de software utilizada por la por la Dirección de Informática, Telecomunicaciones y Medios (DITyM), que permite a desarrolladores y usuarios coordinarse a través de Intranet de la Universidad para satisfacer las condiciones de término del usuario para cada requerimiento.

El sistema fue desarrollado por etapas, una primera versión fue a través de una Intranet, seguida por la versión Web, luego esta versión Web se integro al Sistema Nacional de Investigación en Ciencia, Tecnología e Innovación (SICTI) mediante

Web Services, hasta la última versión sobre una plataforma basada en flujo de trabajo (Workflow).

En cuanto a los resultados, se destaco que la implementación del software en la Universidad de Talca ha contribuido a incrementar significativamente el número de investigadores activos, así como la cantidad de proyectos de investigación ejecutados, el número de publicaciones y la adjudicación de fondos externos e internos. Los autores concluyeron que la versión bajo plataforma tecnológica de Workflow, actualmente vigente, ha logrado mejorar sustancialmente la eficiencia del quehacer administrativo interno asociado a los proyectos de investigación

Esta investigación no sostiene el presente estudio puesto que aborda el desarrollo del software con la metodología RUP y no utiliza modelos de calidad para los requisitos del software. Sin embargo, en relación al caso de estudio seleccionado provee un gran soporte para esta investigación ya que el dominio del negocio es el mismo y proporciona información sobre los procesos de negocios y requisitos del sistema. En la siguiente tabla se puede apreciar el resumen del trabajo.

Tabla 2.2 Resumen de la investigación de Palomo et al. (2007)

Pregunta	**Variable**	**Dimensión**	Palomo y otros (2007)
Que?	Cual fue el motivo de la investigación?	Objeto de Estudio	Implementación de un Sistema de Gestión bajo la plataforma basada en flujo de trabajo (workflow)
Como?	Que procedimiento se llevo a cabo? Que criterios de calidad uso?	Metodología	RUP No utilizo criterios de calidad
	Que herramientas se usaron?	Tecnología	Arquitectura Web, Integración de Aplicaciones, Web services, Worflow
Cuando?	A quienes está dirigida la	Caso de Estudio	Programa de Investigación de la Universidad de Talca de Chile

	investigación		
Para que?	Cuáles fueron las conclusiones del estudio?	Resultados	Aumento del número de investigadores activos, proyectos de investigación, publicaciones y financiamiento a proyectos.
	En que sustenta la presente investigación?	Aporte	Ofrece un soporte en el conocimiento del dominio del caso de estudio.

Fuente: El autor

Bajo otras perspectivas se tiene a Canelón et al. (2009) quienes realizaron una investigación cuyo objeto de estudio fue definir un modelo de calidad para el dominio de las aplicaciones móviles sensibles al contexto. Por consiguiente, se definieron requisitos funcionales y no funcionales que permitieron asegurar la calidad del servicio a los usuarios de este tipo de aplicaciones, pues bien, tienen características propias que afectan el proceso de desarrollo de software y deben ejecutarse en diferentes plataformas de computación ajustándose a diversos dispositivos de acceso y adaptándose a diferentes contextos de uso, esto quiere decir que, es determinante la gestión de calidad de estos servicios.

Para la clasificación de los requisitos utilizaron la taxonomía del Modelo RECLAMO que permitió identificar los requisitos funcionales y no funcionales, luego se integró el estándar ISO 9126-1, para identificar las características y subcaracterísticas de calidad asociada a cada requisito.

Por otra parte, seleccionaron como caso de estudio una Aplicación M-LEARNING, un aprendizaje electrónico independiente del lugar, tiempo y espacio, el aprendizaje móvil permite que los aprendices simulen sus experiencias de aprendizaje en ambientes de aulas de clase, proporcionando la continuidad y adaptabilidad entre contextos de aprendizajes, con el soporte de dispositivos móviles y herramientas de visualización.

En cuanto a los resultados, los autores plantearon que el modelo de calidad logrado es un estándar para el dominio de aplicaciones móviles al contexto y además será usado para el diseño arquitectónico, dentro del proceso de desarrollo de aplicaciones móviles sensibles al contexto y desde una perspectiva de líneas de productos de software. A continuación se refleja el modelo de calidad generado:

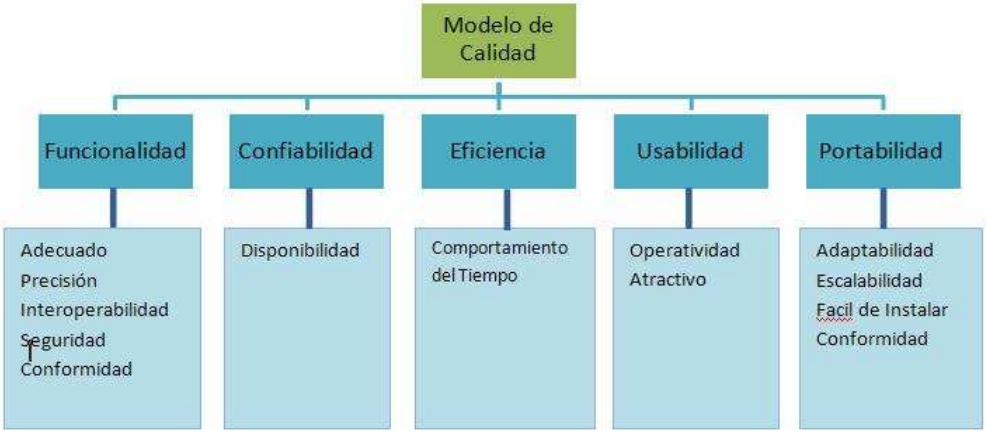


Figura 2.1 Modelo de Calidad estándar para Aplicaciones móviles sensibles al contexto

Fuente: Canelón y otros (2009)

Dicha investigación fundamenta el presente estudio, ya que utiliza la norma de calidad del producto ISO 9126-1 y el modelo RECLAMO para la clasificación e identificación de los requisitos de calidad. Aunque en esta investigación se utilizó la norma ISO/IEC 25010 que vino a reemplazar a la ISO/IEC 9126-1. La presente tabla muestra un resumen del trabajo realizado.

Tabla 2.3 Resumen del trabajo de Canelón y otros (2009)

Pregunta	Variable	Dimensión	Canelón y otros (2009)
Que?	Cual fue el motivo de la investigación?	Objeto de Estudio	Crear un modelo conceptual para aplicaciones móviles sensibles al contexto con criterios de calidad
Como?	Que procedimiento se llevo a cabo? Que criterios de calidad se uso?	Metodología	Modelo de clasificación de requisitos RECLAMO, el modelo de calidad del producto software ISO/IEC 9126-1
	Que herramientas se usaron?	Tecnología	UML modelo conceptual
Cuando?	A quienes está dirigida la investigación	Caso de Estudio	Aplicaciones M-Learning
Para que?	Cuáles fueron las conclusiones del estudio?	Resultados	Se definió un modelo de calidad estándar para aplicaciones móviles con especificación de requisitos de calidad, este modelo será utilizado para el diseño arquitectónico.
	En que sustenta la presente investigación?	Aporte	Utiliza criterios de calidad, se aplico el estándar 9126-1 para definir el modelo de calidad.

Fuente: El autor

En este mismo orden y dirección se puede citar a Castillo et al. (2010) quienes diseñaron un modelo Conceptual denominado Requirements, Aspects Software Quality (REASQ). Para lograr este propósito, consideraron la metodología de desarrollo de software orientada a aspectos: Aspects Oriented Software Development (AOSD) (Jacobson, 2005) que propone la especificación de requisitos no funcionales relacionados al comportamiento de la funcionalidad del sistema o contexto en etapas más tempranas, el Modelo RECLAMO y el estándar de calidad ISO/IEC 25000.

La investigación justifica el proceso de desarrollo de software orientado a aspectos en contra de la orientada a objeto puesto que permiten definir como mayor claridad aspectos del sistema tales como: interoperabilidad, adaptabilidad, disponibilidad y seguridad, además de ayudar a una mejor comprensión y claridad del código del sistema ya que es organizado por aspectos en lugar de objetos.

El propósito principal de este trabajo fue establecer un único modelo conceptual que permita clarificar la terminología emergente AOSD e integrarla con el estándar ISO/IEC 25030 y el modelo RECLAMO, los conceptos generados en este modelo serán formalizados en tres ontologías relacionadas, escritas en Protege con el fin de facilitar la reutilización, representar los ámbitos de la orientación a aspectos, calidad del software y la ingeniería de requisitos.

Se destaca que en la ingeniería de requisitos tomaron el modelo RECLAMO que permite identificar requisitos de calidad para productos de software, este modelo se basa en la vistas de calidad de la ISO/IEC 9126-1, sin embargo en su lugar utilizaron el nuevo estándar ISO/IEC 25030 para requisitos (Canelón, 2007).

A continuación se presenta una figura con el modelo REAQS.

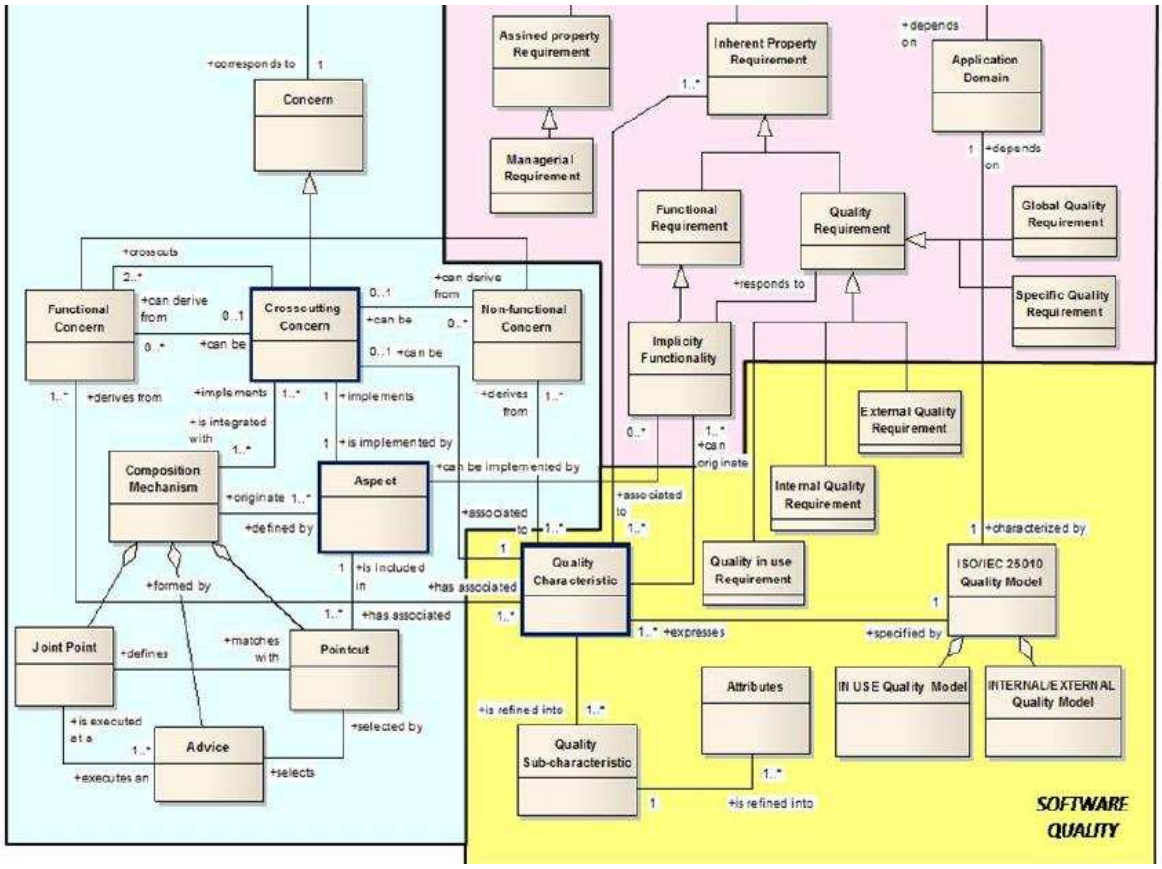


Figura 2.2 Modelo de Requisitos, Aspectos y Calidad de Software
Fuente: Castillo y otros (2010)

En conclusión, los autores plantean que el modelo REASQ obtenido guiará la integración de esas ontologías: Aspects Oriented Software Development (AOSD), el Modelo RECLAMO y el estándar de calidad ISO/IEC 25030, las cuales serán usadas en trabajos futuros para definir procesos de ingeniería de requisitos de calidad orientados a aspectos. Finalmente usaron como caso de estudio para la aplicación del modelo una aplicación Web basada en comercio electrónico, una tienda de juguetes.

Este trabajo establece bases teóricas y metodológicas para la investigación realizada pues integra el modelo de clasificación de requisitos RECLAMO con el estándar ISO/IEC 25030 que se refiere a la división de requisitos de calidad, y se

utiliza en la especificación de requisitos de un producto software a ser desarrollado o como entrada en un proceso de evaluación (ISO/IEC 25000, 2005). La siguiente tabla resume el trabajo realizado por los autores.

Tabla 2.4 Resumen del Trabajo de Castillo y otros (2010)

Vista	Pregunta	Criterio	Castillo y otros (2010)
Que?	Cual fue el motivo de la investigación?	Objeto de Estudio	Integrar los conceptos de AOSD, ingeniería de clasificación de requisitos y modelos de calidad de software.
Como?	Que procedimiento se llevo a cabo?	Metodología	Desarrollo de Software Orientado a Aspectos (AOSD), Modelo RECLAMO, ISO/IEC 9126-1, ISO/IEC 25030.
	Que herramientas se usaron?	Tecnología	UML, protege tool
Cuando?	A quienes está dirigida la investigación	Caso de Estudio	Un e-commerce: tienda de juguete online
Para que?	Cuáles fueron las conclusiones del estudio?	Resultados	Modelo Conceptual REASQ (Requirements Aspects Software Quality). Una ontología para especificar requisitos de calidad en un proceso de desarrollo orientado a aspectos.
	En que sustenta la presente investigación?	Aporte	Especifica los requisitos con RECLAMO y se integro con el nuevo estándar 25030 para definir el modelo de calidad.

De los anteriores planteamientos se puede observar la importancia de aplicar estándares de calidad en las etapas tempranas del proceso de desarrollo de software, ya que garantizan la especificación de los requisitos funcionales y no funcionales con

atributos de calidad, que indicaran los componentes a usar en el diseño arquitectónico. A continuación se presenta una tabla con los aportes y las limitaciones de los trabajos anteriormente presentados para una mejor comprensión.

Tabla 2.5 Aportes y Limitaciones de los Trabajos consultados.

Antecedente	Objetivos	Aportes	Limitaciones
Palomo, I. Veloso, C. y Schmal, R. (2007). *Sistema de Gestión de Investigación de la Universidad de Talca*	Desarrollar un sistema de gestión para la investigación en la Universidad de Talca	Para el caso de estudio seleccionado provee un gran soporte ya que el dominio del negocio es el mismo.	No sirve de base apoyo al estudio puesto que no utiliza criterios de calidad. Y el método de desarrollo usado es RUP.
Canelón R., Losavio F. y Matteo A. (2009) c. *Modelo conceptual para modelación de aplicaciones móviles sensibles al contexto*	Crear un modelo conceptual para el dominio de aplicaciones móviles sensibles al contexto con criterios de calidad	Se definió un modelo de calidad estándar con especificación de requisitos de calidad, bajo la norma ISO 9126-1	Se limita a un dominio específico aplicaciones móviles.
Castillo, I., Losavio, F., Matteo, A. y Boegh, J. (2010). *Requirements, Aspects and Software Quality: The REASQ Model*	Integrar los conceptos de AOSD, modelo de clasificación de requisitos y modelos de calidad de software.	Presenta un Modelo Conceptual para especificar requisitos de calidad en un proceso de desarrollo orientado a aspectos. Usa ISO/25030	Se limita a presentar un modelo con atributos de calidad para procesos de desarrollo de software orientado a aspectos y con la norma ISO 25030.

Fuente: el autor

Bases Teóricas

Para establecer las bases teóricas de esta investigación se detallan algunos conceptos y definiciones que apoyan y ayudan a comprender el fundamento de la misma. La figura 2.3 ilustra el modelo conceptual de las bases que fundamentan la teoría de la investigación:

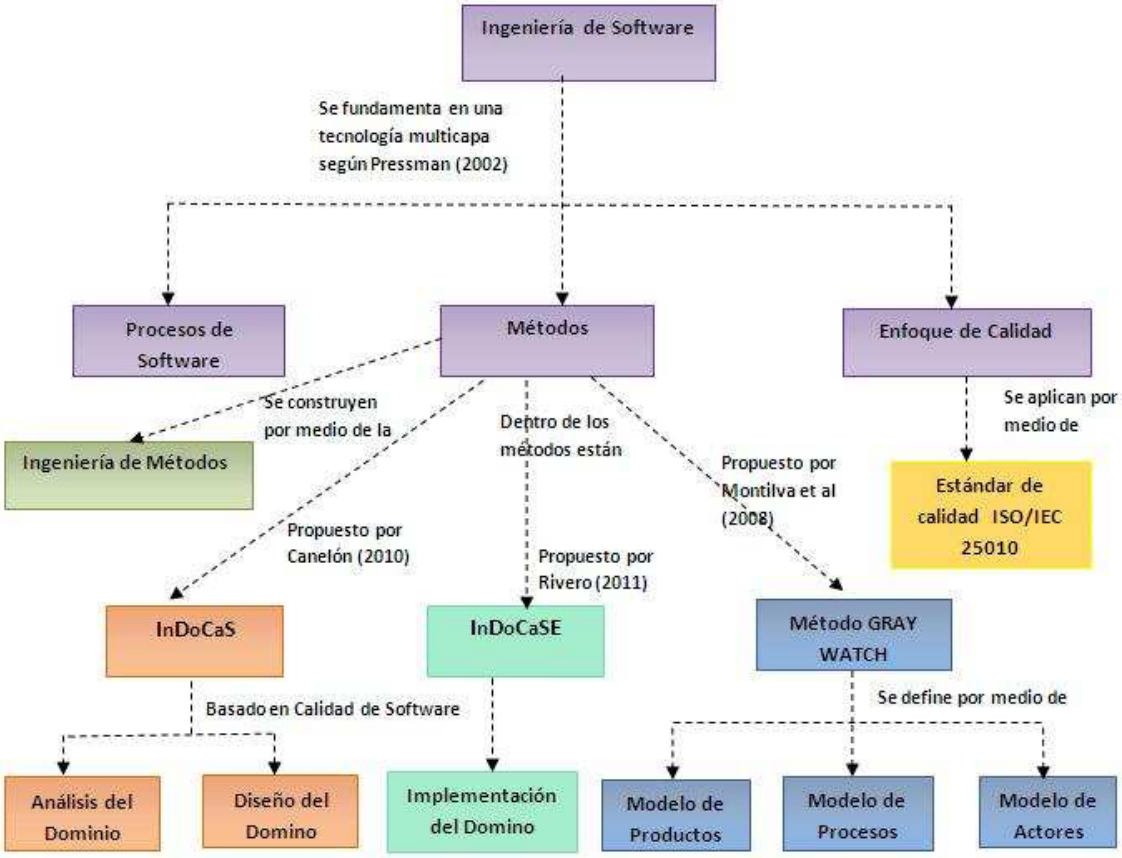


Figura 2.3 Modelo Conceptual a desarrollar en las bases teóricas

Fuente: El autor

Software

La mayoría de personas asocian el término software con programas de computadoras, sin embargo, la definición es mucho más amplia, el software no son solo programas, sino todos los documentos asociados y la configuración de datos que se necesitan para que estos programas operen de manera correcta. Según Campderrich (2003) un software denominado también aplicación es:

> un conjunto de programas que en su forma definitiva se pueden ejecutar, pero comprende también las definiciones de estructura de datos (por ejemplo, definiciones de bases de datos) que utilizan estos programas y también la documentación referente a todo ello (tanto la documentación de ayuda en el uso del software para sus usuarios como la documentación generada durante su construcción, parte de la cual servirá para su mantenimiento.(p 15)

Por otra parte, Pressman (2002) define el software de computadora como "el producto que diseñan y construyen los ingenieros de software" (p. 3). Para la construcción del producto se requiere de un enfoque de ingeniería de software, por esta razón esta disciplina ha creado métodos, técnicas y herramientas que indican cómo desarrollar un producto software. El autor ha caracterizado la ingeniería de software como una tecnología multicapas, representada en la figura 2.5:

Figura 2.4 Tecnología Multicapas de la Ingeniería de Software
Fuente: Pressman (2002)

Dichas capas se describen a continuación:

- Cualquier disciplina de ingeniería (incluida la ingeniería del software) debe descansar sobre un esfuerzo de organización de **calidad**. La gestión total de la calidad y las filosofías similares fomentan una cultura continua de mejoras de procesos que conduce al desarrollo de enfoques cada vez más robustos para la ingeniería del software.
- El fundamento de la ingeniería de software es la **capa proceso**. El proceso define un marco de trabajo para un conjunto de áreas clave, las cuales forman la base del control de gestión de proyectos de software y establecen el contexto en el cual: se aplican los métodos técnicos, se producen resultados de trabajo, se establecen hitos, se asegura la calidad y el cambio se gestiona adecuadamente.
- Los **métodos** de la ingeniería de software indican cómo construir técnicamente el software. Los métodos abarcan una gran gama de tareas que incluyen análisis de requisitos, diseño, construcción de programas, pruebas y mantenimiento. Estos métodos dependen de un conjunto de principios básicos que gobiernan cada área de la tecnología e incluyen actividades de modelado y otras técnicas descriptivas.
- Las **herramientas** de la ingeniería del software proporcionan un soporte automático o semi-automático para el proceso y los métodos, a estas herramientas se les llama herramientas CASE (C*omputer-Aided Software Engineering*).

Hechas las consideraciones anteriores se puede decir que, la ingeniería de software comprende todos los aspectos de la producción del software, a su vez el proceso de software incluye todas las actividades relativas a la gestión y desarrollo del software, y los métodos son guías organizadas para producir software incluyen los modelos. A continuación se define procesos de desarrollo de software.

Proceso de desarrollo de software

Se puede definir Proceso al conjunto ordenado de pasos a seguir para llegar a la solución de un problema u obtención de un producto, en este caso particular, para

lograr la obtención de un producto software que resuelva un problema. Así, Pressman (2002) define un proceso de software "como un marco de trabajo de las tareas que se requieren para construir software de alta calidad" (p. 13)

En este mismo sentido, Sommerville (2005) define un proceso de desarrollo de software como "un conjunto de actividades y resultados asociados que producen un producto de software. Estas actividades son llevadas a cabo por ingenieros de software." (p. 7). El proceso de desarrollo puede involucrar numerosas y variadas tareas, desde lo administrativo, pasando por lo técnico y hasta la gestión y el gerenciamiento.

Por otra parte, están los métodos, que son un procedimiento para obtener algo, los ingenieros de software han creado diferentes métodos y modelos que guían técnicamente la construcción del software, según Sommerville (2005) "un método de ingeniería de software es un enfoque estructurado para el desarrollo de software cuyo propósito es facilitar la producción de software de alta calidad de una forma costeable" (p. 10).

Existen diversos modelos de desarrollo de software, que tienen como propósito la producción eficaz y eficiente de un producto software que reúna los requisitos del cliente, están los enfoques estructurados (modelo en cascada), los tradicionales (desarrollo en espiral, por prototipos), los iterativos e incrementales, es el caso de las metodologías ágiles o livianos (ejemplo XP), pesados y lentos (ejemplo RUP) y variantes intermedias, algunos de esos son Extreme Programming (XP), Rational Unified Process (RUP), Feature Driven Development (FDD), etc. (Sommerville, 2005).

Así surgió la ingeniería de métodos, que según Rolland (1997) "representa el esfuerzo de mejorar la utilidad de los métodos de desarrollo de sistemas, mediante la creación de un marco de adaptación donde los métodos se crean para que coincidan con situaciones concretas de la organización" (p. 1). En base a lo mencionado se puede decir que la disciplina que se utiliza para la creación de métodos es la ingeniería de métodos, que se presenta a continuación.

Ingeniería de Métodos

La ingeniería de métodos (IM) según Brinkkemper (1996) "es la disciplina de la ingeniería para diseñar, construir y adaptar métodos, técnicas y herramientas para el desarrollo de sistemas de información" (p. 276).

La ingeniería de métodos establece una clara separación entre el producto que el método elabora y el proceso que elabora el producto. Según Harmsen (1997) el aspecto conceptual de un método consiste de un modelo de proceso y un conjunto de descripciones de producto. El modelo de procesos describe los pasos y actividades descritas por el método. Las descripciones del producto preescriben los contenidos del producto que será entregado por esos pasos o actividades.

En base a lo anteriormente expuesto, se puede decir que un modelo de producto es una especificación de un producto entregado o requerido dentro del método. Para Brinkkemper (1996) este modelo es la estructura de los productos, que pueden ser entregables, modelos, tablas, diagramas, documentos, etc que se producen en un método de desarrollo y el modelo de procesos es la descripción de los pasos que se realizan en el método, pueden ser determinar los requisitos, analizar los requisitos, diseñar modelo de datos, etc.

Este enfoque es usado por el método GRAY WATCH que metodológicamente se basa en un modelo de procesos, un modelo de productos y un modelo de actores. A continuación se describe este método de desarrollo.

Método GRAY WATCH

El método WATCH es un método de desarrollo de software, particularmente orientado al desarrollo de software empresarial, surgió inicialmente por la necesidad de un modelo de proceso para el desarrollo de proyectos de software pequeños y medianos desarrollados por pequeñas empresas. (Montilva (et al. 2000). Luego fue

extendido por Hamar (2003) a través del método denominado WATCH COMPONENT, para desarrollar el ciclo de vida de un componente reutilizable, desde su especificación hasta su liberación.

Recientemente Montilva et al (2008) presentaron su más reciente versión denominada GRAY WATCH Método de desarrollo de software para aplicaciones empresariales, 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. 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. (Montilva et al. ,2008)

El método GRAY 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 relación al paradigma de desarrollo usado, el Método GRAY WATCH promueve la reutilización de activos de software, lo que reduce costos y aumenta la calidad de los productos de software elaborados usando el método. Entre estos activos están los siguientes: arquitecturas de dominio, patrones de diseño, componentes de software reutilizables y plantillas de documentos (Montilva et al,2008).

Los elementos metodológicos del método GRAY WATCH están basados en los principios y en los conceptos de la Ingenierita de Métodos. A continuación se ilustra el modelo conceptual:

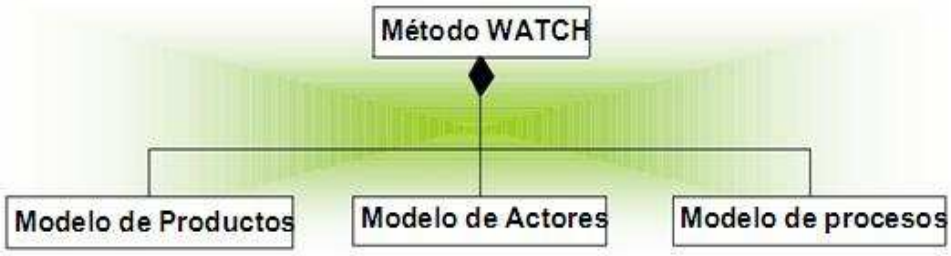


Figura 2.5 Modelo Conceptual del método GRAY WATCH
Fuente : Montilva (2007)

Como puede observarse en la figura anterior el método GRAY WATCH está compuesto por tres modelos fundamentales:

1. Un **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.
2. Un **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
3. Un **modelo de procesos** que describe detalladamente los procesos técnicos, gerenciales y de soporte que los equipos de desarrollo deberán emplear para elaborar las aplicaciones.

Modelo de Productos

El modelo del producto describe las características generales que tienen las aplicaciones empresariales e identifica los productos intermedio y finales que se

deben producir durante el desarrollo de una aplicación. La siguiente figura representa el modelo de productos del GRAY WATCH.

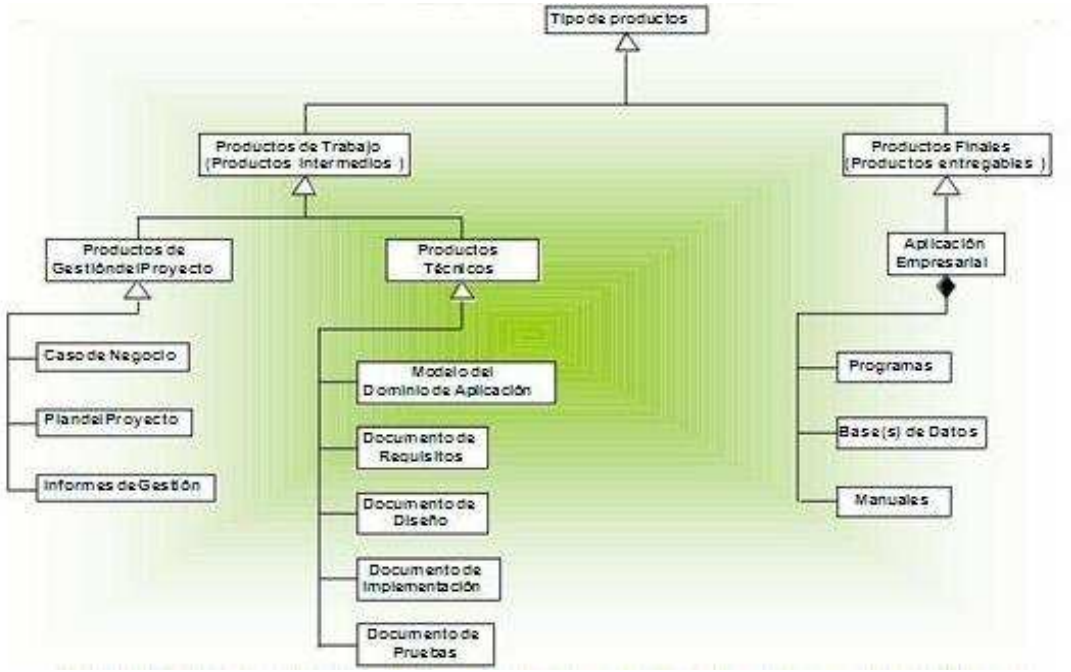


Figura 2.6 Modelo de Productos del método GRAY WATCH
Fuente : Montilva (2007)

Modelo de Procesos

El modelo de procesos del método GRAY WATCH establece los procesos necesarios para gestionar proyectos de desarrollo de aplicaciones empresariales y llevar a cabo las actividades técnicas y de soporte que requieren estos proyectos. Para efectos del presente trabajo se describirán los procesos de desarrollo que son los procesos técnicos que se deben ejecutar para producir el software.

Los procesos técnicos del método se dividen en tres grupos: Procesos de Análisis, Procesos de Diseño y Procesos de Implementación. Los procesos de análisis cubren los procesos de Modelado del Negocio o del dominio de la Aplicación y el de Ingeniería de Requisitos; los procesos de diseño cubren los procesos de Diseño

Arquitectónico y Diseño Detallado; mientras que, los procesos de implementación agrupan los procesos de Construcción & Integración, Pruebas de la Aplicación y Entrega de la Aplicación.

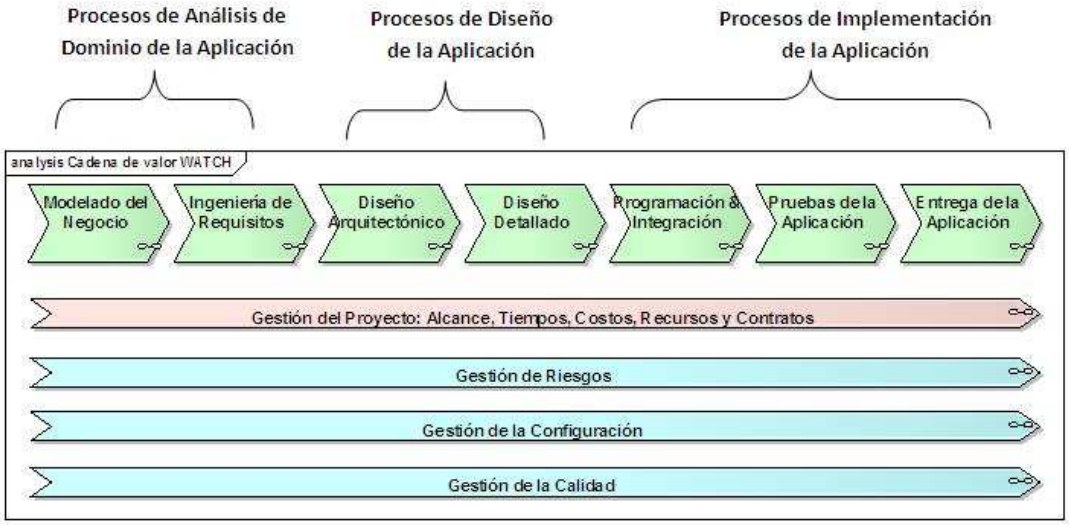


Figura 2.7 Cadena de Valor del Método GRAY WATCH

Fuente : Montilva et al (2008)

Procesos de Análisis

Los procesos de Análisis tienen como objetivos principales los siguientes: (1) entender y modelar el Sistema de Negocios que constituye el dominio de la aplicación empresarial; y (2) definir y especificar el conjunto de requisitos funcionales y no-funcionales que la aplicación empresarial debe satisfacer.

El proceso de Ingeniería de Requisitos requiere de la ejecución de cinco subprocesos complementarios para especificar los requisitos de la aplicación empresarial. Se generan productos intermedios y un documento de definición de requisitos y un documento de especificación de requisitos. A continuación se muestra la representación de los subprocesos.

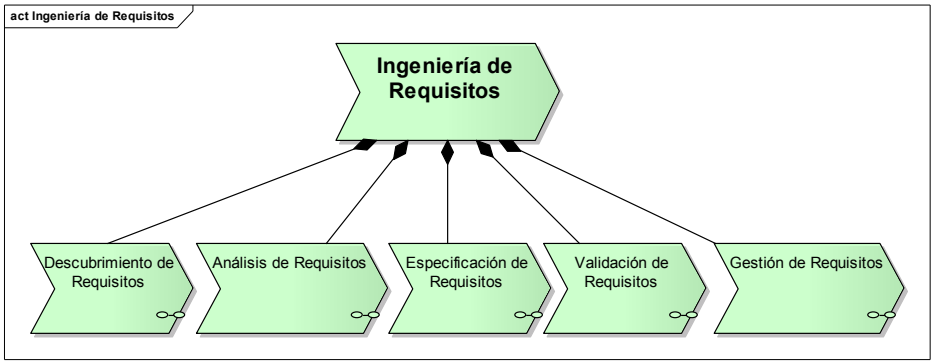


Figura 2.8: Subprocesos de la Ingeniería de Requisitos del Proceso Análisis
Fuente: Montilva et al (2008)

Procesos de Diseño

Describen los procesos técnicos de diseño relacionados con el *cómo* debe ser construida la aplicación empresarial para satisfacer los requisitos previamente recolectados. Este grupo de procesos está compuesto por los procesos de Diseño Arquitectónico y Diseño Detallado de la Aplicación, Montilva et al (2008). El Diseño Arquitectónico produce la estructura de la aplicación representada como una arquitectura de software que muestra los componentes de la aplicación, sus conectores y las restricciones arquitectónicas, además se identifican los subsistemas que tiene la aplicación, las relaciones entre ellos y las características, interacción y distribución de componentes .

El Diseño detallado permite especificar de manera precisa cada uno de los componentes de la arquitectura, incluyendo las interfaces, el modelo de datos y las conexiones previstas en la arquitectura. Estos procesos generan artefactos, a continuación se ilustran los productos del proceso de Diseño arquitectónico.

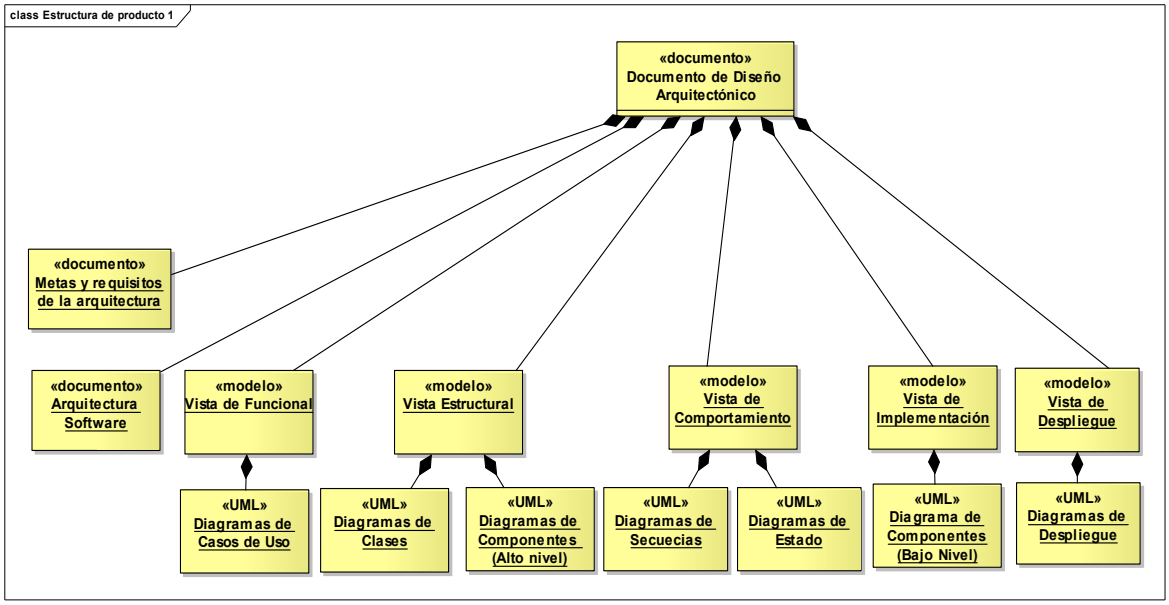


Figura 2.9: Modelo de Productos del Diseño Arquitectónico

Fuente: Montilva et al (2008)

El Diseño Detallado describe cómo se debe implementar cada uno de estos componentes arquitectónicos. Montilva et al (2008).

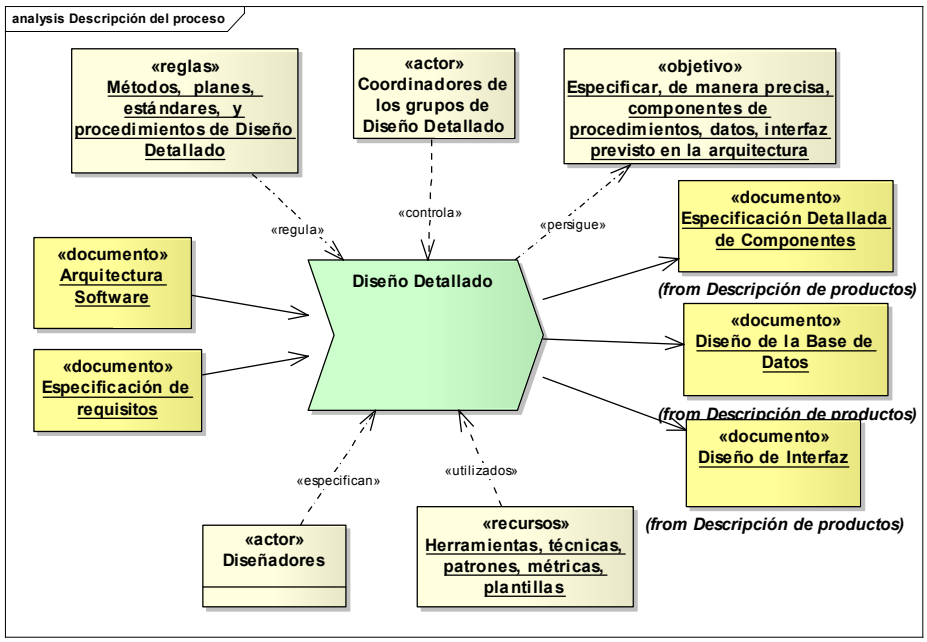


Figura 2.10: Descripción del Proceso de Diseño detallado del método GRAY WATCH

Fuente: Montilva et al (2008)

Procesos de Implementación

Los procesos de implementación relacionados con la programación, pruebas y puesta en operación de la aplicación en sus diferentes versiones, se caracterizan por producir una versión de la aplicación de acuerdo a las especificaciones de diseño arquitectónico y detallado elaboradas en los procesos de diseño; asegurarse de que la versión cumple con todos los requisitos acordados y satisface las necesidades del cliente; y poner en producción la nueva versión en la infraestructura o plataforma de operación instalada para tal efecto. (Montilva et al. 2008).

Este grupo está compuesto por los procesos de Programación & Integración, Pruebas de la Aplicación y Entrega de la Aplicación. El proceso de Programación & Integración consiste en elaborar, codificar o adaptar cada uno de los componentes que integran las diferentes versiones de la aplicación empresarial; probar cada componente como una unidad; integrar estos componentes de acuerdo a la arquitectura diseñada; y probar la integración de estos componentes. A continuación se muestra el aprovisionamiento de componentes.

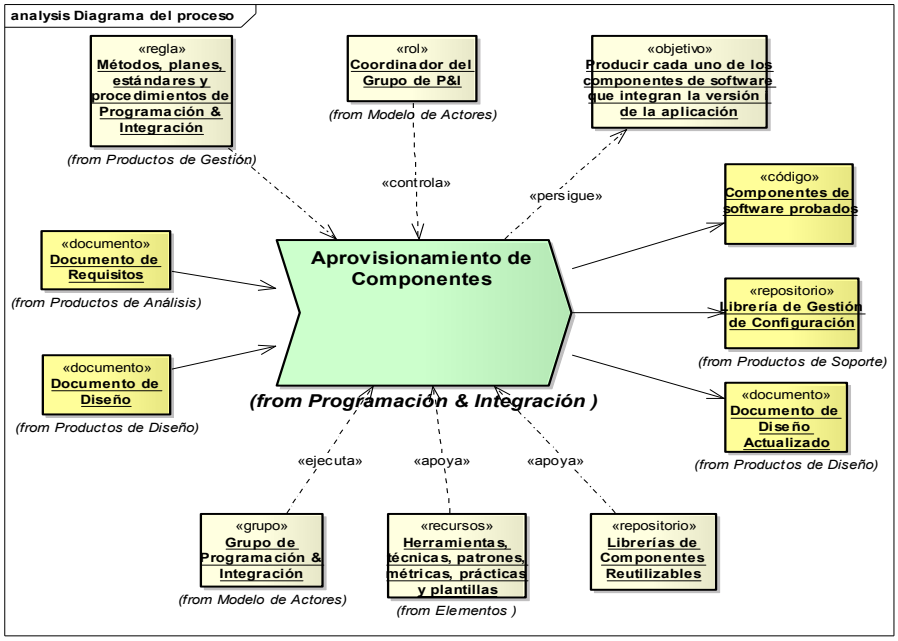


Figura 2.11: Descripción del Proceso Aprovisionamiento de Componentes

Fuente: Montilva et al (2008)

Proceso para la Ingeniería de Dominio basado en Calidad de Software (**I**n**D**o**C**a**S**)

El proceso para la Ingeniería del Dominio basado en Calidad de Software denominado **I**n**D**o**C**a**S** (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. Se considera dominio como una familia de sistemas de software que tienen características comunes.

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 variabilidad. (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 **I**n**D**o**C**a**S** para la evaluación arquitectural.

Análisis del Dominio InDoCaS

En la fase de Análisis del Dominio se especifican todos los aspectos comunes de las familias del dominio y las 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

A continuación se ilustra la figura con el diagrama de actividades del Análisis, 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 indican que técnicas se aplican en cada una de las actividades.

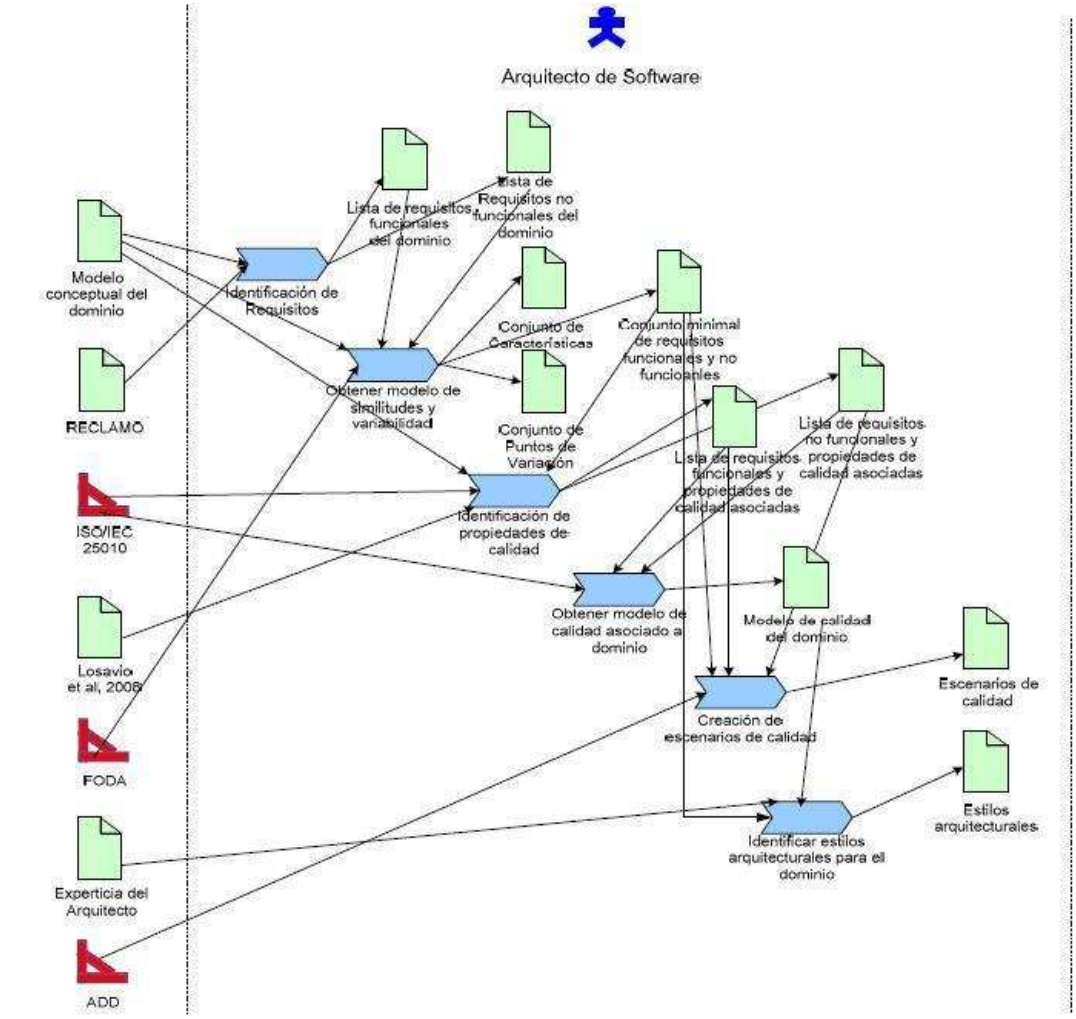


Figura 2.12: Diagrama de Actividades del Análisis del Dominio de **InDoCaS**

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, este subproceso propone soluciones consistentes a un conjunto de requisitos arquitecturales. Como se muestra en la figura, las entradas a esta fase son los artefactos: Conjunto mínimal de requisitos funcionales y no funcionales, Estilos arquitecturales y Catálogos de

patrones arquitecturales obtenidos en la disciplina del Análisis del Dominio.

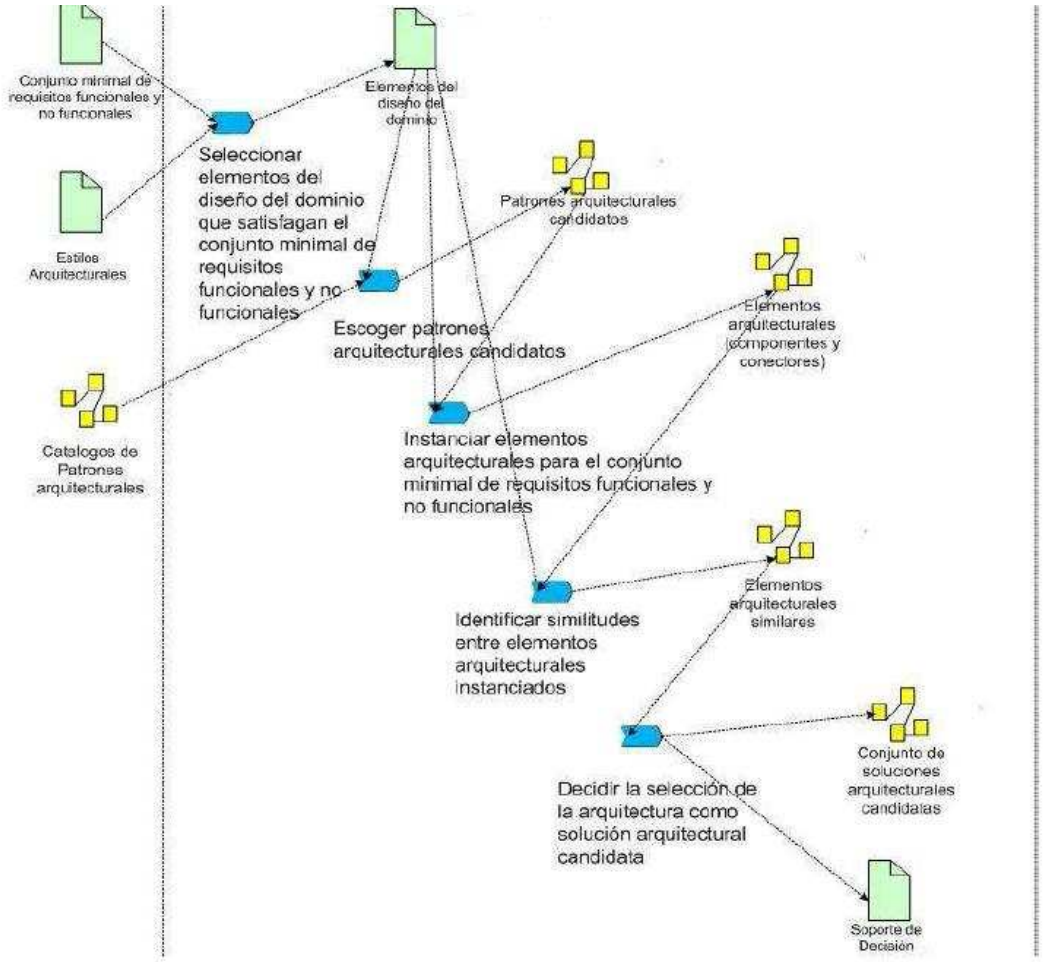


Figura 2.13: Diagrama de Actividades del Diseño del Dominio. Síntesis Arquitectural
Fuente: Canelón (2010)

Como se observa en la figura anterior, 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 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 qué grado se satisfacen los atributos de calidad. A continuación se muestra el diagrama de actividades de este subproceso.

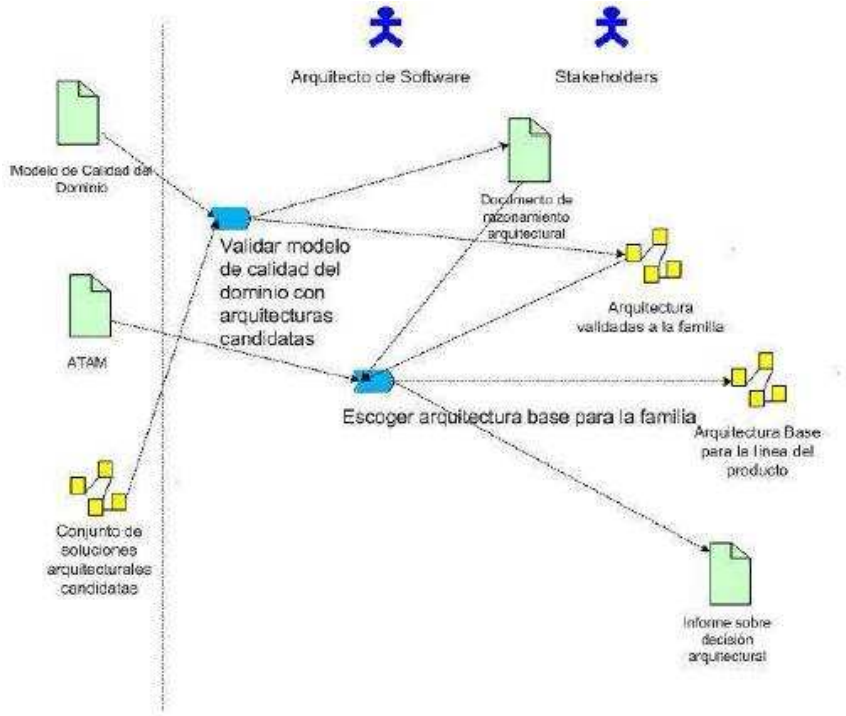


Figura 2.14: Diagrama de Actividades Diseño del Dominio. Evaluación Arquitectural
Fuente: Canelón (2010)

InDoCaSE

El proceso **InDoCaS Expandido** fue propuesto por Rivero (2011), y es una combinación de **InDoCaS** (Canelón, 2010) y el método **WATCH-COMPONENT** (Hamar, 2003). Este proceso para la ingeniería de Dominio se utiliza en el enfoque de desarrollo de líneas de producto de software, y está orientado a la disciplina de

Implementación del dominio. A continuación se muestra una figura que ilustra la implementación del dominio.

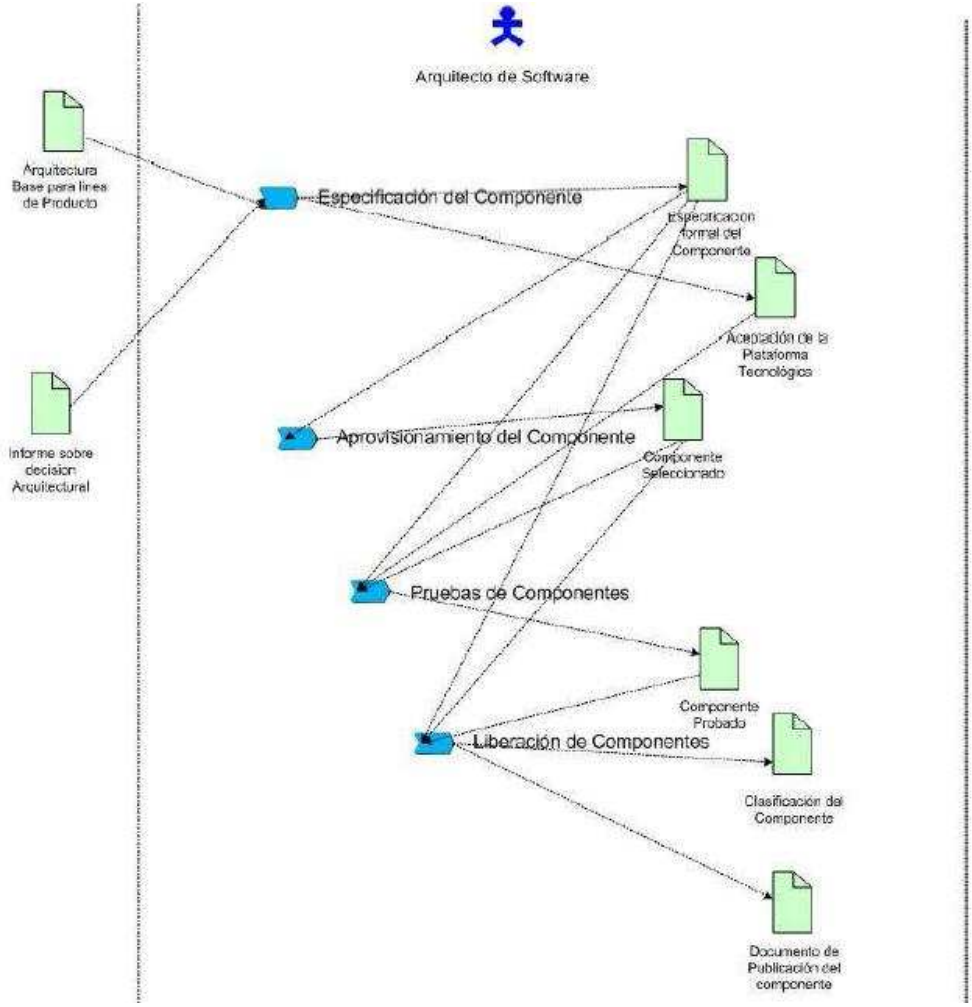


Figura 2.15 Diagrama de Actividades Implementación del Dominio
Fuente: Rivero (2011)

Calidad del Software

En la actualidad el término calidad se utiliza con frecuencia en diversos ámbitos de la sociedad. La Organización Internacional de Estándares (ISO, por sus siglas en inglés) ha publicado varios estándares relacionados con la calidad en general, la norma ISO 8402 (citado por Valdivia,2005) conceptualiza la calidad como "Totalidad de características de un producto que le confieren su aptitud para satisfacer unas necesidades expresadas o implícitas" (p.13), debe entenderse que para esta norma la expresión "necesidades expresadas o implícitas" se refiere a que las necesidades pueden ser definidas por un contrato o son inherentes a la funcionalidad del producto.

Por consiguiente, la calidad es la cualidad de un producto para satisfacer las necesidades del usuario, cabe destacar que no solamente se refiere a productos tangibles sino a programas. Es por ello, que dentro del contexto de la Ingeniería de Software, ha surgido una definición de calidad, según la IEEE "es el grado con el que un sistema, componente o proceso cumple los requisitos especificados y las necesidades o expectativas del cliente o usuario". (Citado por López y Cabrera, 2008, p.327)

Esta definición coincide con la concepción de Pressman (2002), que indica que la calidad del sistema comprende cumplir con los requisitos, especificaciones y el diseño del sistema. Visto de esta forma, el autor plantea la calidad desde dos perspectivas:

> La calidad de *diseño* se refiere a las características que especifican los ingenieros de software para un elemento. El grado de materiales, tolerancias y las especificaciones del rendimiento contribuyen a la calidad del diseño. La *calidad de concordancia* es el grado de cumplimiento de las especificaciones de diseño durante su realización. Una vez más, cuanto mayor sea el grado de cumplimento, más alto será el nivel de calidad de concordancia. (p. 132-133)

La calidad de concordancia es un aspecto centrado principalmente en la implementación. Si la implementación sigue el diseño, y el sistema resultante cumple los objetivos de requisitos y de rendimiento, la calidad de concordancia es alta. Dicho

de otra manera, la calidad del software se debe diferenciar entre la calidad del producto y la calidad del proceso de desarrollo de éste (calidad de diseño y fabricación). No obstante, las metas que se establezcan para la calidad del producto van a determinar los objetivos del proceso de desarrollo. Esta investigación se centrara en la calidad del diseño del producto.

Dentro de este marco de referencia tenemos las normas ISO que definen la calidad de software.

Estándar ISO/IEC 25000

La nueva norma ISO/IEC 25000 proporciona una guía para el uso de las nuevas series de estándares internacionales, llamados Requisitos y Evaluación de Calidad de Productos de Software (SQuaRE: Software Product Quality Requirements and Evaluation). (ISO/IEC 25000,2005). Estas constituyen una serie de normas basadas en la ISO 9126 (calidad del Producto) y en la ISO 14598 (Evaluación del Software), y su objetivo principal es guiar el desarrollo de los productos de software con la especificación y evaluación de requisitos de calidad. Establece criterios para la especificación de requisitos de calidad de productos software, sus métricas y su evaluación.

Según Calero et al. (2010), esta serie de estándares interpreta la calidad de un sistema software como "el grado en que el sistema satisface las necesidades implícitas y explicitas de sus diferentes usuarios (stakeholders)" (p.55). Estas necesidades se representan en SQUARE en diferentes modelos, el modelo de calidad de producto software, el modelo de calidad de datos y el modelo de calidad en uso. SQuaRE está formada por las divisiones siguientes:

- ISO/IEC 25000. División de gestión de calidad. Los estándares que forman esta división definen todos los modelos comunes, términos y referencias a los que se alude en las demás divisiones de SQuaRE.

- ISO/IEC 25010. División del modelo de calidad. El estándar que conforma esta división presenta un modelo de calidad detallado, incluyendo características para la calidad interna, externa y en uso.
- ISO/IEC 25020. División de mediciones de calidad. Los estándares pertenecientes a esta división incluyen un modelo de referencia de calidad del producto software, definiciones matemáticas de las métricas de calidad y una guía práctica para su aplicación. Presenta aplicaciones de métricas para la calidad de software interna, externa y en uso.
- ISO/IEC 25030. División de requisitos de calidad. Los estándares que forman parte de esta división ayudan a especificar los requisitos de calidad. Estos requisitos pueden ser usados en el proceso de especificación de requisitos de calidad para un producto software que va a ser desarrollado ó como entrada para un proceso de evaluación. El proceso de definición de requisitos se guía por el establecido en la norma ISO/IEC 15288 (ISO, 2003).
- ISO/IEC 25040. División de evaluación de la calidad. Estos estándares proporcionan requisitos, recomendaciones y guías para la evaluación de un producto software, tanto si la llevan a cabo evaluadores, como clientes o desarrolladores.
- ISO/IEC 25050–25099. Estándares de extensión SQuaRE. Incluyen requisitos para la calidad de productos de software "Off-The-Self" y para el formato común de la industria (CIF) para informes de usabilidad.

La siguiente figura ilustra la familia de ISO 25000:

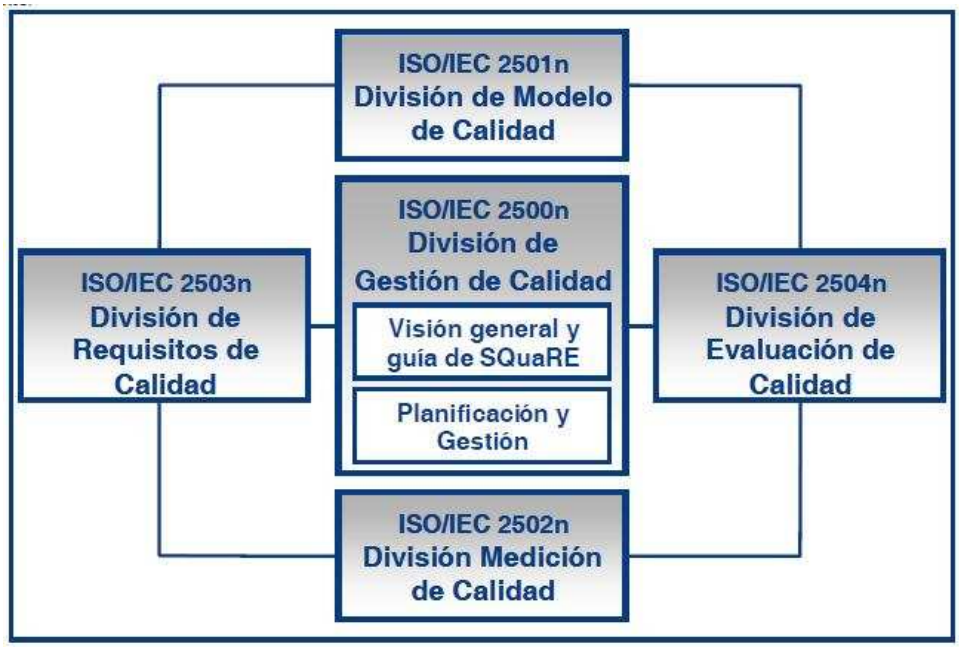


Figura 2.16: Familia de normas ISO 25000.

Fuente: ISO/IEC 25000 (2005)

Cabe destacar, que para lograr el propósito de esta investigación se ha tomado el estándar ISO/IEC 25010 de la División del modelo de calidad.

Estándar ISO/IEC 25010

Este modelo define tres vistas diferenciadas en el estudio de la calidad del producto, las cuales se mencionaran a continuación:

- Vista interna: esta vista se ocupa de las propiedades del software como: el tamaño, la complejidad o la conformidad con las normas de orientación a objetos.
- Vista externa: vista que analiza el comportamiento del software en producción y estudia sus atributos, por ejemplo: el rendimiento de un software en una

máquina determinada, el uso de memoria de un programa o el tiempo de funcionamiento entre fallos.

- Vista en uso: mide la productividad y efectividad del usuario final al utilizar el software.

En referencia a la clasificación anterior se puede decir que, la primera vista puede utilizarse desde las primeras fases del desarrollo, permitiendo detectar deficiencias en el software en edades muy tempranas del ciclo de vida del software. La segunda, sin embargo, necesita que el producto software este completo y se utilizará por tanto en el pase a producción del producto, siendo muy dependiente de la máquina donde se ejecute. Por último la tercera vista que también estudia el producto software finalizado será dependiente del usuario y estará condicionada a los factores personales del mismo.

ISO/IEC 25010 describe un modelo bipartito para la calidad del producto de software:

a) Calidad interna y calidad externa.
b) Calidad en el uso.

La primera parte del modelo especifica ocho características para la calidad interna y externa, que se subdividen más a fondo en subcaracterísticas, que se manifiestan externamente cuando el software se utiliza como parte de un sistema informático, y es resultado de las cualidades internas del software. Este estándar internacional no elabora el modelo para la calidad interna y externa debajo del nivel de subcaracterísticas. La segunda parte del modelo especifica cinco características de la calidad en uso, 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. Las características y subcaracterísticas proveen terminología consistente para la calidad de producto de software. También proveen

una estructura para requisitos de calidad específicos para software, y obtener compensaciones entre las capacidades del producto de software.

El estándar ISO/IEC 25010 define ocho características de calidad: **funcionalidad, seguridad, interoperabilidad, confiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad**. La siguiente figura ilustra el modelo de calidad con las vistas internas y externas.

Figura 2.17 Modelo de Calidad interna y externa ISO/IEC 25010

Fuente: Rodríguez (2010)

Como se puede observar en la figura anterior las modificaciones de este estándar con respecto al ISO/IEC 9126-1 son las siguientes: Seguridad: ha sido añadida como una característica con subcaracterísticas asociadas, en lugar de ser una sucaracterística de Funcionalidad. Compatibilidad: ha sido añadida como una característica con subcaracterísticas propias que no estaban en el anterior modelo Se han añadido algunas sub-características que no estaban en el modelo anterior tales como: completitud funcional, disponibilidad, protección contra errores del usuario, modularidad y reusabilidad. Además los nombres de las características son más específicos para evitar la confusión con otros significados más generales.

Por otra parte, se tiene el modelo de calidad de la vista en uso, los atributos se categorizan en seis, como se muestra en la siguiente figura:

Figura 2.18 Modelo de Calidad en uso ISO/IEC 25010

Fuente: ISO/IEC 25010 (2011)

Por su parte, Canelón (2007), expone que la calidad en uso "es una medida de la calidad del sistema en su ambiente operacional. Es determinado por la naturaleza del software, del hardware, del ambiente de funcionamiento, y de las características de los usuarios, de las tareas y del ambiente social." (p. 15). Esto quiere decir que la calidad en uso es el grado en que un producto utilizado por usuarios específicos, satisface sus necesidades para conseguir los objetivos establecidos la calidad uso para los contextos específicos del uso.

CAPITULO III

MARCO METODOLÓGICO

En este capítulo se describen las técnicas que se utilizaron para llevar a cabo la investigación, apoyándose en las bases teóricas definidas y descritas en el capitulo anterior relacionadas con el objeto de estudio.

Naturaleza de la Investigación

De acuerdo a la naturaleza y características del problema objeto de estudio, esta investigación se enmarcó en la modalidad de Proyectos Especiales, según el Manual para la Elaboración de Trabajos de Grado, de Especialización, Maestría y Tesis Doctorales UPEL (2006), los Proyectos Especiales se definen como:

> Trabajos que lleven a creaciones tangibles, susceptibles de ser utilizadas como soluciones a problemas demostrados, o que respondan a necesidades e intereses de tipo cultural. Se incluyen en esta categoría los trabajos de elaboración de libros de texto y de materiales de apoyo educativo, el desarrollo de software, prototipos y de productos tecnológicos en general (p. 22).

Ciertamente, el presente estudio admitió la creación de un proceso de calidad basado en el estándar ISO/IEC 25010 que se adaptó al método de desarrollo de software GRAY WATCH.

De acuerdo al tipo de investigación, constituyó una investigación documental, ya que depende fundamentalmente de la información que se obtiene o se consulta en documentos. La investigación documental según Bernal (2006) “consiste en un análisis de la información escrita sobre un determinado tema, con el propósito de establecer relaciones, diferencias, etapas, posturas del tema objeto de estudio”. (p.41)

Diseño de la Investigación

El diseño de la investigación es la estrategia que emplea el investigador para responder al problema planteado, aquí se plantea el plan global de investigación, Tamayo (1999) expresa que el diseño "es la estructura a seguir en una investigación, ejerciendo el control de la misma a fin de encontrar resultados confiables y su relación con los interrogantes surgidos de los supuestos e hipótesis-problema" (p. 71).

Procedimiento

Para está investigación, se aplicó el siguiente método de recolección: Revisión Bibliográfica, que permitió conocer conceptos y características de los modelos de calidad del estándar ISO/IEC 25010, se estudiaron los procesos de análisis, diseño e implementación del método GRAY WATCH , por otra parte se revisaron las actividades y artefactos obtenidos en el proceso **I**n**D**o**C**a**S** para el análisis y diseño del dominio con calidad de software, y las actividades y artefactos obtenidos en el proceso **I**n**D**o**C**a**SE** para la implementación del dominio con calidad de software

En cuanto a los aspectos metodológicos, se siguieron los principios propuestos por Montilva et al. (2004) que plantean que un método está estructurado por un modelo de productos, un modelo de procesos y un modelo de grupo (actores). Para el desarrollo del modelo de procesos y el modelo de producto se tomarán como base las actividades y artefactos del proceso **I**n**D**o**C**a**S** propuesto por Canelón (2010) y del proceso **I**n**D**o**C**a**SE** propuesto por Rivero (2011).

En base a lo anterior, la investigación se llevó a cabo mediante la ejecución de las siguientes fases:

Fase I. Definición de las actividades y de los productos que permitirán expandir el método GRAY WATCH aplicando el estándar de Calidad ISO/IEC 25010

En la primera fase se determinaron las actividades en el proceso de Análisis y Diseño del Método GRAY WATCH, aplicando el estándar de calidad ISO/IEC 25010, se consideró como base las actividades del proceso denominado **I**n**D**o**C**a**S** (Canelón, 2010). Además se describió el modelo del Producto, el cual especifica los artefactos generados en estos procesos.

Fase II. Incorporación de las actividades y productos en el Proceso de Análisis y Diseño del Método GRAY WATCH aplicando el estándar de Calidad ISO/IEC 25010.

En esta segunda fase, se agregaron y adaptaron las actividades dentro del Proceso de Análisis y Diseño del método GRAY WATCH y se hizo una representación gráfica de los pasos, actividades o tareas junto con los productos generados.

Fase III. Acoplar el proceso de Implementación con los productos generados en el proceso de Análisis y Diseño del Método GRAY WATCH expandido.

En esta fase se acoplaron el proceso de implementación con los artefactos generados en el proceso de Análisis y Diseño del Método GRAY WATCH extendido. Para ello se consideró como base las actividades del proceso denominado **I**n**D**o**C**a**SE** (Rivero 2011).

Fase IV. Aplicación del método GRAY WATCH extendido al caso de estudio: Programa de Investigación UPEL-IPB.

Se aplicó la propuesta del método GRAY WATCH extendido al caso de estudio: Programa de Investigación UPEL-IPB.

CAPITULO IV

PROPUESTA DEL ESTUDIO

Luego de haber examinado los antecedentes y el marco teórico relacionado con esta investigación, se profundizó acerca de las características del método GRAY WATCH, así como las tareas y actividades de los procesos técnicos de análisis, diseño e implementación de la aplicación del método. Sobre la base del estudio realizado, en este capítulo se presenta la propuesta del método extendiendo sus procesos de análisis y diseño de la aplicación, basado en el estándar de calidad ISO/IEC 25010.

En primer término, se describe la justificación de la propuesta, seguidamente se definen los objetivos, generales y específicos que permitirán llevar a cabo la misma. Posteriormente se presenta la descripción de la propuesta, que especifica los procesos del método GRAY WATCH que van hacer modificados.

De igual manera, se detalla la estructura del modelo a seguir, tomando como referencia el proceso **I**n**D**o**C**a**S** (Canelón, 2010), y se determina que actividades y tareas se necesitan para lograr la extensión. Adicionalmente se muestra los diagramas de los procesos que son extendidos, basados en la nomenclatura de SPEM.

Finalmente se presenta el caso de estudio Programa de Investigación de la UPEL-IPB, y la aplicación del método GRAY WATCH extendido.

Justificación

El método GRAY WATCH diseñado por Montilva et al (2008), posee una base conceptual sólida y una metodología muy bien sustentada, emplea las mejores prácticas de desarrollo de software, entre ellas tiene: desarrollo a través de versiones,

promueve la reutilización de activos de software, manejo eficiente de los requisitos, desarrollo basado en modelos elaborados en lenguajes de modelado, aseguramiento de la calidad de los procesos y verificación y validación del Software.

En este propósito, se presenta una extensión del método GRAY WATCH en su más reciente versión (Montilva et al 2008), con la finalidad de incorporar atributos de calidad del producto basado en el estándar de calidad ISO/IEC 25010 en sus primeras etapas de desarrollo, para ello se utilizara el proceso **I**n**D**o**C**a**S** (Canelón, 2010), que aplica este estándar de calidad del producto.

El método GRAY WATCH lleva a cabo tres procesos técnicos para el desarrollo de software: Análisis, Diseño e Implementación de la aplicación, como se mencionó anteriormente esta investigación centra el interés en las primeras fases: los procesos de Análisis y Diseño de la Aplicación, lo que permite garantizar un diseño arquitectónico que satisfaga los requisitos de calidad asociados al estándar ISO/IEC 25010, y de esta manera se garantiza la selección de la arquitectura más adecuada y con atributos de calidad.

Finalmente, los productos resultantes se acoplan en los procesos de implementación. A continuación se presenta los objetivos de la propuesta.

Objetivos

Objetivo General

Diseñar la extensión del método GRAY WATCH en sus procesos de análisis y diseño de la Aplicación basado en el estándar de calidad ISO/IEC 25010.

Objetivos Específicos

1. Especificar las actividades y los productos que permitirán expandir los procesos de Análisis y Diseño del método GRAY WATCH aplicando el estándar de calidad ISO/IEC 25010 sobre la base del proceso **InDoCaS.**
2. Incorporar las actividades y los productos en los procesos de Análisis y Diseño del método GRAY WATCH aplicando el estándar de calidad ISO/IEC 25010.
3. Acoplar el proceso de implementación del método GRAY WATCH con los artefactos generados en los procesos extendidos de análisis y diseño considerando el proceso **InDoCaSE.**
4. Aplicar el método GRAY WATCH extendido al caso de estudio: Programa de Investigación UPEL-IPB.

Descripción de la Propuesta

En el capítulo 2 se presentó las características del método GRAY WACTH, que considera los procesos técnicos definidos en tres grupos: Procesos de Análisis, Procesos de Diseño y Procesos de Implementación. A continuación se muestra la cadena de procesos del método GRAY WATCH.

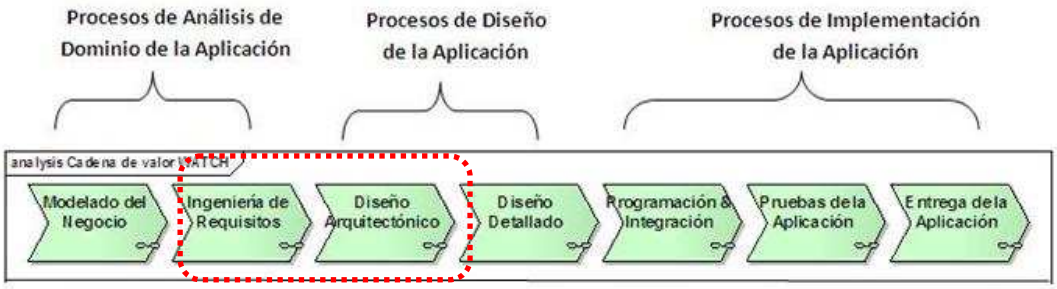


Figura 4.1 Cadena de procesos GRAY WATCH
Fuente: Montilva et al (2008)

De acuerdo a la figura anterior, la propuesta consiste en aplicar calidad en los procesos de Análisis, en este sentido se escoge el proceso ***Ingeniería de Requisitos***,

para ello, se identifican los requisitos funcionales y no funcionales y las propiedades de calidad asociadas y se construye el modelo de calidad según el estándar de calidad del producto ISO/IEC 25010.

De igual manera, la calidad de los Procesos de Diseño se enfoca en el proceso de ***Diseño Arquitectónico***, para ello se identifican estilos y patrones arquitecturales y selecciona la arquitectura que satisfaga los atributos de calidad. Finalmente los productos generados en el Procesos de Análisis y Diseño se acoplan en el Proceso de Implementación.

Para el diseño de la propuesta se utiliza como modelo a seguir el proceso **I**n**D**o**C**a**S** (Canelón, 2010), que dispone de actividades y artefactos en las disciplinas del análisis y diseño del dominio que servirán para aplicar calidad del producto basada en el estándar ISO/IEC 25010 al método GRAY WATCH.

También se utiliza el proceso **I**n**D**o**C**a**SE** (Rivero, 2011), que dispone de las actividades y artefactos necesarios para las disciplina de implementación del dominio. A continuación se presenta la estructura del modelo propuesto.

Estructura del modelo propuesto

Para la extensión del método GRAY WATCH se escogieron algunas de las actividades definidas en el proceso **I**n**D**o**C**a**S** (Canelón, 2010), que permiten obtener requisitos de calidad y una arquitectura basada en los requisitos de software y considera el estándar de calidad del producto ISO/IEC 25010.

La siguiente tabla muestra las actividades y los artefactos tomados del proceso **I**n**D**o**C**a**S** que se utilizaron para expandir el método GRAY WATCH.

InDoCaS Disciplinas	**ACTIVIDADES**	**ARTEFACTOS**
Análisis del dominio	**A_01** Identificación de Requisitos	**1.** Lista de Requisitos funcionales. **2.** Lista de Requisitos no funcionales
	A_03 Identificación de propiedades de calidad	**6.** Requisitos funcionales y propiedades de calidad asociadas. **7.** Requisitos no funcionales y propiedades de calidad asociadas.
	A_04 Obtener modelo de calidad asociado al dominio	**8.** Modelo de calidad del dominio
	A_06 Identificar estilos arquitecturales para el dominio.	**10.** Estilos arquitecturales
Diseño del dominio	**A_08** Escoger patrones arquitecturales candidatos	**12.** Patrones arquitecturales candidatos
	A_09 Instanciar elementos arquitecturales para los elementos del diseño del dominio	**13.** Elementos arquitecturales (componentes y conectores)
	A_12 Validar modelo de calidad del dominio con arquitecturas candidatas	**18.** Arquitecturas validadas a la familia
	A_11 Decidir la selección de la arquitectura como solución candidata	**20.** Informe sobre decisión arquitectural

Tabla 4.1 Actividades y artefactos tomados de **InDoCaS**
Fuente: (Canelón 2010)

Para describir las actividades y los artefactos se utiliza la nomenclatura de **InDoCaS**, para representar los diagramas de actividad que muestran la extensión de las actividades y artefactos en los subprocesos del método GRAY WATCH se utiliza la diagramación mediante SPEM**.**

Adicionalmente, para identificar las actividades que son incorporadas a los subprocesos del método GRAY WATCH se enmarcan en un círculo con líneas punteadas verdes. A continuación se presenta dos secciones que describen las

actividades y artefactos a incorporarse en los procesos de Análisis y Diseño de la aplicación de método GRAY WATCH.

1. ACTIVIDADES Y ARTEFACTOS EN EL PROCESO DE ANALISIS

El proceso de Análisis del método GRAY WATCH está constituido por dos procesos: Modelado de Negocios e Ingeniería de Requisitos. Para efectos de la extensión se toma el proceso de **Ingeniería de Requisitos** específicamente el subproceso de **Análisis de los requisitos**, que se encarga de definir y especificar el conjunto de requisitos funcionales y no funcionales que la aplicación debe satisfacer. A continuación se muestra el flujo de trabajo de este subproceso:

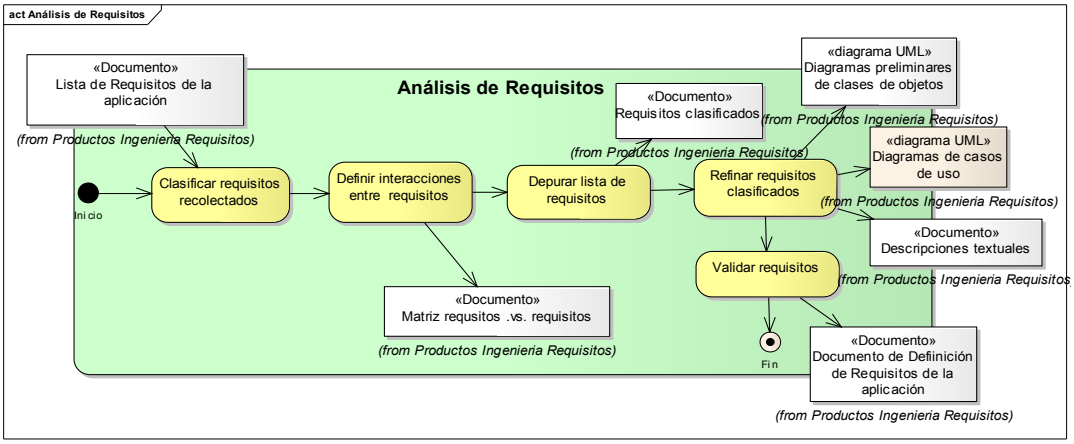


Figura 4.2 Diagrama de actividades subproceso Análisis de Requisitos

Fuente: Montilva et al (2008)

La figura anterior muestra las actividades que se realizan dentro del subproceso de Análisis de requisitos. Para la extensión se hace necesario incorporar la actividad "Identificación de requisitos", y fusionarla con la actividad "clasificar requisitos recolectados", que permite identificar los requisitos funcionales y no funcionales.

Con la nueva actividad se aplica el modelo de clasificación de requisitos RECLAMO (Chirinos et al, 2004), que guía la clasificación de los requisitos funcionales que se derivan de los requerimientos de los usuarios y representan la

funcionalidad del sistema, y los requisitos no funcionales se derivan del dominio del problema, del ambiente de ejecución del sistema, de los requisitos de definición de datos y de las reglas del negocio.

A continuación se muestra las tablas con las actividades y artefactos que se tomaron de **InDoCaS** y se incorporan y acoplan al método WATCH en el subproceso "**Análisis de Requisitos"**

ACTIVIDAD	**DESCRIPCION**
Nombre	*Identificar requisitos*
Actor	Analista de requisitos
Objetivo	Identificar requisitos funcionales y no funcionales
Técnica(s) utilizada(s)	RECLAMO
Artefactos de entrada	Lista de requisitos de la aplicación
Artefactos de salida	Requisitos funcionales y no funcionales
Recuperada de InDoCaS. (ref. A-01).	Requerida para clasificar los requisitos en Funcionales y No funcionales aplicando el modelo de clasificación de RECLAMO.

Tabla 4.2 ACTIVIDAD Identificar Requisitos
Fuente: el autor

ARTEFACTO	**DESCRIPCION**
Nombre	Requisitos Funcionales de la aplicación
Constructor	Analista de requisitos
Formato asociado	Adaptado de **InDoCaS**.(**ref. 1)**

Nro. Identificación	**Requisitos Funcionales**

Tabla 4.3 ARTEFACTO Requisitos funcionales **DISEÑO**
Fuente: el autor

ARTEFACTO	DESCRIPCION
Nombre	Requisitos No Funcionales de la aplicación
Constructor	Analista de requisitos
Formato asociado	Adaptado de **InDoCaS**.(**ref. 2)**

Nro.	Reglas del negocio asociadas al dominio	Requisitos no Funcionales derivados de las reglas del negocio
	Políticas	
	Procesamiento	
	Implementación	

Tabla 4.4 ARTEFACTO Requisitos no funcionales
Fuente: el autor

Luego de incorporar la actividad y los artefactos anteriormente descritos dentro del subproceso Análisis de Requisitos del método GRAY WATCH se representa a continuación el diagrama de actividad acoplado con los artefactos de entrada y salida.

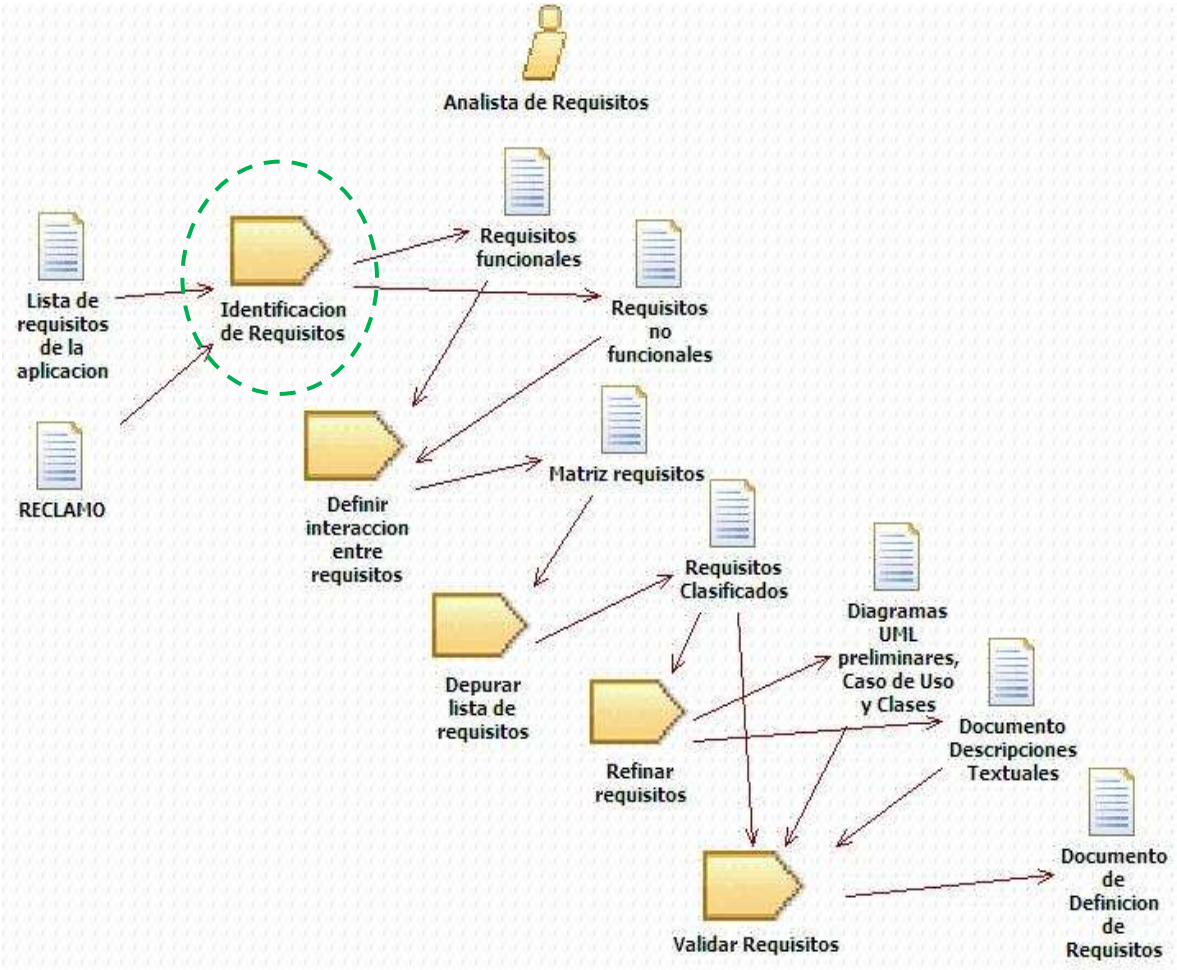


Figura 4.3 Diagrama de Actividades del Subproceso Análisis de Requisitos método GRAY WATCH

Fuente: el autor

En el diagrama anterior se observa el primer subproceso de los procesos de Análisis de la aplicación del método GRAY WATCH extendido, la incorporación de la actividad "**Identificación de requisitos"** permitió clasificar los requisitos en funcionales y no funcionales de acuerdo a RECLAMO (Chirinos et al 2004). Ahora bien, se procede a modificar y acoplar los subprocesos del proceso de Diseño de la aplicación del método GRAY WATCH, como se muestra a continuación.

2. ACTIVIDADES Y ARTEFACTOS DEL PROCESO DE DISEÑO

El proceso de Diseño Arquitectónico permite establecer el conjunto de componentes que integraran la aplicación, las relaciones y restricciones de interacción entre ellos, las relaciones con otras aplicaciones externas y la distribución física de cada uno de estos componentes. (Montilva et al 2008) Para elaborar el diseño de la arquitectura se requiere de la ejecución de cuatro subprocesos que son los siguientes:

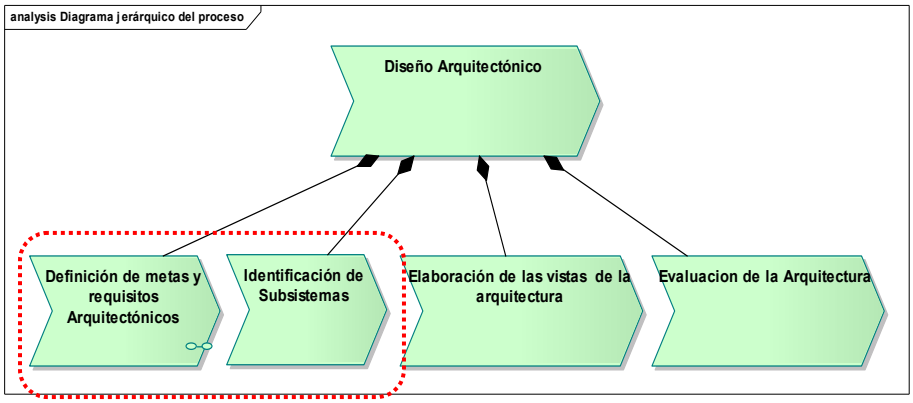


Figura 4.4 Subprocesos del Diseño Arquitectónico
Fuente: Montilva et al (2008)

En el diagrama anterior se muestran seleccionados los dos subprocesos del proceso Diseño arquitectónico que fueron extendidos, además de ello fue necesario realizar una reingeniería de los subprocesos para acoplar las actividades y los artefactos de acuerdo al objetivo determinado que es aplicar calidad del producto mediante el ISO/IEC 25010. A continuación explicaremos detalladamente cómo se incorporo las actividades y artefactos del proceso InDoCaS a cada uno de los subprocesos: Definición de metas y requisitos arquitectónicos e Identificación de subsistemas.

1) Subproceso Definición de metas y requisitos arquitectónicos

Inicialmente se muestra un diagrama con las tareas que se realizan en este subproceso.

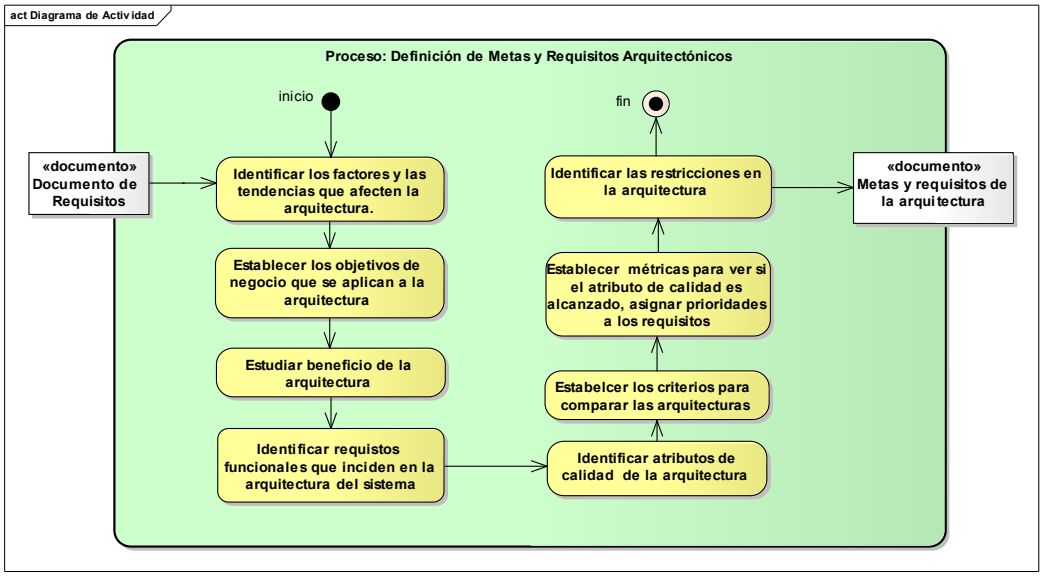


Figura 4.5: Diagrama del subproceso definición de metas y requisitos arquitectónicos.
Fuente: Montilva et al (2008)

Como se observa en la figura 4.5, este subproceso permite identificar los requisitos funcionales y no funcionales que estén relacionados o incidan en la selección de la arquitectura de software de la aplicación. Sin embargo, requiere de la incorporación de actividades que permitan identificar los atributos de calidad basados en la norma de calidad ISO/IEC 25010, para este propósito se identifican características y subcaracterísticas para cada uno de los requisitos suministrados.

También se agrega a este subproceso la identificación de los estilos arquitecturales y la selección de los patrones arquitectónicos que satisfacen los objetivos y el modelo de calidad, de manera que se define el diseño inicial de la arquitectura. En base a estas consideraciones, que especifican el diseño arquitectónico inicial el proceso ahora se denomina "**definición de diseño de arquitectura"**.

A continuación se muestra las tablas de las actividades que son incorporadas con los respectivos artefactos:

ACTIVIDAD	DESCRIPCION
Nombre	*Identificación de propiedades de calidad*
Actor	Analista de requisitos
Objetivo	Identificar propiedades de calidad para requisitos funcionales y no funcionales
Técnica(s) utilizada(s)	Identificar características y subcaracterísticas de calidad usando el modelo de referencia ISO/IEC 25010.
Artefactos de entrada	Requisitos Funcionales y No Funcionales
Artefactos de salida	- Requisitos funcionales y propiedades de calidad. - Requisitos no funcionales con propiedades de calidad.
Recuperada de InDoCaS. (ref. A-03).	Permite obtener los requisitos de calidad mediante la aplicación del estándar ISO/IEC 25010.

Tabla 4.5 ACTIVIDAD Identificación de propiedades de calidad
Fuente: el autor

ARTEFACTO	DESCRIPCION
Nombre	Requisitos Funcionales y propiedades de calidad
Constructor	Analista de requisitos
Formato asociado	Adaptado de **InDoCaS**.(**ref. 6)**

Nro.	Requisitos funcionales	Propiedades de calidad asociadas ISO/IEC 250210

Tabla 4.6 ARTEFACTO Requisitos funcionales y propiedades de calidad
Fuente: el autor

<table>
<tr><th>ARTEFACTO</th><th>DESCRIPCION</th></tr>
<tr><td>Nombre</td><td>Requisitos no Funcionales y propiedades de calidad</td></tr>
<tr><td>Constructor</td><td>Analista de requisitos</td></tr>
<tr><td colspan="2">Formato asociado Adaptado de InDoCaS.(ref. 7)
<table>
<tr><th>Nro. Identificación</th><th>Reglas del negocio</th><th>Requisitos no funcionales derivados de las reglas de negocio</th><th>Propiedades de calidad asociadas ISO/IEC 25010</th></tr>
<tr><td></td><td>Políticas</td><td></td><td></td></tr>
<tr><td></td><td>Procesamiento</td><td></td><td></td></tr>
<tr><td></td><td>Implementación</td><td></td><td></td></tr>
</table>
</td></tr>
</table>

Tabla 4.7 ARTEFACTO Requisitos no funcionales y propiedades de calidad
Fuente: el autor

ACTIVIDAD	DESCRIPCION
Nombre de la actividad	*Obtener modelo de calidad*
Actor	Analista de requisitos
Objetivo	Identificar las características y subcaracterísticas del modelo de calidad
Técnica(s) utilizada(s)	Identificar características y subcaracterísticas de calidad usando el modelo de referencia ISO/IEC 25010
Artefactos de entrada	Requisitos de la aplicación y propiedades de calidad, ISO/IEC 25010.
Artefactos de salida	Modelo de calidad como instancia de ISO/IEC 25010
Recuperada de InDoCaS. (ref. A-04).	Se requiere para conocer los atributos de calidad según el modelo de ISO/IEC 25010.

Tabla 4.8 ACTIVIDAD Obtener modelo de calidad
Fuente: el autor

ARTEFACTO	DESCRIPCION
Nombre	Modelo de Calidad
Constructor	Analista de requisitos
Formato asociado: Según estándar ISO/IEC 25010	Adaptado de **InDoCaS.(ref. 8)**

Tabla 4.9 ARTEFACTO Modelo de calidad
Fuente: el autor

ACTIVIDAD	DESCRIPCION
Nombre de la actividad	*Revisar arquitecturas y estilos arquitectónicos*
Actor	Arquitecto de Software
Objetivo	Revisar otras arquitecturas y estilos arquitecturales
Técnica(s) utilizada(s)	Usar el modelo de calidad y de acuerdo a la lista de requisitos identificar y aportar soluciones arquitecturales
Artefactos de entrada	Requisitos funcionales y no funcionales, Estilos arquitecturales, experticia del arquitecto.
Artefactos de salida	Estilos arquitecturales
Método GRAY WATCH (acoplada)	

Tabla 4.10 ACTIVIDAD Revisar arquitecturas y estilos arquitectónicos
Fuente: el autor

ARTEFACTO	DESCRIPCION
Nombre	Estilos arquitecturales
Constructor	Arquitecto de Software
Formato asociado: ADL, UML	**Método GRAY WATCH**

Tabla 4.11 ARTEFACTO Estilos arquitecturales
Fuente: el autor

ACTIVIDAD	DESCRIPCION
Nombre de la actividad	*Escoger patrones arquitecturales*
Actor	Arquitecto de Software
Objetivo	Revisar los patrones arquitecturales como una posible solución a los requisitos de calidad
Técnica(s) utilizada(s)	Evaluar la aplicación de los patrones a cada elemento de los requisitos de calidad
Artefactos de entrada	Requisitos funcionales y no funcionales, Estilos arquitecturales, catalogo de patrones arquitecturales
Artefactos de salida	Patrones arquitecturales candidatos
Recuperada de InDoCaS. (ref. A-08).	Se utiliza para escoger los patrones necesarios para la arquitectura.

Tabla 4.12 ACTIVIDAD Escoger Patrones Arquitecturales
Fuente: el autor

ARTEFACTO	DESCRIPCION
Nombre	Patrones arquitecturales candidatos
Constructor	Arquitecto de Software

Formato asociado: Recuperado de **InDoCaS**.(**ref. 12)**

Nro Patrón	Nombre	Descripción	Favorece	Componentes	Conectores	Ejemplo

Tabla 4.13 ARTEFACTO Patrones arquitecturales candidatos
Fuente: el autor

ACTIVIDAD	DESCRIPCION
Nombre de la actividad	*Instanciar elementos arquitecturales*
Actor	Arquitecto de Software
Objetivo	Se seleccionan patrones y tipos de componentes que satisfacen los requisitos
Técnica(s) utilizada(s)	Se muestran las alternativas de diseño y se escoge el patrón
Artefactos de entrada	Requisitos funcionales y no funcionales, Estilos arquitecturales, patrones candidatos
Artefactos de salida	Elementos arquitecturales
Recuperada de InDoCaS. (ref. A-09).	Se requiere para adaptar los patrones a los requisitos funcionales y no funcionales.

Tabla 4.14 ACTIVIDAD Instanciar Elementos arquitecturales
Fuente: el autor

ARTEFACTO	DESCRIPCION
Nombre	Elementos arquitecturales
Constructor	Arquitecto de Software

Formato asociado: Recuperado de **InDoCaS**.(**ref. 13)**

Elementos de diseño	Características de Calidad ISO/IEC 25010	Nro. De Patrón	Nombre del patrón escogido

Tabla 4.15 ARTEFACTO Elementos arquitecturales
Fuente: el autor

Luego de detallar las actividades y los artefactos que serán incorporadas dentro del subproceso "**Definición de diseño de la arquitectura"** del método GRAY WATCH se procede a integrarlas y acoplarlas. Se presenta a continuación el diagrama de actividad con las actividades y artefactos integrados y acoplados:

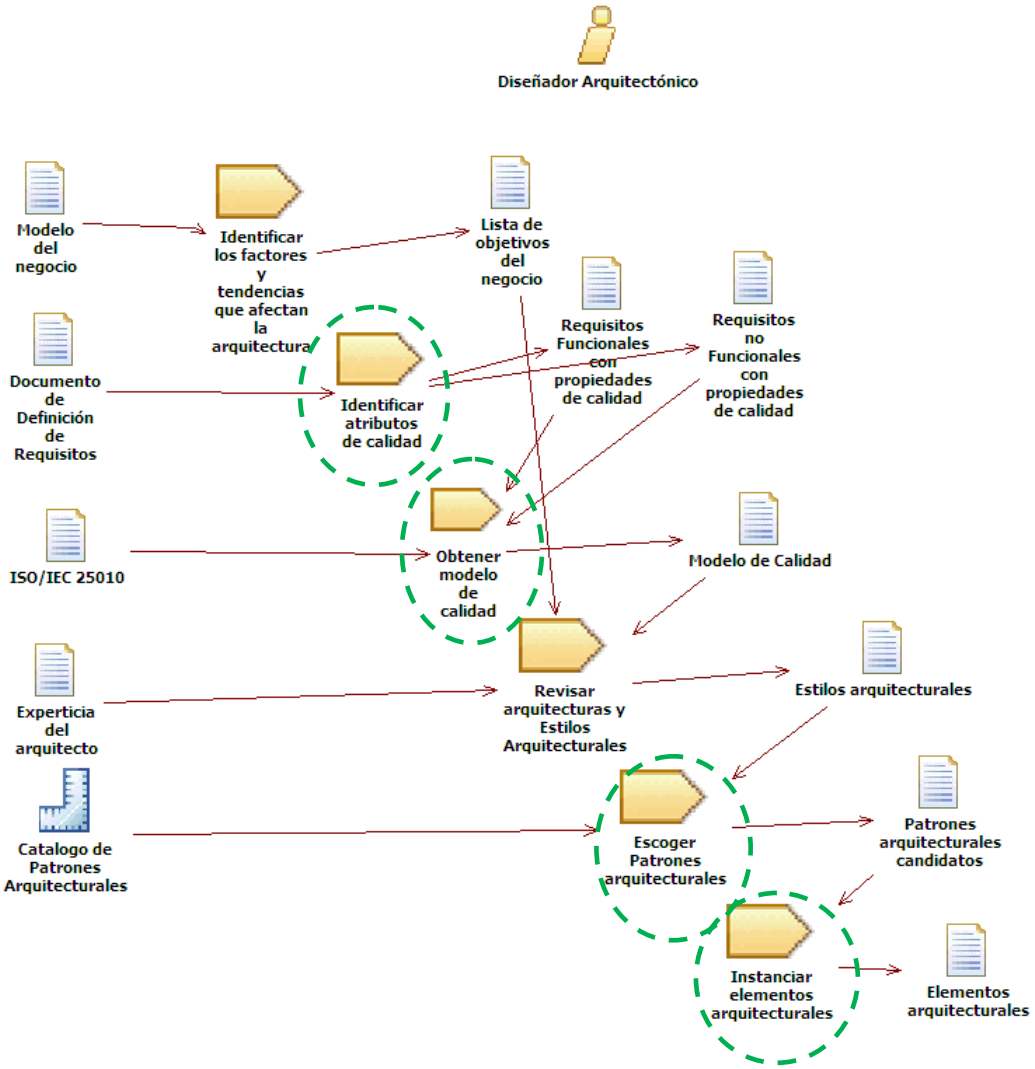


Figura 4.7 Diagrama de Actividades del Subproceso Definición de diseño de arquitectura del método GRAY WATCH

Fuente: el autor

En la figura anterior se observa el subproceso **"definición de diseño de arquitectura"** del método GRAY WATCH extendido, se incorporaron tres actividades que fueron tomadas del proceso **InDoCaS** y se modificaron y acoplaron el resto de las actividades y artefactos. De ello, se genera un modelo de calidad en base a los atributos de calidad según el estándar de calidad ISO/IEC 25010, este modelo se utiliza para la elección de los estilos arquitecturales y para escoger los patrones arquitectónicos que mejor se adapten.

2) Subproceso Identificación de Subsistemas

Inicialmente se muestra un diagrama con las tareas que se realizan en este subproceso.

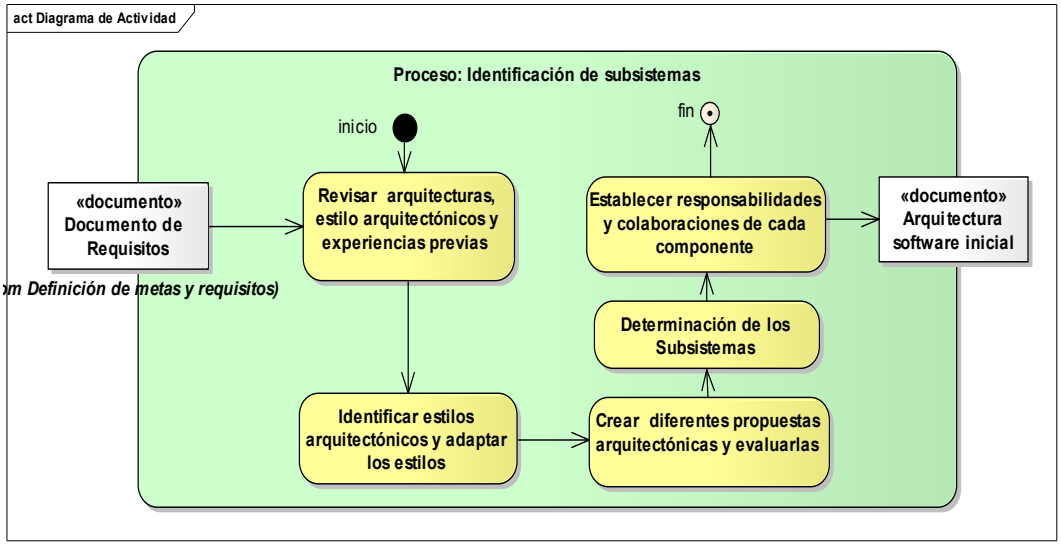


Figura 4.6: Diagrama del subproceso Identificación de subsistemas

Fuente: Montilva et al (2008)

Como se observa en la figura 4.6 el subproceso Identificación de subsistemas se encarga de revisar estilos arquitecturales y patrones arquitecturales, crear propuestas arquitectónicas y en base a ello dividir el sistema en subsistemas y describir cada uno de ellos, sus componentes e interacciones con otros subsistemas. Para extender este subproceso, se hace necesario agregar las actividades "validar soluciones con modelo de calidad" y "Decidir la solución arquitectural" del proceso **InDoCaS** lo que permitirá validar las propuestas arquitectónicas generadas y posteriormente seleccionar la mejor propuesta que satisfaga el conjunto de requisitos y el modelo de calidad.

Por esta razón, el subproceso extendido se le denomina "**selección de la arquitectura".** A continuación se muestra las tablas de las actividades que son incorporadas con los respectivos artefactos.

ACTIVIDAD	DESCRIPCION
Nombre	*Validar soluciones con modelo de calidad*
Responsable	Arquitecto de Software
Objetivo	Validar el cumplimiento del modelo de calidad en las soluciones arquitecturales
Técnica(s) utilizada(s)	
Artefactos de entrada	Modelo de Calidad, conjunto de soluciones arquitecturales.
Artefactos de salida	Arquitecturas validadas
Recuperada de InDoCaS. (Ref. A-12).	

Tabla 4.16 ACTIVIDAD Validar soluciones con modelo de calidad
Fuente: el autor

ARTEFACTO	DESCRIPCION
Nombre	Arquitectura validadas
Constructor	Arquitecto de Software
Formato asociado: ADL, UML	

Tabla 4.17 ARTEFACTO Arquitecturas validadas
Fuente: el autor

ACTIVIDAD	DESCRIPCION
Nombre	*Decidir la selección de la arquitectura como solución arquitectural*
Actor	Arquitecto de Software
Objetivo	Escoger arquitectura para la aplicación
Técnica(s) utilizada(s)	Evaluar resultados de cumplimiento del modelo de calidad y escoger la mejor arquitectura
Artefactos de entrada	Estilos arquitecturales, elementos arquitecturales y arquitecturas validadas
Artefactos de salida	Informe de Arquitectura de Software
Recuperada de InDoCaS. (Ref. A-11).	

Tabla 4.18 ACTIVIDAD Decidir la selección de la arquitectura como solución arquitectural
Fuente: el autor

ARTEFACTO	DESCRIPCION
Nombre	Informe de Arquitectura de software
Constructor	Arquitecto de Software
Formato asociado: ADL,UML, documento **Método GRAY WATCH**	

Tabla 4.19 ARTEFACTO Informe de Arquitectura de software
Fuente: el autor

Luego de detallar las actividades y los artefactos que son incorporados dentro del subproceso "**Selección de la arquitectura"** del método GRAY WATCH se presenta a continuación el flujo de trabajo con las actividades y artefactos integrados y acoplados:

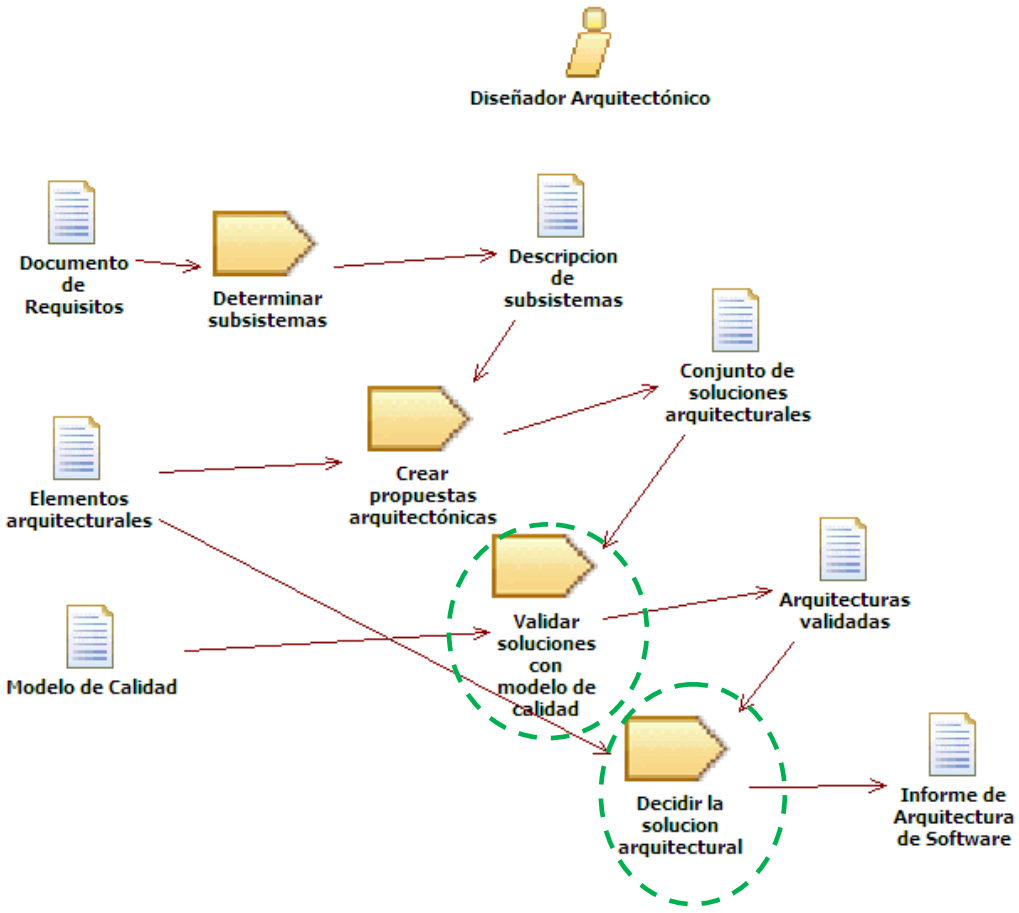


Figura 4.8 Diagrama de Actividades del Subproceso Seleccionar arquitectura
Fuente: el autor

En el diagrama anterior se muestra el flujo de trabajo del subproceso **"selección de la arquitectura"** del método GRAY WATCH, que obtiene el informe con la arquitectura de software del sistema, que es el artefacto más importante que permite observar el sistema desde diversos puntos de vista.

En síntesis, el proceso de diseño arquitectónico guía la selección de la arquitectura del sistema, mediante la identificación de características y subcaracteristicas del modelo de calidad ISO/IEC 25010 con respecto a los requisitos funcionales y no funcionales del sistema, luego se observan diferentes estilos arquitecturales y patrones arquitectónicos y se crean diversas propuestas de arquitectura que posteriormente son validadas con el modelo de calidad, para escoger la que mejor se adapte.

A continuación se muestra el diagrama de Diseño arquitectónico modificado con los dos subprocesos que se modificaron:

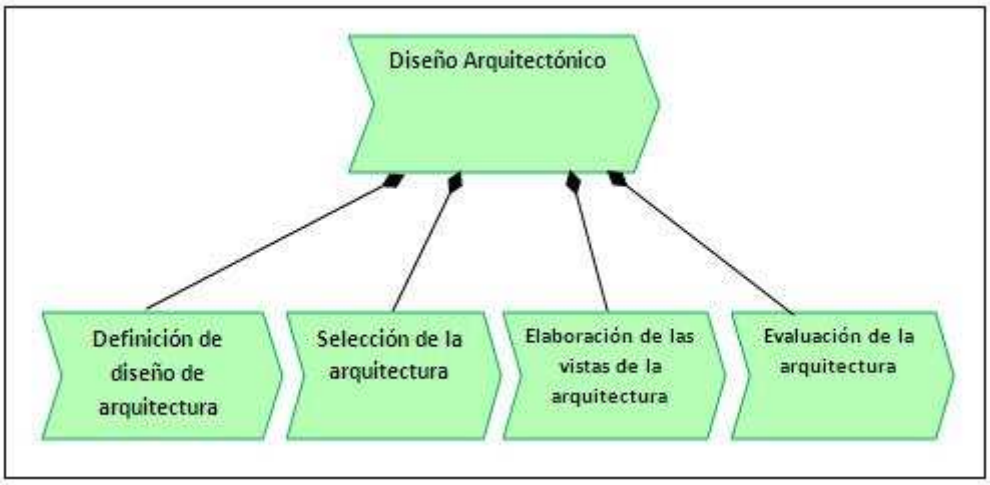


Figura 4.9 Subprocesos del Diseño Arquitectónico modificado
Fuente: el autor

3. ACOPLAMIENTO DE LOS PRODUCTOS GENERADOS EN LOS PROCESOS DE ANALISIS Y DISEÑO EN EL PROCESO DE IMPLEMENTACION DEL METODO GRAY WATCH EXTENDIDO.

En las secciones anteriores se pudo constatar que los productos intermedios generados en los procesos de análisis y diseño del método GRAY WATCH se ajustaron dentro de cada subprocesos de acuerdo a su uso. Por otra parte el producto final generado en el proceso técnico de **Diseño Arquitectónico**, Documento con la Arquitectura de Software, junto con el Documento de Especificación de Requisitos generado en el proceso técnico **Ingeniería de Requisitos** son los artefactos de entrada para el subproceso de **Diseño Detallado** que especifica las características que tiene cada uno de los componentes de la aplicación, la interfaz usuario/sistema y el modelo de datos que se implementara.

Los procesos técnicos de Diseño de la aplicación finalizan generando un Documento que tiene todas las especificaciones de diseño tanto de los componentes, como de la interfaz y de la base de datos y que es el artefacto de entrada a los procesos de implementación del método GRAY WATCH extendido.

Los procesos de implementación del método GRAY WATCH involucran los procesos de Programación e Integración, Pruebas de la Aplicación y Entrega de la Aplicación, (Montilva et al 2008). Los componentes se codifican o se adaptan, se prueban y se integran. Los artefactos de entrada son el documento de requisitos y el documento de diseño con todas las especificaciones, que guían el proceso de aprovisionamiento de los componentes que se encarga de buscar, adquirir, adaptar o codificar los componentes de software. Una vez que estos componentes se elaboren o se adapten, según sea el caso, se procede a probarlos separadamente usando estrategias, técnicas y herramientas de pruebas de unidades.

De esta manera quedan acoplados los productos generados en los procesos técnicos de análisis y diseño de la aplicación del método GRAY WATCH extendido.

CASO DE ESTUDIO: Programa de Investigación de la UPEL-IPB

El Programa de Investigación de la UPEL- IPB está constituido por los proyectos y actividades concebidos para el logro de los objetivos institucionales de investigación. El propósito de este programa es gestionar el desarrollo de proyectos, actividades y eventos de investigación de alto nivel científico que posibiliten la excelencia académica a la vez que la obtención de resultados de alto impacto que contribuyan al desarrollo económico, social, tecnológico y cultural del país. (UPEL, 1998).

El investigador realizo una exploración mediante la observación directa y una entrevista no estructurada para conocer el programa de Investigación, de esta manera se pudo observar, que los procesos administrativos y académicos no están automatizados, y que la Unidad no dispone de herramientas tecnológicas para gestionar el desarrollo de proyectos y actividades de investigación, lo que trae como consecuencias retardos en la ejecución de las tareas de la Unidad y que no haya una garantía de la integridad, confiabilidad y coherencia de los datos.

Por otra parte, se evidenció la necesidad que tiene actualmente el programa de proporcionar información de estadísticas a otras unidades de la Institución, tales como Planificación, Ejecución presupuestaria, Coordinación de Posgrado, Vicerrectorado de Investigación y Posgrado, entre otros. Aunado a ello, los organismos del Estado como MPPES, OPSU y MCT periódicamente solicitan información de los investigadores y de los proyectos de investigación realizados en la institución, y se hace difícil enviar dicha información, pues no se dispone de los mecanismos de comunicación necesario.

En base a la problemática anteriormente expuesta, la presente investigación pretende proporcionar un aporte al Programa de Investigación de la UPEL-IPB, diseñando un modelo de software que gestione los procesos académicos administrativos del programa, y permita emitir información acerca de las actividades de investigación que se realizan.

Así pues, se requiere de una aplicación que de soporte a los procesos del negocio mediante una interfaz web que permita el intercambio de datos e información a través de una red Intranet, extranet o internet. La aplicación debe satisfacer los requerimientos tanto de nivel operativo, táctico como estratégico de las autoridades institucionales, sus investigadores y de los diferentes actores interesados en el trabajo investigativo y sus resultados. La aplicación debe gestionar los procesos de negocios en el área de Investigación y generar los indicadores de gestión requeridos. Dicho software ofrecerá servicios relacionados con la labor investigativa a las diferentes unidades internas de la institución y permitirá a los organismos del Estado como MPPES, OPSU y MCT tomar la información requerida en cuanto a los indicadores de productividad de los investigadores y de sus actividades investigativas.

A nivel estratégico el sistema aspira disponer de un conjunto de indicadores de gestión y de estadísticas relacionadas con las capacidades y resultados de la investigación que se desarrolla en la Universidad.

El sistema debe ofrecer una gama de servicios de información de utilidad exclusiva de determinados usuarios en base a su perfil ejecutivos, investigadores, visitantes y otros, respecto de programas, proyectos, eventos y productos resultantes de las actividades de investigación desarrolladas en la institución.

A continuación se hace uso del método GRAY WATCH extendido, donde se ejecutan los subprocesos: Análisis de Requisitos, Definición de diseño de la arquitectura y selección de la arquitectura con la finalidad de validar los subprocesos que han sido extendidos. Para ello se utiliza el caso de estudio: Programa de Investigación de la UPEL-IPB, descrito al comienzo de esta sección.

SUBPROCESO: ANALISIS DE REQUISITOS

En este subproceso se ejecutó la actividad **Identificación de requisitos**, para ello se aplicó el modelo RECLAMO (Chirinos et al., 2004) que facilitó la clasificación de los requisitos en funcionales y no funcionales (de acuerdo a las reglas

del negocio). Así pues, se generó dos artefactos que se muestran en las siguientes tablas:

ARTEFACTO	**Requisitos Funcionales de la aplicación**
Constructor	Analista de requisitos
Actividad que lo genera	Identificar requisitos

Nro. Identificación	**Requisitos Funcionales**
1	Ofrecer una gama de servicios de información de utilidad exclusiva a determinados usuarios en base a su perfil directivos, investigadores, administrativos y otros
2	Gestionar el registro y seguimiento de proyectos de investigación al personal de investigación.
3	Emitir reportes sobre los proyectos de investigación: registrados, en desarrollo y culminados.
4	Gestionar las actividades de investigación realizadas por los investigadores de la Institución.
5	Emitir un conjunto de indicadores de gestión y de estadísticas sobre las actividades de investigación.
6	Proporcionar servicios de información de estadísticas de investigadores y de las actividades de investigación a otros entes internos de la Universidad.
7	Suministrar información de los investigadores activos y de los proyectos de investigación a los organismos del Estado como MPPES, OPSU y MCT.

Tabla 4.20 ARTEFACTO Requisitos funcionales Caso de estudio: Programa de Investigación UPEL-IPB

Fuente: el autor

ARTEFACTO	**Requisitos No Funcionales de la aplicación**

Constructor	Analista de requisitos
Actividad que lo genera	Identificación de requisitos

Nro.	Reglas del negocio asociadas al dominio	Requisitos no Funcionales derivados de las reglas del negocio
	Políticas	
1	Seguridad en el acceso al sistema	La aplicación debe ofrecer mecanismos de seguridad, que permita solo a los usuarios registrados acceder a las funcionalidades del sistema, que le corresponde según su rol.
2	El sistema debe estar disponible en la red de la Institución.	Garantizar el servicio en la red, considerar ancho de banda y protocolos de comunicación.
	Procesamiento	
3	Los datos deben ser consistentes.	- El sistema deberá almacenar todos los datos en una base de datos SQL estándar, donde pueda ser accesado por otros programas. - Debe contemplar requerimientos de consistencia transaccional. Ante la falla del aplicativo, se debe contar con mecanismos que contemplen la interrupción de transacciones para que estas finalicen de manera correcta. - En caso de fallas no debe haber pérdida de la información.
4	La solución debe ofrecer adecuados niveles de servicio donde la disponibilidad y recuperación de fallos sea garantizada.	El sistema debe tener tolerancia a fallas(el sistema debe mantenerse operativo y recuperarse de posibles fallas)
	Implementación	
5	El sistema debe permitir cambios sin afectar el rendimiento del mismo.	El sistema debe permitir la incorporación de nuevos componentes sin que causen un impacto en la ejecución de la aplicación.
6	La interacción entre el usuario y el sistema debe ser amigable y sencilla.	La aplicación debe tener una interfaz sencilla y fácil de usar.
7	Capacidad de interacción con otros sistemas de la Institución y externos	La aplicación debe soportar la capacidad de comunicarse con sistemas externos a nivel de datos y procesos.

Tabla 4.21 ARTEFACTO Requisitos no funcionales Caso de estudio: Programa de Investigación UPEL-IPB

Fuente: el autor

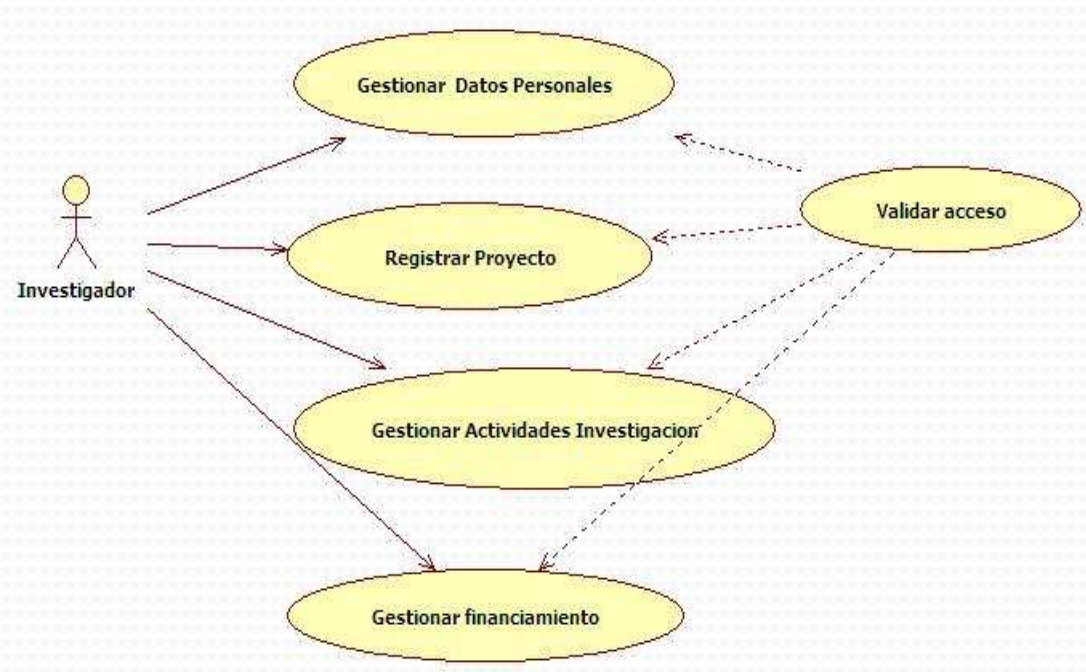
ARTEFACTO	**Diagramas UML**

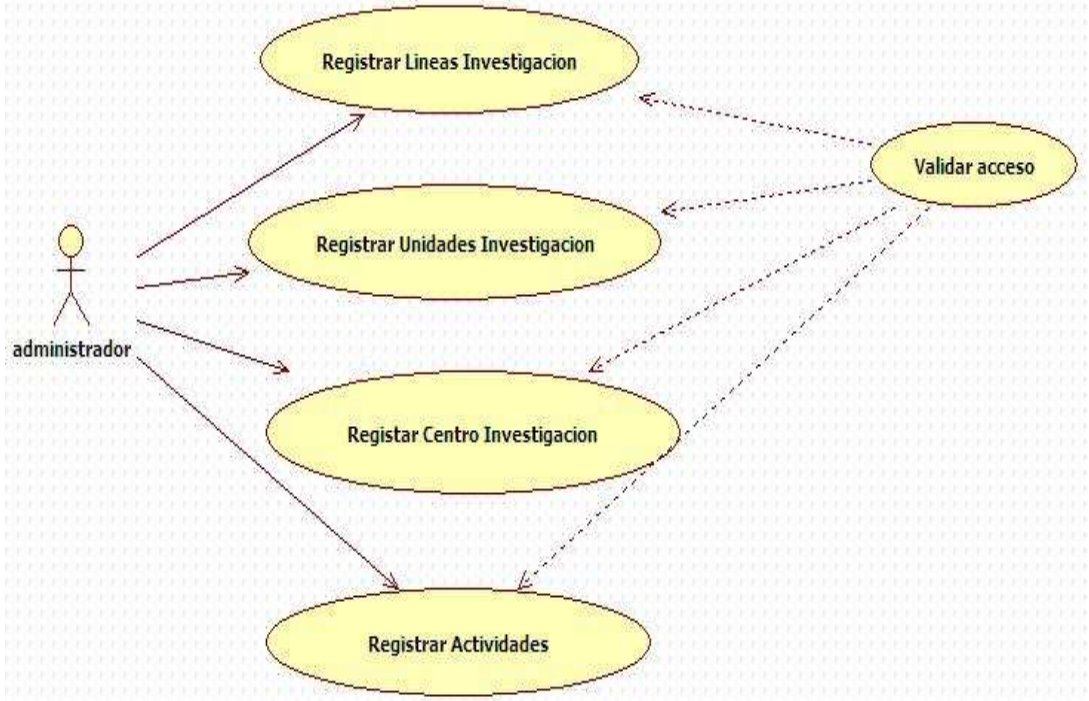
Constructor	Analista de requisitos
Actividad que lo genera	Refinar requisitos clasificados

Luego de obtener los requisitos clasificados se procede a describir en mayor detalle los requisitos para ello se elaboran los diagramas de caso de uso para explorar la funcionalidad y definir escenarios. Por otra parte se elaboran diagramas de clases de objetos requeridos para la funcionalidad.

Los actores son los siguientes:

A continuación se presentan algunos Casos de Uso:




Registrar Lineas Investigacion
Validar acceso
Registrar Unidades Investigacion
administrador
Registar Centro Investigacion
Registrar Actividades

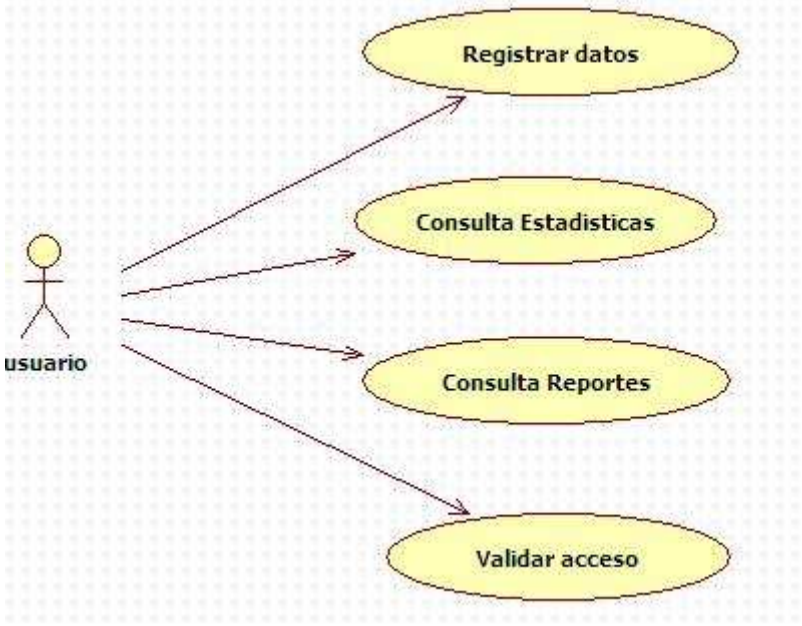

Registrar datos
Consulta Estadisticas
usuario
Consulta Reportes
Validar acceso

Diagrama de Clases:

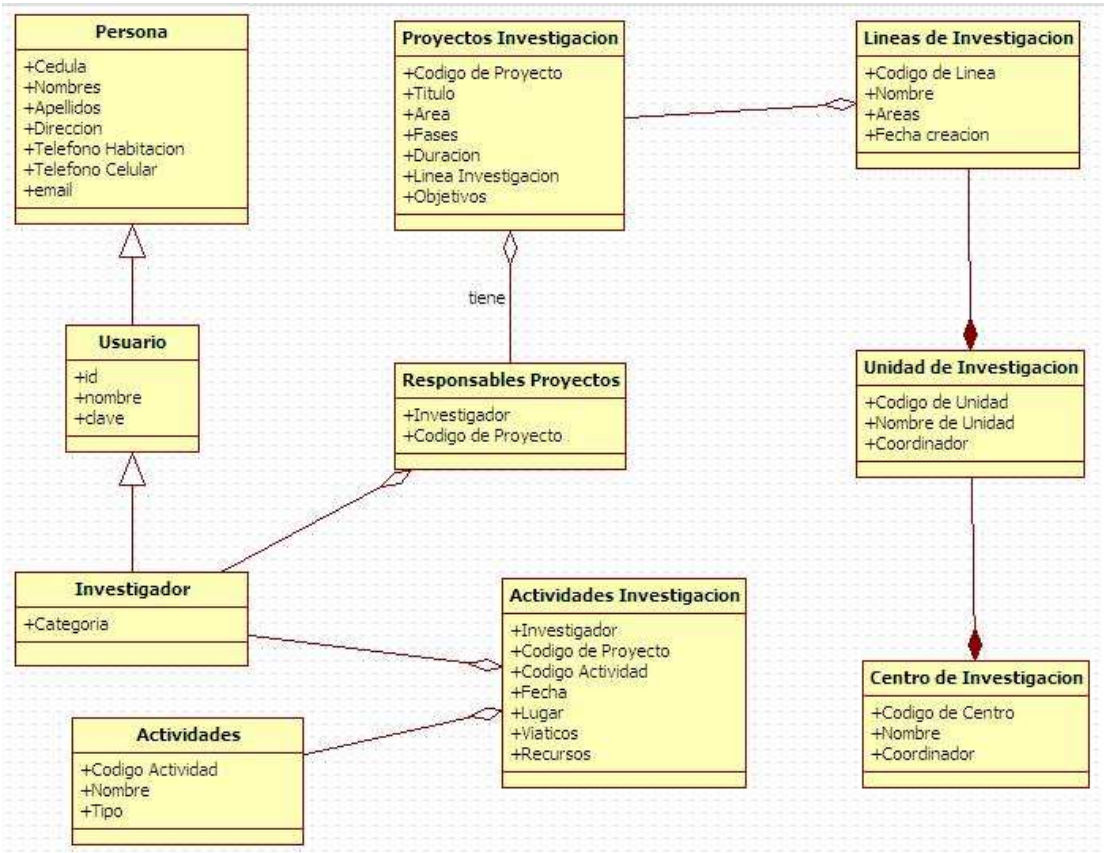


Tabla 4.22 ARTEFACTO Diagramas UML Caso de estudio: Programa de Investigación UPEL-IPB

Fuente: el autor

SUBPROCESO: DEFINICION DE DISEÑO DE LA ARQUITECTURA

En este subproceso se ejecutaron las siguientes actividades:

Subproceso	ACTIVIDADES	ARTEFACTOS
Diseño de la Arquitectura	Identificar los factores y las tendencias que afectan la arquitectura.	**Tabla 4.24**
	Identificar los atributos de calidad	**Tabla 4.25 y 4.26**
	Obtener modelo de calidad	**Tabla 4.27**
	Revisar arquitecturas y estilos arquitecturales.	**Tabla 4.28**
	Escoger patrones arquitecturales.	**Tabla 4.29**
	Instanciar elementos arquitecturales.	**Tabla 4.30**

Tabla 4.23: Actividades ejecutadas con sus artefactos. Subproceso Definición Diseño de la Arquitectura
Fuente: el autor

De estas actividades se desprende, las lista de las incidencias en el diseño de la arquitectura, también se identifican las características y subcaracterísticas de calidad de cada requisito basándose en el estándar de calidad ISO/IEC 25010. Luego, se genera un modelo de calidad que se construye a partir de los atributos de calidad obtenidos en la actividad previa. Finalmente este modelo se utiliza para definir los estilos arquitecturales, en este caso el modelo seleccionado fue Modelos en Capas, que permite organizar el sistema en capas y cada capa proporciona un servicio, esta arquitectura soporta el desarrollo incremental que usa el método GRAY WATCH, ayuda a la mantenibilidad y a la portabilidad de la aplicación (Sommerville,2005).

A continuación se presentan los artefactos generados:

ARTEFACTO	Lista de objetivos del negocio
Constructor	Arquitecto de Software
Actividad que lo genera	Identificar los factores y las tendencias que afectan la arquitectura.

1. El Programa de Investigación de la UPEL-IPB requiere de una aplicación web que gestione los procesos del negocio del Programa y automatice el flujo de trabajo de los procesos que soporta, facilitando el acceso a los datos usando interfaces graficas basadas en tecnología web.
2. La interfaz del usuario deberá ser tan familiar, práctica e intuitiva como sea posible a los usuarios que han usado otras aplicaciones Web.
3. La aplicación Web tiene que ser completamente escalable sin que un aumento de los recursos dedicados a la misma suponga modificación alguna en su comportamiento o capacidades
4. La aplicación debe operar en la intranet de la Institución y vía Internet
5. El acceso debe ser controlado con nombres de usuario y contraseñas.
6. La aplicación debe poder integrarse con sistemas externos a nivel de datos y procesos.

Tabla 4.24 ARTEFACTO Lista de objetivos del negocio
Fuente: el autor

ARTEFACTO	Requisitos Funcionales y propiedades de calidad
Constructor	Diseñador Arquitectónico
Actividad que lo genera	Identificar los atributos de calidad

Nro.	Requisitos funcionales	Propiedades de calidad asociadas ISO/IEC 250210
1	Ofrecer una gama de servicios de información de utilidad exclusiva a determinados usuarios en base a su perfil directivos, investigadores, administrativos y otros	**Funcionalidad:** Completitud **Seguridad:** Integridad **Usabilidad:** Atractivo Operable
2	Gestionar el registro y seguimiento de proyectos de investigación al personal de investigación.	**Funcionalidad:** Completitud **Usabilidad** Adecuado
3	Emitir reportes sobre los proyectos de investigación: registrados, en desarrollo y culminados.	**Funcionalidad:** Preciso **Usabilidad** Adecuado Fácil Aprendizaje
4	Gestionar las actividades de investigación realizadas por los investigadores de la Institución.	**Funcionalidad:** Apropiado **Usabilidad:** Atractivo Operable
5	Emitir un conjunto de indicadores de gestión y de estadísticas sobre las actividades de investigación.	**Funcionalidad:** Apropiado **Eficiencia:** Tiempo compartido
6	Proporcionar servicios de información de estadísticas de investigadores y de las actividades de investigación a otros entes internos de la Universidad.	**Usabilidad:** Accesibilidad **Compatibilidad:** Coexistencia **Eficiencia:** Tiempo compartido
7	Suministrar información de los investigadores activos y de los proyectos de investigación a los organismos del Estado como MPPES, OPSU y MCT.	**Confiabilidad:** Disponibilidad **Compatibilidad:** Interoperabilidad **Seguridad:** Confidencialidad

Tabla 4.25 ARTEFACTO Requisitos funcionales y propiedades de calidad
Fuente: el autor

ARTEFACTO	Requisitos No Funcionales y propiedades de calidad
Constructor	Analista de requisitos
Actividad que lo genera	Identificar los atributos de calidad

Nro.	Requisitos no funcionales derivados de las reglas de negocio	Propiedades de calidad asociadas ISO/IEC 25010
1	La aplicación debe ofrecer mecanismos de seguridad, que permita solo a los usuarios registrados acceder a las funcionalidades del sistema, que le corresponde según su rol.	**Seguridad:** Autenticidad Confidencialidad **Funcionalidad:** Apropiado
2	Garantizar el servicio en la red, considerar ancho de banda y protocolos de comunicación.	**Confiabilidad:** Disponibilidad **Eficiencia:** Tiempo compartido Utilización de recursos
4	El sistema deberá almacenar todos los datos en una base de datos SQL estándar, donde pueda ser accesado por otros programas.	**Seguridad:** Integridad
5	Debe contemplar requerimientos de consistencia transaccional. Ante la falla del aplicativo, se debe contar con mecanismos que contemplen la interrupción de transacciones para que estas finalicen de manera correcta. En caso de fallas no debe haber pérdida de la información.	**Confiabilidad:** Madurez Recuperabilidad Tolerancia a fallas
6	El sistema debe tener tolerancia a fallas(el sistema debe mantenerse operativo y recuperarse de posibles fallas)	**Confiabilidad:** Tolerancia a fallas
7	El sistema debe permitir la incorporación de nuevos componentes sin que causen un impacto en la ejecución de la aplicación.	**Mantenibilidad:** Facilidad al cambio
8	La aplicación debe tener una interfaz sencilla y fácil de usar.	**Usabilidad:** Operable Fácil Aprendizaje
9	La aplicación debe soportar la capacidad de comunicarse con sistemas externos a nivel de datos y procesos.	**Compatibilidad:** Interoperabilidad Coextistencia

Tabla 4.26 ARTEFACTO Requisitos no funcionales y propiedades de calidad
Fuente: el autor

ARTEFACTO	Modelo de Calidad
Constructor	Analista de requisitos
Actividad que lo genera	Obtener modelo de calidad

Tabla 4.27 ARTEFACTO Modelo de calidad
Fuente: el autor

ARTEFACTO	Estilos arquitecturales
Constructor	Arquitecto de Software
Actividad que lo genera	Revisar arquitecturas y estilos arquitecturales.

Arquitectura multicapas:

En este estilo arquitectónico, la lógica del negocio se instala y ejecuta separadamente del manejo de los datos y de la interfaz usuario/sistema de la aplicación. Esta arquitectura de software es de tres o más capas y cada capa está compuesta de un conjunto de componentes de software interrelacionados. A continuación se describen las capas y sus componentes: La capa de presentación implementa la interfaz U/S de la aplicación empresarial. Está formada por dos tipos de componentes:

• **Componentes del lado del cliente**.- Son los componentes de la interfaz U/S que se instalan y corren en las máquinas clientes; por ejemplo, los *applets* en una aplicación web.
• **Componentes del lado del servidor web**.- Son los componentes de la interfaz U/S que se instalan en el servidor web. Por ejemplo, los componentes JSP y Servlets que implementan

los aspectos dinámicos de la interfaz web.

La capa de lógica de negocios implementa la funcionalidad de la aplicación. Está formada por dos tipos de componentes:

• **Componentes de procesos**.- Implementan las funciones que requieren los usuarios y automatizan los flujos de trabajo.
• **Componentes de entidades de negocios** (componentes de negocio).- Manejan los datos asociados a los objetos o entidades de negocio de la aplicación.

La capa de datos se encarga de la administración de los datos de la aplicación. Está formada por:
• Una o más bases de datos o almacenes de datos XML que pueden ser locales o distribuidas.
A continuación se muestra la figura con la representación grafica:

Tabla 4.28 ARTEFACTO Estilos arquitecturales
Fuente: el autor

ARTEFACTO	Patrones arquitecturales candidatos
Constructor	Arquitecto de Software
Actividad que lo genera	Escoger patrones arquitecturales

Nro.	Nombre	Descripción	Favorece	Componentes Conectores
1	Master-Slave (Buschmann , 2001)	Divide el trabajo en subtareas idénticas y el resultado	**Confiabilidad:** Tolerancia a Fallas	Master, Slave, mecanismos de comunicación de clases.
2	Proxy (Buschmann , 2001)	Permite acceder y controlar los servicios de un componente.	**Seguridad:** Autenticidad	Proxy, Abstract original, original.
3	View Handler (Buschmann , 2001)	Administra y controla las vistas	**Usalibilidad:** accesibilidad operable	View Handler, SpecificView, AbstractView, Supplier
4	Publisher Subscriber (Buschmann , 2001)	Mantiene un registro actualizado del estado de los componentes suscritos	**Confiabilidad:** Recuperabilidad	Publisher, Proxy Publisher, Proxy Subscriber, Subscriber
5	Transform View (Fowler, 2003)	Organiza las vistas por cada elemento de entrada.	**Funcionalidad:** Apropiado **Usabilidad:** Atractivo Adecuado	Model, Transformer
6	Interceptor (Schmidt, 2000)	Adiciona servicios en forma transparente y pueden ser disparados automáticamente.	**Mantenibilidad:** Fácil de modificar	Concrete Interceptor, Interceptor, Dispatcher, Application, Context
7	Extension Interface (Schmidt, 2000)	Multiples extensiones a un mismo componente, lo que ayuda a los desarrolladores a modificar la funcionalidad del componente	**Mantenibilidad:** Fácil de modificar	Root Interface, Extension Interface, Component, Factory
8	Wrapper Facade (Schmidt, 2000)	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:** adaptabilidad **Mantenibilidad:** Fácil de modificar **Confiabilidad:** Tolerancia a Fallas	Aplicación Componente Funciones API Conectores: Mecanismo de comunicación de las clases

10	Network Communicatio Protocol .	Ancho de banda, medida de dirección durante la ejecución	**Eficiencia** Tiempo compartido	
11	Data mapper, Data Access object, Domain Model (Fowler, 2003)	Permite la manipulación de los datos centrales de las bases de datos, a través de componentes.	**Funcionalidad:** Apropiado Completitud **Eficiencia** Utilización de recursos **Confiabilidad:** Disponibilidad	**Componentes** Repositorio de datos centrales **Conectores** Mecanismos y protocolos de comunicación entre componentes
12	Data Transfer Object (Fowler, 2003)	Contenedores de datos para el transporte entre los subsistemas de aplicación de software.	**Eficiencia** Tiempo compartido **Compatibilidad:** Coexistencia Interoperabilidad	Data Transfer Object, Domain Object, Assembler
13	Server sesión state (Fowler, 2003)	Mantiene el estado de sesión en un sistema de servidor de una forma serializada	**Seguridad:** Autenticidad Accesibilidad Integridad	
14	Gateway (Fowler, 2003)	Permite encapsular los datos para el acceso de otros subsistemas	**Compatibilidad** Interoperabilidad coexistencia	
15	Remote facade (Fowler, 2003)	Abstrae la interacción con un servicio remoto	**Compatibilidad** coexistencia	Remote Facade, Domain object
16	Transaction Script (Fowler, 2003)	Organiza la lógica de negocio para cada transacción en un solo procedimiento, hace llamadas directamente a la base de datos o a través de un envoltorio.	**Confiabilidad:** Disponibilidad Madurez **Seguridad:** Integridad	Transaction Script

Tabla 4.29 ARTEFACTO Patrones arquitecturales candidatos
Fuente: el autor

ARTEFACTO	Elementos arquitecturales
Constructor	Arquitecto de Software
Actividad que lo genera	Instanciar elementos arquitecturales

A continuación se procede a seleccionar los patrones que satisfacen los requisitos funcionales y no funcionales asociados a las propiedades de calidad

Nro.	Requisitos arquitecturales	Nro De Patrón
1	Ofrecer servicios de información de utilidad exclusiva a determinados usuarios en base a su perfil directivos, investigadores, administrativos y otros	2,3,5
2	Proporcionar servicios de información de estadísticas de investigadores y de las actividades de investigación a otros entes internos de la Universidad.	2,10,11,12
3	Suministrar información de los investigadores activos y de los proyectos de investigación a los organismos del Estado como MPPES, OPSU y MCT.	2,10,11,14
4	La aplicación debe ofrecer mecanismos de seguridad, que permita solo a los usuarios registrados acceder a las funcionalidades del sistema, que le corresponde según su rol.	2,13
5	Garantizar el servicio en la red, considerar ancho de banda y protocolos de comunicación.	10
6	El sistema deberá almacenar todos los datos en una base de datos SQL estándar, donde pueda ser accesado por otros programas.	11,16
7	Debe contemplar requerimientos de consistencia transaccional. Ante la falla del aplicativo, se debe contar con mecanismos que contemplen la interrupción de transacciones para que estas finalicen de manera correcta. En caso de fallas no debe haber pérdida de la información.	4,16
8	El sistema debe tener tolerancia a fallas(el sistema debe mantenerse operativo y recuperarse de posibles fallas)	1,8
9	El sistema debe permitir la incorporación de nuevos componentes sin que causen un impacto en la ejecución de la aplicación.	6,7,8
10	La aplicación debe tener una interfaz sencilla y fácil de usar.	3,5
11	La aplicación debe soportar la capacidad de comunicarse con sistemas externos a nivel de datos y procesos.	12,14,15

Tabla 4.30 ARTEFACTO Elementos arquitecturales
Fuente: el autor

SUBPROCESO SELECCIÓN DE LA ARQUITECTURA

En este subproceso se ejecutaron las siguientes actividades:

Subproceso	ACTIVIDADES	ARTEFACTOS
Selección de la Arquitectura	Determinar Subsistemas	**Tabla 4.32**
	Validar soluciones con modelo de calidad	**Tabla 4.33**
	Decidir la solución arquitectural	**Tabla 4.34**

Tabla 4.31: Actividades ejecutadas con sus artefactos. Subproceso Selección de la Arquitectura
Fuente: el autor

De las actividades ejecutadas se obtuvo los posibles subsistemas a desarrollar dentro de la aplicación, como son: Proyectos, Investigadores, Actividades y eventos, y estadísticas. Luego se chequeo el cumplimiento de las características de calidad del modelo de calidad con los patrones seleccionados. Y finalmente se emite el informe con la arquitectura escogida y validada.

A continuación se presenta los artefactos generados:

ARTEFACTO	Descripción de Subsistemas
Constructor	Arquitecto de Software
Actividad que lo genera	Determinar Subsistemas

La aplicación se descompone subsistemas que permite establecer el control y la comunicación entre ellos. A continuacion se muestra el diagrama de subsistemas:

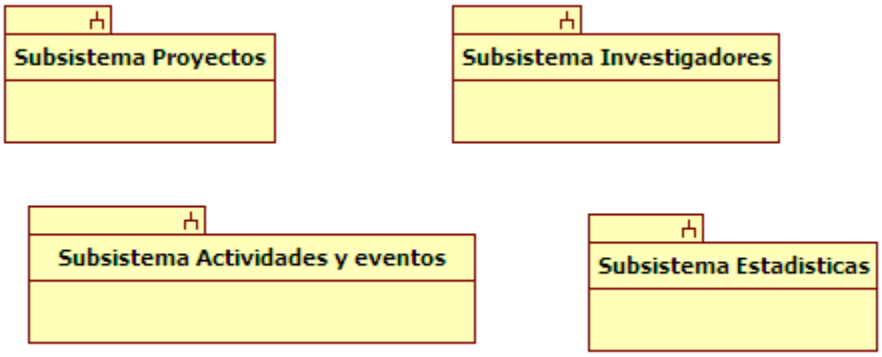


Tabla 4.32 ARTEFACTO Descripción de Subsistemas
Fuente: el autor

ARTEFACTO	Elementos arquitecturales
Constructor	Arquitecto de Software
Actividad que lo genera	Validar solución con modelo de calidad

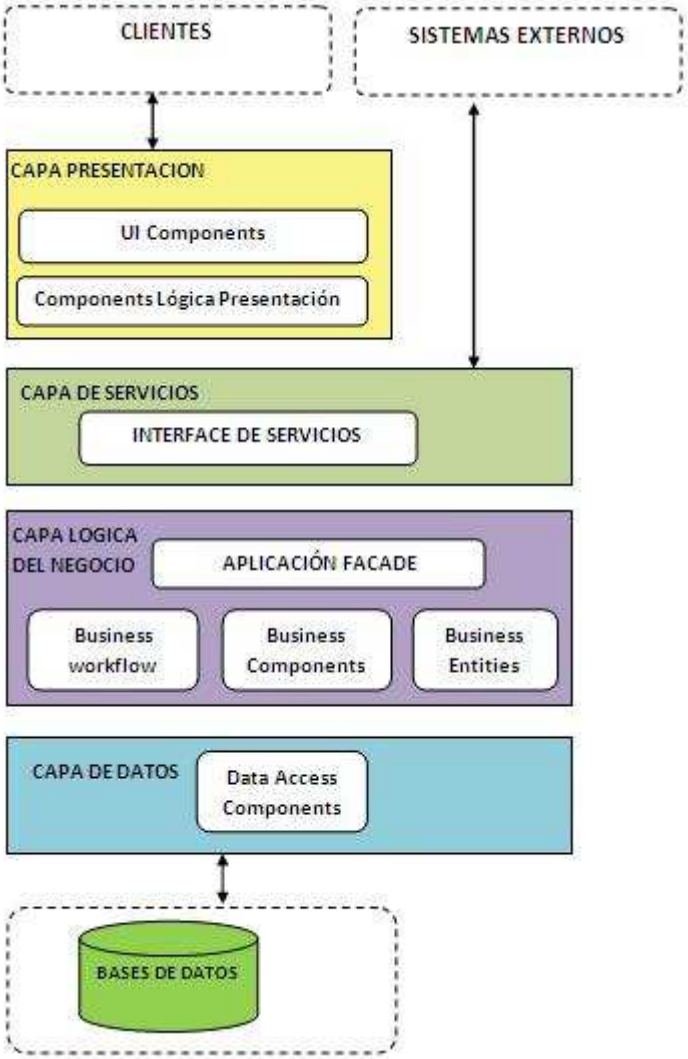
Nro	Requisitos no funcionales derivados de las reglas de negocio	Propiedades de calidad asociadas ISO/IEC 25010	Patrón arquitectural
1	Ofrecer una gama de servicios de información de utilidad exclusiva a determinados usuarios en base a su perfil directivos, investigadores, administrativos y otros	**Funcionalidad:** Completitud **Seguridad:** Integridad **Usabilidad:** Atractivo Operable	Proxy, View Handler, Transform View
2	Proporcionar servicios de información de estadísticas de investigadores y de las actividades de investigación a otros entes internos de la Universidad.	**Usabilidad:** Accesibilidad **Compatibilidad:** Coexistencia **Eficiencia:** Tiempo compartido	Proxy, Network Protocol Comunication, , Data Transfer Object

3	Suministrar información de los investigadores activos y de los proyectos de investigación a los organismos del Estado como MPPES, OPSU y MCT.	**Confiabilidad:** Disponibilidad **Compatibilidad:** Interoperabilidad **Seguridad:** Confidencialidad	Proxy, Protocolos de redes de comunicacion,
4	La aplicación debe ofrecer mecanismos de seguridad, que permita solo a los usuarios registrados acceder a las funcionalidades del sistema, que le corresponde según su rol.	**Seguridad:** Autenticidad Confidencialidad **Funcionalidad:** Apropiado	Proxy, Server session state
5	Garantizar el servicio en la red, considerar ancho de banda y protocolos de comunicación.	**Confiabilidad:** Disponibilidad **Eficiencia:** Tiempo compartido Utilización de recursos	Protocolos de redes de comunicacion
6	El sistema deberá almacenar todos los datos en una base de datos SQL estándar, donde pueda ser accesado por otros programas.	**Seguridad:** Integridad	Data mapper, Data Access object, Transaction script
7	Debe contemplar requerimientos de consistencia transaccional. Ante la falla del aplicativo, se debe contar con mecanismos que contemplen la interrupción de transacciones para que estas finalicen de manera correcta. En caso de fallas no debe haber pérdida de la información.	**Confiabilidad:** Madurez Recuperabilidad	Publisher Subscriber, Transaction script
8	El sistema debe tener tolerancia a fallas(el sistema debe mantenerse operativo y recuperarse de posibles fallas)	**Confiabilidad:** Tolerancia a fallas	Master slave, Wrapper Facade
9	El sistema debe permitir la incorporación de nuevos componentes sin que causen un impacto en la ejecución de la aplicación.	**Mantenibilidad:** Facilidad al cambio	Interceptor, Extension Interface, Wrapper Facade
10	La aplicación debe tener una interfaz sencilla y fácil de usar.	**Usabilidad:** Operable Fácil Aprendizaje	View Handler, Transform View
11	La aplicación debe soportar la capacidad de comunicarse con sistemas externos a nivel de datos y procesos.	**Compatibilidad:** Interoperabilidad Coextistencia	Data Transfer object Gateway, Remote Facade

Tabla 4.33 ARTEFACTO Arquitecturas validadas
Fuente: el autor

ARTEFACTO	Informe de Arquitectura de software
Constructor	Arquitecto de Software
Actividad que lo genera	Decidir la solución arquitectural

El método GRAY WATCH emplea el paradigma de desarrollo de software basado en la reutilización de componentes, así esta aplicación empresarial utiliza la arquitectura de software de tres o más capas, en la que cada una de las capas está compuesta por componentes. El principal beneficio de este enfoque es que la funcionalidad puede ser optimizada independientemente de otras características o funcionalidades, además si una funcionalidad falla, esto no causara que otras funcionalidades fallen también, Este enfoque ayuda hacer la aplicación más fácil de entender y diseñar y facilita el manejo de sistemas complejos. La siguiente figura muestra el modelo de arquitectura seleccionado.



CAPA DE PRESENTACION: Esta capa contiene la funcionalidad orientada al usuario y es responsable de manejar la interacción del usuario con el sistema, y consiste de componentes que proveen un puente común a la lógica del negocio encapsulada en la capa de negocio. A continuación se describen los componentes de esta capa:

UI Components (Componentes de Interface de Usuario): Son elementos visuales (vistas) que muestran la información al usuario y aceptan la entrada del usuario.
Componentes Lógica Presentación: Manejan los aspectos no visuales de la Interface de usuario, este incluye frecuentemente la validación, los eventos y la comunicación entre los UI Components y la interacción de los usuarios.

CAPA DE SERVICIO: La capa de servicio de la aplicación permite interactuar con clientes de otros sistemas, los componentes de la capa de servicio proporciona a otros clientes y aplicaciones la manera de accede a la lógica del negocio en la aplicación y hace uso de la funcionalidad de la aplicación por medio de mensajes, está compuesta por:

Interface de Servicio: funciona como una fachada que representa la lógica del negocio implementada en la aplicación y que permite acceder a los clientes externos.

CAPA DE NEGOCIO: En esta capa se implementan las funcionalidades básicas del sistema y se encapsula la lógica del negocio. Está compuesta por lo siguiente:

Application façade: provee una interface simplificada a los componentes de la lógica del negocio, se combinan múltiples operaciones del negocio en una simple operación que hace más fácil el uso de la lógica del negocio. Ayuda a reducir la dependencia porque los clientes externos no necesitan conocer detalles de los componentes del negocio y las relaciones entre ellos.

Business Workflow components: definen y coordinan procesos del negocio que son complejos y requieren de múltiples pasos, y deben ser orquestados y ejecutados en el orden correcto.

Business components: se encargan de la recuperación, procesamiento, transformación y gestión de datos de la aplicación, la aplicación de reglas de negocio y políticas, y garantizar la consistencia y validez.

Business Entity components. Son los objetos del negocio y encapsulan la lógica del negocio y los datos necesarios para representar elementos del mundo real dentro de la aplicación

CAPA DE DATOS: Los componentes de la capa de datos facilitan el acceso a los datos que se aloja dentro de los límites del sistema, y los datos expuestos por otros sistemas

en red. Está compuesta por:

Data Access component: son usados para la abstracción de la lógica requerida para acceder a los almacenes de datos.

En lo que se refiere a seguridad, se implementara en cada capa mecanismos de seguridad, a través de componentes que se encarguen de identificar los usuarios, y permitirles utilizar y manipular la información adecuada.

Tabla 4.34 ARTEFACTO Informe de Arquitectura de software
Fuente: el autor

CONCLUSIONES Y RECOMENDACIONES

El desarrollo de software es un proceso complejo, requiere la aplicación de principios, métodos, modelos y técnicas de Ingeniería de Software y Gerencia de Proyectos. Esta investigación considero el método GRAY WATCH diseñado por Montilva et al (2008) como un método de desarrollo de software, cuyo marco metodológico describe los procesos técnicos, gerenciales y de soporte que deben emplear los equipos de trabajo, para desarrollar los componentes arquitectónicos de una aplicación e integrarla al sistema de negocios para el cual ella es desarrollada.

La propuesta presentada en este estudio incorporo al método GRAY WATCH la aplicación del estándar ISO/IEC 25010, lo que permitió profundizar acerca del modelo de productos más adecuado que describe los artefactos generados en el proceso de calidad al aplicar esta norma.

De igual forma, se pudo conocer los pasos a seguir para obtener los requisitos de calidad, modelos de calidad y el diseño arquitectónico. Para ello, se modificaron y extendieron subprocesos del proceso de Análisis y Diseño de la aplicación del método GRAY WATCH, los procesos y los productos generados se adaptaron al resto de los procesos de la Cadena del método GRAY WATCH.

Particularmente se aplicó la extensión del método GRAY WATCH al caso de estudio Programa de Investigación de la UPEL-IPB, que requería una aplicación Web que gestionara los procesos del negocio. El uso del método GRAY WATCH extendido como metodología de desarrollo fue de gran utilidad para el equipo de desarrollo ya que se ajusto a las necesidades de la aplicación empresarial que se pretende construir, asimismo fue posible aplicar los procesos técnicos que fueron modificados puesto describían paso a paso las actividades a realizar para construir los productos.

La utilización del método GRAY WATCH extendido también permitió obtener un modelo de calidad y una arquitectura que satisface los requisitos arquitectónicos

basados en el estándar ISO/IEC 25010, esto optimiza la gestión de la calidad del producto, asegurando que la aplicación desarrollada cumpla con los estándares de calidad y con las especificaciones solicitadas por el cliente, en este caso el Programa de Investigación de la UPEL-IPB.

Por otro lado, en cuanto a las recomendaciones, se sugiere revisar otros procesos técnicos del método GRAY WATCH, ya que esta investigación estuvo limitada a las primeras fases de desarrollo, así se puede examinar los procesos de implementación tales como Programación, Integración, Pruebas y puesta en marcha de la aplicación para determinar cómo es la calidad del producto que se utiliza.

Sin embargo, al observar el abanico de posibilidades que tiene el estándar ISO/IEC 25000 SQUARE, se encuentra que el estándar de evaluación de calidad ISO/IEC 25040 que proporciona una guía para la evaluación de la calidad, pudiera ser incorporado para expandir los procesos de Prueba de la Aplicación del método GRAY WATCH.

Por otro lado, se tiene el estándar de mediciones de calidad ISO/IEC 25020, que presenta aplicaciones de métricas para la calidad de software interna, externa y en uso, que pudiera ser de gran utilidad para incorporar métricas de calidad que permitan evaluar la calidad en uso del Sistema desarrollado.

Finalmente, se invita a la comunidad de Ingeniería de Software a considerar otros aspectos o elementos que pudieran evolucionar la extensión del método GRAY WATCH y crear nuevas experiencias en cuanto al desarrollo de software con normas de calidad.

También, se espera compartir la experiencia de esta investigación en congresos, conferencias y/o revistas a fin de encontrar la retroalimentación con actores de otros trabajos de investigación.

REFERENCIAS BIBLIOGRAFICAS

Bernal, C. (2006). *Metodología de la Investigación*. Pearson Educación. Mexico.

Brinkkemper, S. (1996). *Method engineering: engineering of information systems development methods and tools*. University of Twente . Information and Software Technology. [Article en lineal] Disponible: http://doc.utwente.nl/18012/1/Brinkkemper96method.pdf

Buschmann,F., Meunier, R., Rohnert, H., Sommerlad, P. y Stal, M. (2001). *Pattern Oriented Software Architecture*. Willey&Sons. England.

Campderrich, B. (2003).*Ingeniería del Software*. Editorial UOC. Barcelona

Canelón, R. (2007). *ISO - 25010, Software engineering-Software product Quality Requirements and Evaluation (SQuaRE) Quality model.* Universidad Centroccidental Lisandro Alvarado (UCLA). Barquisimeto.

Canelón, R., Losavio, F., Matteo, A. y Chirinos, L. (2009). *Modelo Conceptual para Modelación de Aplicaciones Móviles sensibles al contexto*. Rev. Fac. Ing. UCV. Vol 24(2), p. 93-103.

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

Calero, C., Moraga M. y Piattini, M. (2010). Calidad del Producto y Proceso de software. Editorial Ra-Ma. España.

Castillo, I., Losavio, F., Matteo, A. y Boegh, J. (2010). *Requirements, Aspects and Software Quality: The REASQ Model*. [Articulo en línea] Disponible: http://www.jot.fm/issues/issue_2010_07/article4.pdf [consulta: abril, 2011]

Chirinos, L., Losavio, F. y Matteo, A. (2004). *Identifying Quality Based Requirements.* [Articulo en línea] Disponible: http://www.auerbach-publications.com/dynamic_data/2744_1617_quality.pdf [consulta: abril 2011]

Cortes, E., Prieto, J., Sampalo, M. y Garzón, M. (2006). *Sistemas y Aplicaciones Informáticas.* Editorial MAD S.L. Sevilla

Crosby, P. (1979). *Quality is Free*. MsGraw-Hill. New York.

FitzPatrick, R. (1996). *Software quality: Definitions and Strategic Issues*. Staffordshire University. [Articulo en línea] Disponible en: http://www.comp.dit.ie/rfitzpatrick/papers/quality01.pdf [consulta: mayo, 2011]

Fowler, M. (2003). *Enterprise Application Architecture Patterns*. AddisonWesley. USA.

García, R. (2001). *El Modelo de Capacidad de Madurez y su Aplicación en Empresas mexicanas de software*. Tesis de Grado. Universidad de las Américas: Puebla.

Hamar, V. (2003). *Aspectos metodológicos del desarrollo y reutilización de componentes de software.* [Tesis en línea] Disponible: http://tesis.ula.ve/harvester/index.php/record/view/28025

Harmsen, A. (1997). *Situational Method Engineering*. University of Twente. Netherlands.

ISO/IEC 9126-1 (2001) Software Engineering— Software Product Quality—Part 1: Quality Model.

ISO/IEC 25010 (2011). *Software Engineering – Software Product Quality Requirements and Evaluation* (SQuaRE) – Quality model and guide.

ISO/IEC 25000. (2005). *Calidad del Producto de Software y la norma* ISO/IEC 25000. [Sitio Web] Disponible: http://iso25000.com/ [consulta: abril,2011]

Juran, J. (1982). Upper Management and *Quality*. Juran Institute. New York

Kruchten, P. y Kroll, P.(2004). The Rational Unified Process made easy. Addison-Wesley Professional. Boston.

López, A. y Cabrera, C. (2008). *Introducción a la Calidad de Software.* [Articulo en línea]. Disponible: http://www.utp.edu.co/php/revistas/ScientiaEtTechnica/docsFTP/111233326-331.pdf [consulta: abril, 2011]

Losavio, F., Chirinos, L., Levy, N., y Ramdane-Cherif, A. (2003). Quality Characteristics for Software Architecture. . [Articulo en línea]. Disponible: http://www.jot.fm/issues/issue_2003_03/article2/ [consulta: mayo, 2011]

Losavio, F., Matteo, A. y Pacilli, I. (2009). Proceso dirigido por objetivos para análisis de dominio bajo estándares de calidad. *Enl@ce Revista Venezolana de Información, Tecnología y Conocimiento*, 6 (3), 11-28.

Montilva, J. Hazam, K., and Gharawi, M. (2000). "*The Watch Model for Developing Business Software in Small and Midsize Organizations*". IV World Multiconference on Systemics, Cybernetics and Informatics - SCI´2000. Orlando, Florida, Julio, 2000. Vol. XII, pp. 263-268.

Montilva, J. (2004). *Desarrollo de aplicaciones Empresariales. Método WATCH*. Universidad de los Andes. Mérida.

Montilva, J., Barrios, J. y Hamar, V. (2004). *Aspectos metodológicos del desarrollo de componentes de software reutilizables*. SEPG Conferencia Latinomaericana. México.

Montilva, J. (2007). *WATCH: El método del Reloj. Un método para desarrollar aplicaciones empresariales*. VII Congreso de Expotecnología UVM. [presentación en línea] Disponible: http://www.scribd.com/doc/23939280/Metodo-Watch [consulta: mayo, 2011]

Montilva, J., Barrios, J. y Rivero M. (2008). GRAY WATCH Método de desarrollo de software para aplicaciones empresariales. Proyecto METHODIUS. Mérida.

O'Regan, G. (2002). *A practical approach to software quality*. Springer-Velarg. New York.

Odell, J. (1996). *A primer to method Ingenieering*. INFOSYS. USA.

Palomo, I. , Veloso, C. y Schmal, R. (2007). *Sistema de Gestión de Investigación de la Universidad de Talca*. Información Tecnológica Vol 18(1):97-106

Pressman, R. (2002). *Ingeniería del Software*. Editorial McGraw Hill. Madrid

Rivero, L. (2011). *Una línea de producción de software para sistemas transaccionales .Una aplicación al proceso de desarrollo de software en la Coordinación Nacional de Tecnología de Información de la U.N.E.X.P.O***.** Trabajo de Grado. UCLA

Rodríguez, M. (2010). *Calidad del Producto de Software ISO/IEC 25000. IX Cursos de Verano Santander* [Presentación en línea]. Disponible en: http://alarcos.inf-cr.uclm.es/per/fruiz/cur/santander/mrodriguez-iso25000-update.pdf [consulta: abril, 2011]

Rodríguez, M., Verdugo, J., Coloma, R., Genero, M. y Piattini, M. (2010). *Metodología para la evaluación de la calidad en los modelos UML*. REICIS. Vol 6(1): 16-35.

Rolland, C. (1997). *A primer for method engineering*. Conference INFORSID. Universite de Paris. Toulouse.

Rolland, C., Ben, C., Cauvet, C., Ralyté, J., Sutcliffe, A. , Maiden, M., Jarke, M., Haumer, P., Pohl, K., Dubois, E. y Heymans, P. (1998).*A Proposal for a Scenario Classification Framework, Requirements Engineering Journal* (REJ), Vol. 3, N°1, pp. 23-47.

Scalone, F. (2006). *Estudio Comparativo de los modelos y estándares de calidad de software.* Trabajo de Grado. Universidad Tecnológica Nacional. [Tesis en línea.] Disponible: http://laboratorios.fi.uba.ar/lsi/scalone-tesis-maestria-ingenieria-en-calidad.pdf [consulta abril, 2011]

Schmidt, D., Buschmann,F., Rohnert, H., y Stal, M. (2000). *Patterns for Concurrent and Networked Objects*. Willey&Sons. England

Sommerville, I. (2005). *Ingeniería de Software*. Pearson Educación. Madrid

Tamayo, M. (1999). *La Investigación*. Serie Aprender a Investigar.ARFO EDITORES LTDA. Bogotá.

Universidad Centroccidental "Lisandro Alvarado" (2002). *Manual para la Elaboración el Trabajo Conducente a Grado Académico de Especialización, Maestría y Doctorado*. Barquisimeto: Autor

Universidad Pedagógica Experimental Libertador, Vicerrectorado de Investigación y Posgrado. (1998). *Políticas de Investigación*. Caracas: Autor.

Universidad Pedagógica Experimental Libertador, Vicerrectorado de Investigación y Posgrado. (2006). *Manual de Trabajos de Grado de Especialización y Maestría y Tesis Doctorales*. FEDEUPEL. Caracas

Valdivia, D. y Valdivia, E. (2005). *Estándares de Calidad para Pruebas de Software.* Trabajo de Grado. Universidad Nacional Mayor de San Marcos. [Tesis en línea] Disponible: http://www.cybertesis.edu.pe/sisbib/2005/valdivia_ee/pdf/valdivia_ee-TH.4.pdf [consulta: abril, 2011]

Veenendaaal, E., Hendriks, R. y Vonderen, R. (2007) *Measuring software product quality. Fundamental Concepts for the software quality engineer*. Volumen 2. American Society for Quality. Milwaukee

ANEXO A. CURRÍCULO VITAE DEL AUTOR

Datos Personales

Nombres y Apellidos: Iliana Cristina Sánchez

Lugar y Fecha de Nacimiento: Barquisimeto, 21 de Marzo 1979
Cédula de Identidad: V 13843868
Dirección Habitación: Urb. La Teura calle F casa F-08, Barquisimeto Estado Lara.
Teléfono: Móvil: 0426-5518026

Datos Académicos

EDUCACIÓN SUPERIOR Universidad Centroccidental "Lisandro Alvarado"
Año de Graduación: 2002
Titulo Recibido: Ingeniero en Informática.

Cargos Desempeñados

- **Institución:** Instituto Pedagógico de Barquisimeto UPEL-IPB
  Fecha: de 01-02-2002 hasta el 2005
  Nombre del Cargo: Analista Programador de Sistemas

- **Institución:** Banco Central de Venezuela
  Fecha: de 01-12-2005 hasta el 2006
  Nombre del Cargo: Consultor de Sistemas

- **Institución:** Instituto Pedagógico de Barquisimeto UPEL-IPB
  Fecha: de 01-09-2006 Actual
  Nombre del Cargo: Analista Programador de Sistemas

Estudios de Perfeccionamiento

- **Nombre del Curso:** "Diplomado en Entornos Virtuales de Aprendizaje"
  Institución: FATLA
  Fecha: 01-02-2009
  Duración: 400 horas
  Actuación: PARTICIPANTE
- **Nombre del Curso:** "Diplomado en Tecnologías de Información y Comunicación "
  Institución: UCLA
  Fecha: Octubre 2002
  Duración: 400 horas
  Actuación: PARTICIPANTE