Centro de Investigación en Matemáticas, A. C.

Proceso para el desarrollo de Arquitecturas de Software basado en DFSS

Reporte Técnico de Investigación que para obtener el grado de

Maestro en Ingeniería de Software

presenta

Araceli Núñez Mora

Director:

Dr. Cuauhtémoc Lemus Olalde

Guanajuato, Gto., México, a 30 de julio del 2006.

A mis queridos padres, Lolita y Toñito,
a mis hermanos, Pepe Toño, Nano, Raquelucha, Mó,
Dundun, Hemi y
a mis sobrinos, Leo y Emiliano
por su incondicional apoyo y constante cariño.
Gracias

Agradecimientos

El presente trabajo final representa un concentrado de conocimientos, aportaciones y lecciones aprendidas que recibí de las personas a las cuales agradezco y reconozco en estas líneas:

Al Concejo de Ciencia y Tecnología del Estado de Guanajuato (CONCYTEG), por su parcial financiamiento a través del Proyecto Número GTO-2002-C01-5333 titulado “Promoviendo calidad en la industria de software: Recursos humanos, investigación, servicios” como programa de beca, lo cual me permitió estudiar una Maestría en Ingeniería de Software única en Latinoamérica.

Al Dr Cuauhtémoc Lemus Olalde, mi director de investigación, por sus enseñanzas y sus recomendaciones dadas para la elaboración de este documento.

A mis compañeros de generación de la maestría, por que de cada persona hay mucho que aprender, también quiero agradecer a dos compañeros de la generación pasada que estuvieron apoyándonos en todo momento, Juan Antonio y Karen y finalmente a un gran amigo que me ha apoyado y me ha aconsejado durante en transcurso de la maestría, Rodi.

Atentamente, ANM.

Tabla de Contenido

Lista de figuras

Lista de tablas

1 Introducción

La base de cualquier sistema de software es su arquitectura y por años los diseñadores de software han estado desarrollando sistemas sin tomar en cuenta este aspecto y por lo tanto los sistemas resultantes no son lo que los clientes desean ó en el mejor de los casos no cumple con todos los objetivos establecidos, por lo tanto se realizan sistemas sin calidad y se requiere de más tiempo y esfuerzo para repararlos. No es si no hasta 1990 cuando el termino de “Arquitectura de Software” comenzó a tomar aceptación tanto en la comunidad de investigadores como en la industria. Las bases sobre las cuales éste campo de la Ingeniería de Software subyace fueron creadas por sus fundadores, David Parnas, Fred Brooks, Edsger Dijkstra, entre otros. [2]

La Arquitectura de Software ha emergido como una parte de gran importancia en el proceso de diseño y es el objeto de investigación de este trabajo. El cual tiene como objetivo, definir un proceso de Arquitecturas de Software basado en Diseño para Seis Sigma (DFSS, por sus siglas en inglés, Design for Six Sigma), el cual servirá de apoyo académico en la impartición de la materia de Arquitecturas de Software de la Maestría en Ingeniería de Software (MIS) que es impartida en el CIMAT, A.C, y en la cual no se enseña un proceso de Arquitecturas de Software como tal.

DFSS es una metodología para el diseño de un nuevo producto o servicio y la cual toma lo mejor de seis sigma para reducir el tiempo de entrega y costo de desarrollo e incrementar la efectividad del producto o servicio y así lograr la satisfacción del cliente [4]. DFSS tiene sus orígenes en el área de manufactura pero actualmente empresas como: Hewlet Packard, AT&T, NASA, General Electric, entre otras han utilizado DFSS dentro del diseño de Software. Esencialmente el proceso de DFSS se resume en los siguientes elementos: 1) Dirige el proceso de diseño para que este orientado al cliente, 2) Predice la calidad del diseño desde el inicio, 3) Dirige medidas de calidad y mejoras de pronosticabilidad en fases tempranas del diseño, 4) usa la capacidad de los procesos en la toma de decisiones, 5) monitorea la varianza de los procesos para verificar que los requerimientos del cliente se conocen.

El proceso del desarrollo de la Arquitectura de Software es un paso muy importante en el desarrollo de sistemas de software complejos y grandes, donde un gran número de personas deben colaborar para el desarrollo de la misma (clientes, usuarios finales, desarrolladores, administradores de proyectos, los que le darán mantenimiento, etc.), donde se define a alto nivel los componentes y su interrelación, así como su responsabilidad dentro del sistema. La Arquitectura de Software deberá plasmar el(los) estilos arquitectónicos en que estará basada, así como fundamentar el uso del mismo para satisfacer los requerimientos funcionales y los atributos de calidad, dejando plasmado en un documento todas las decisiones que se hayan tomado (tradeoffs, reutilización de componentes, evaluación de la arquitectura, etc.) para que este a su vez sirva como medio de comunicación entre los stakeholders. Pero aún con la importancia que representa la definición del proceso de la Arquitectura de Software, no existe en la literatura un proceso que permita identificar de manera clara, las actividades, los elementos de entrada y salida de cada una de ellas así como las herramientas a utilizar, además que para las empresas es difícil publicar un proceso que podría darles ventaja competitiva ante la competencia.

Cabe señalar la existencia de trabajos que investigadores como Bass, Kazman, Bosh, entre otros, han desarrollado en contribución a esta problemática y cuyos trabajos han servido de base para que los arquitectos de software y la academia tengan una base para el desarrollo de la Arquitectura de Software. Entre los trabajos más representativos tenemos: Bass y Kazman los cuales propusieron un proceso de desarrollo basado en Arquitecturas [12], Bosh y Molin propusieron un diseño de arquitecturas de software: evaluación y transformación [6], Kazman, Carriére y Woods con su arquitectura basada en escenarios [21], entre otros.

Por lo tanto y considerando la importancia de la arquitectura de software, así como la problemática mencionada anteriormente, el presente trabajo de investigación propone el diseño de un proceso de desarrollo de Arquitecturas de Software basado en DFSS, el cual pretende utilizar como base la experiencia adquirida durante el curso de Arquitecturas de Software, así como contestar a la incógnita: Como alumno de la MIS, ¿Cuál es el proceso de arquitecturas de software que me hubiera gustado seguir para el desarrollo de la misma?. La propuesta estará centrada en mapear los pasos que DFSS define en su metodología y adecuarlos al desarrollo de Arquitecturas de Software dejando fuera de éste trabajo la adecuación de las herramientas que DFSS define y que posiblemente puedan utilizarse en Software.

El presente trabajo se encuentra estructurado de la siguiente manera:

Parte 1: **Introducción**.- Describe a manera de resumen cual es la problemática actual en el proceso de desarrollo de Arquitecturas de Software y como el uso de metodologías como DFSS pueden servir de apoyo a la Ingeniería de Software.

Parte 2: Trabajo previo.- Describe el trabajo previo realizado por algunos de los investigadores de Arquitecturas de Software, así como el trabajo realizado en la materia.

Parte 3: Descripción del problema.- Describe cual es la problemática a la que se enfrenta la Ingeniera de Software en el área de Arquitecturas de Software.

Parte 4: Propuesta de solución.- Describe la propuesta de solución a la problemática presentada y como dicha propuesta puede servir de apoyo académico en la Maestría de Ingeniería de Software que imparte el CIMAT A.C., con la finalidad de que los alumnos tengan un proceso base para el desarrollo de Arquitecturas de Software.

Parte 5: Conclusiones y trabajos futuros.- Describe las conclusiones finales y los trabajos que de esta investigación se derive.

Parte 6: Glosario.- Describe los conceptos y acrónimos utilizados en el presente trabajo de investigación, con la finalidad de tener un mejor entendimiento del tema.

Parte 7: Referencias.- Describe las referencias bibliográficas que en las que esta basado el presente trabajo de investigación; artículos, libros y páginas Web.

2 Trabajo previo

En esta sección se presentan los trabajos previos relacionados con la metodología seis sigma, DFSS y procesos de arquitecturas de diseño de arquitecturas.

2.1 Seis Sigma y DFSS

El concepto de implementación de la metodología de seis sigma fue iniciada por Motorola en los 1980´s con el objetivo de reducir los costos de calidad. La metodología seis sigma ha evolucionado de métodos orientados estadísticamente a mejoras de la calidad de los proceso, productos o servicios. Esto es una estrategia de mejora de los negocios usada para mejorar la rentabilidad, para evitar gastos en el proceso de negocios y mejorar la eficiencia de todas las operaciones que conocen o sobrepasan las necesidades y expectativas del cliente. Un nivel de desempeño seis sigma es igual a 3.4 defectos por millón de oportunidades, donde sigma es una medida estadística del nivel de variación de un proceso. El nivel de variación promedio para muchas compañías es de tres sigmas, lo que equivale a 66 800 defectos por millón de oportunidades. [11]

De acuerdo con [11] las organizaciones que han adoptado los principios y conceptos de la metodología seis sigma han notado que una vez que llegan a un nivel cinco sigma les resulta costoso el ciclo de vida del producto. Si un error de diseño es detectado durante la etapa de manufactura costará cien veces más repararla, que si el mismo error se repara en la etapa de diseño. Para lo cual DFSS fue creado y entre sus más importantes beneficios están:

- Reduce el time to market de los productos
- Reduce los costos del ciclo de vida asociados a los productos
- Esta enfocado en el entendimiento de las expectativas y prioridades de los clientes
- Reduce el número de cambios de diseño

2.2 Diseño para Seis Sigma

Al igual que en la Ingeniería de Software las compañías manufactureras, tienen un diseño orientado al cliente, donde existe una transformación entre las necesidades del cliente y el diseño de la solución. Este proceso es trasladado sobre varias fases, comenzando desde la fase conceptual donde concibes, evalúas y seleccionas un buen diseño que de solución a las necesidades del cliente, lo cual en muchas ocasiones no es una tarea fácil y tiene consecuencias enormes. Las compañías de manufactura y diseño usualmente operan de dos maneras:

- Fire prevention.- Concibe entidades conceptuales viables y robustas, este método es preventivo.
- Firefighting.- Resuelve problemas de entidades de diseño. Desafortunadamente éste último consume recursos humanos, no humanos y tiempo, ya que este método no es preventivo si no correctivo.

Las más recientes tendencias en la investigación de diseño se presentan en dos sentidos: a) mejorar el desempeño de los diseños de acuerdo al ambiente en el que se utilizan. Éste método de diseño robusto fue sugerido por G.Taguchi(1994), en el cuál el objetivo es tener una solución robusta para satisfacer los objetivos funcionales y puedan cumplirse a través del proceso de diseño [8]. b) el segundo tiene que ver con los métodos conceptuales, el cual es sugerido por Suh(1990). El interés de Suh tiene que ver con las vulnerabilidades de diseño que son introducidas en la solución de diseño cuando ciertos principios son violados. Suh propone el uso de axiomas [8],

ambos están creados para mejorar el proceso de diseño en orden de mejorar las soluciones de diseño.

2.2.1 Qué es Diseño para Seis Sigma

El principal objetivo de DFSS es “diseñar bien y a la primera” para evitar malas y costosas experiencias. El termino seis sigma en el contexto de DFSS puede ser definido como el nivel en el cual las vulnerabilidades de diseño son mínimas. Generalmente, dos aspectos en la vulnerabilidad del diseño que pueden afectar la calidad del mismo son:

- Vulnerabilidades conceptuales que son establecidas cuando son violados los principios y axiomas de diseño.
- Vulnerabilidades operacionales debido a la falta de robustez en el uso del ambiente. La eliminación o reducción de éste tipo de vulnerabilidades es el objetivo de las iniciativas de calidad incluidas en Seis Sigma.

La implementación de DFSS en la fase conceptual es la meta principal y puede lograrse siguiendo una metodología de diseño, combinada con conceptos de calidad y métodos “upfront” [8]; para atacar tanto las vulnerabilidades conceptuales como las operacionales.

DFSS integra herramientas que permiten eliminar o reducir dichas vulnerabilidades, pero en general el principal problema es que los métodos de diseño actuales son empíricos. Ellos representan lo mejor de la comunidad de diseño, que desafortunadamente, carece de base científica. Por lo tanto, cuando las compañías sufren de la no satisfacción del cliente, la experiencia y el juicio pueden ser no suficientes para obtener una solución optima, y DFSS precisamente ataca éste problema desde las etapas tempranas del diseño, motivando el hecho de que las decisiones de diseño hechas en etapas tempranas tienen un gran impacto en el costo total y calidad del sistema. El área de investigación de manufactura, incluyendo el desarrollo de productos esta actualmente haciendo esfuerzos para reducir los costos de desarrollo y manufactura, reducir el costo total del ciclo de vida (LCC, por sus siglas en inglés, Life cicle cost) y mejorar la calidad de las entidades de diseño en forma de productos, servicios o procesos. En la experiencia de [8], al menos el 80% de la calidad del diseño esta comprometida en las fases tempranas.

2.2.2 Fases del Diseño para Seis Sigma

DFSS tiene las siguientes cuatro fases:

- Identificación de los requerimientos
- Caracterización del diseño
- Optimización del diseño
- Verificación del diseño

Fase 1. Identificación de los requerimientos.- Los proyectos de DFSS pueden ser categorizados como diseño de entidades o rediseño de entidades. “Diseño creativo” es el termino que usaremos para indicar nuevos diseños y diseños incrementales para rediseños.

Paso 1: Establecer el estatus del proyecto (project charter).- Se establece la duración del proyecto, que generalmente son largos y con costos iniciales altos. La duración de proyectos largos es por que la compañía esta rediseñando ó diseñando diferentes entidades, no parchando un diseño ya existente. Los costos iniciales altos son debido a que existen muchos requerimientos de los clientes para que sean identificados y estudiados, dado que se necesitan identificar todas las métricas críticas que satisfacen dichos requerimientos (CTS) para concebir y optimizar mejores diseños. También se establece el equipo de trabajo con representantes tanto internos como externos (clientes y proveedores), estableciendo los roles, responsabilidades y recursos necesarios.

Paso 2: Identificar requerimientos de negocio y del cliente.- En este paso, los clientes son totalmente identificados y sus necesidades recolectadas y analizadas, con la ayuda de la herramienta QFD, y análisis de Kano [11]. Entonces el más apropiados conjunto de métricas CTSs son determinadas y ordenadas para medir y evaluar el diseño (QFT y análisis de Kano también ayudan para establecer los limites y objetivos numéricos para los CTSs).

En resumen, la siguiente lista de pasos enumera los pasos que se siguen en esta fase:

- Identificar los métodos para obtener las necesidades de los clientes.
- Obtener las necesidades de los clientes y transformarlas a una lista de VOC (voice of customer).
- Traducir la lista de VOC dentro de requerimientos funcionales y medibles.
- Finalizar los requerimientos:
    - Estableciendo la definición de los requerimientos mínimos.
    - Identificar y rellenar los huecos en los requerimientos proporcionados por el cliente.
    - Validar la aplicación y ambiente de uso.
- Identificar los CTSs como: Críticos para la calidad (CTQ), críticos para la entrega (CTD), críticos para el costo (CTC), y así sucesivamente.
- Cuantificar los CTSs
    - Establecer métricas para los CTSs
    - Establecer niveles de desempeño y funcionamiento de ventanas aceptables

Herramientas usadas en esta fase.- Las herramientas usadas en esta fase incluyen:

- Investigación de mercado y del cliente
- QFD
- Análisis de kano
- Análisis de riesgos

Fase 2. Caracterización del diseño.

Paso 1: Traducir los requerimientos del cliente (CTSs) a requerimientos funcionales del producto y del proceso.- Los requerimientos del cliente, CTSs dan ideas de lo que satisfacera al cliente, pero éstos no peden ser usados directamente como los requerimientos para el diseño del producto o proceso. Se necesita traducir los requerimientos del cliente a requerimientos funcionales. QFD puede ser usado para agregar esta transformación.

Paso 2: Generar alternativas de diseño.- Después de la resolución de los requerimientos funcionales para la nueva entidad de diseño (producto, servicio o proceso), se necesita caracterizar (diseñar) las entidades de diseño que estarán disponibles para satisfacer los requerimientos funcionales. En general, existen dos posibilidades:

- La existencia de tecnología o conceptos de diseño conocidos disponibles para satisfacer todos los requerimientos satisfactoriamente. Por lo tanto éste paso llega a ser casi un ejercicio trivial.
- La tecnología existente o conceptos de diseño no están disponibles para satisfacer todos los requerimientos satisfactoriamente, entonces nuevos conceptos de diseño deberán ser desarrollados. Este nuevo diseño puede ser creativo o incremental, por lo tanto el método TRIZ podría ayudar a generar innovadores conceptos de diseño en ese paso.

Paso 3: Evaluar las alternativas de diseño.- Varias alternativas de diseño pueden ser generadas en el paso anterior, por lo tanto se necesita evaluarlas y tomar la decisión de cual concepto será usado. Existen varios métodos que pueden ser utilizados en la evaluación del diseño incluyendo técnicas de selección de conceptos Pugh, revisiones de diseño, análisis de vulnerabilidades del diseño y FMEA. Después de la evaluación un concepto será seleccionado. Durante la evaluación, muchas debilidades del conjunto inicial del concepto de diseño serán

expuestas y los conceptos serán revisados y mejorados. Si estamos diseñando un proceso, técnicas de administración de procesos serán usadas como herramientas de evaluación.

Herramientas usadas en esta fase.- Las herramientas usadas en esta fase incluyen:

- TRIZ
- QFD
- Diseño robusto
- DFMEA y PFMEA
- Revisiones de diseño (design review)
- CAD/CAE
- Simulación
- Administración de procesos

Fase 3. Optimización del diseño. El resultado de esta fase es la optimización de las entidades de diseño con todos los requerimientos liberados en un nivel de desempeño seis sigma. Como el concepto de diseño esta finalizado, existen parámetros que pueden ser ajustados y cambiados. Herramientas como simuladores y/o probadores de hardware pueden ser útiles en esta fase. Usualmente en DFSS esta fase de optimización de parámetros esta seguida de un paso de optimización de tolerancia. El objetivo es facilitar una base lógica y objetiva para configurar las tolerancias de manufactura.

Herramientas usadas en esta fase.- Las herramientas usadas en esta fase incluyen:

- Herramientas de diseño y simulación
- Diseño de experimentos
- Método Taguchi, diseño de parámetros y diseño de tolerancia
- Diseño basado en confiabilidad
- Evaluación de robustes

Fase 4. Validación del diseño.- Después de que el diseño de tolerancia y parámetros es completado, las actividades de validación y verificación son ejecutadas.

Paso 1: Prueba piloto y refinamiento.- Ningún producto o servicio deberá ir directamente al mercado sin primero ser piloteado y refinado. En este paso se puede usar DFMEA como la implementación de la prueba piloto en escala pequeña, para probar y evaluar el rendimiento en la vida real.

Paso 2: Validación y control del proceso.- En este paso se valida la nueva entidad para asegurar que el resultado final fue diseñado conociendo los requerimientos de diseño y que el control del proceso en manufactura y producción están establecidos, con el objetivo de asegurar las características criticas.

Paso 3: Presentación comercial y entrega del nuevo proceso al cliente.- Debido a que la entidad de diseño esta validada y el control del proceso esta establecido, se realiza el lanzamiento comercial a gran escala de la nueva entidad.

Herramientas usadas en esta fase.- Las herramientas usadas en esta fase incluyen:

- Modelado de procesos de capacidad
- DOE
- Pruebas de confiabilidad
- Análisis de confidencialidad
- Plan de control de procesos

2.3 Procesos de Arquitecturas de Software

En esta sección se describirán brevemente algunos de los proceso de arquitecturas de software que fueron analizados y comparados con el proceso de DFSS para poder saber si el proceso definido en DFSS era útil para Arquitecturas de Software.

2.3.1 Método de diseño basado en Arquitecturas

En esta sección se describe brevemente el método ABD "Architecture Based Design" [19], el cual provee una estructura para producir la arquitectura conceptual de un sistema. La arquitectura conceptual es una de las cuatro diferentes arquitecturas identificadas por Hofmeister, Nord y Soni. Dicha arquitectura describe el sistema en términos de los elementos de diseño de más alto nivel y las relaciones entre ellos.

El método ABD depende de determinar los drivers arquitectónicos de un sistema, donde dichos drivers es una combinación entre los requerimientos de negocio, funcionales y de calidad.

La Figura 2-1 muestra el método ABD, donde en el nivel más alto el sistema es descompuesto dentro de subsistemas conceptuales y uno o más plantillas de software. En el siguiente nivel, los subsistemas conceptuales son descompuestos dentro de componentes conceptuales y una o mas plantillas de software.

Este método es recursivo, los pasos que se aplican al sistema son los mismos pasos que se aplican a los subsistemas conceptuales. Este método usa el término elemento de diseño para referirse genéricamente al sistema, un subsistema o un componente conceptual. Un elemento de diseño implementara una colección de responsabilidades y tienen una interfase conceptual que encapsula el conocimiento de los datos de entrada y salida. El método ABD finaliza una vez que se han tomado decisiones acerca de clases, métodos, procesos e hilos del sistema operativo.

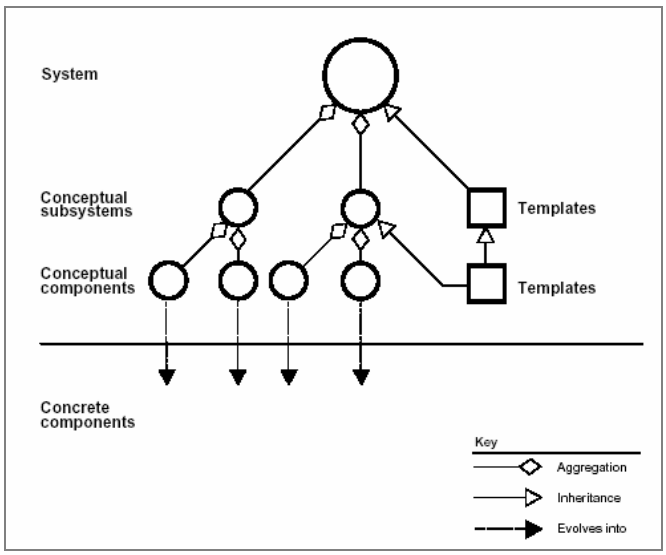

Figura 2-1. Proceso de descomposición de sistema [19]

Resumiendo este método, podemos decir que consta de los siguientes pasos: Descomposición del sistema, Selección de un estilo arquitectónico, Refinar plantillas y Verificación de los escenarios.

2.3.2 Desarrollo basado en Arquitecturas

De acuerdo con Bass y Kazman en [12] el desarrollo de la arquitectura de software es un paso crítico en el desarrollo de sistemas críticos, por lo tanto en 1999 propusieron un proceso para el desarrollo de arquitecturas de software llamado “Architecture-Based Development” Ver Figura 2-2. Los pasos de dicho proceso son:

- Obtención de los requerimientos de la arquitectura.-Este paso genera un listado de:
  a) Los requerimientos funcionales (casos de uso)
  b) Requerimientos arquitectónicos específicos y
  c) Escenarios de calidad.

- Diseño de la arquitectura.- Esta fase de diseño esta expresada en vistas arquitectónicas como:
  a) Funcional
  b) Código
  c) Concurrencia
  d) Física y
  e) Desarrollo.
- Documentación de la arquitectura.- Se diseña para que sirva de base y soporte a los programadores y analistas. En esta fase los autores dan una serie de recomendaciones para generar una documentación mantenible.
- Análisis de la arquitectura.- Evalúa la arquitectura para identificar riesgos potenciales y verificar los requerimientos de calidad que han sido tomados en cuenta en el diseño. Dicha evaluación se hace por medio del método ATAM.
- Generación de la arquitectura.- Codificación de la arquitectura a partir del diseño generado.
- Mantenimiento de la arquitectura.- En esta fase se dan una serie de recomendaciones para tener una arquitectura mantenible, ya que consideran que es un riesgo no tener la arquitectura documentada y consistente con las decisiones de diseño.

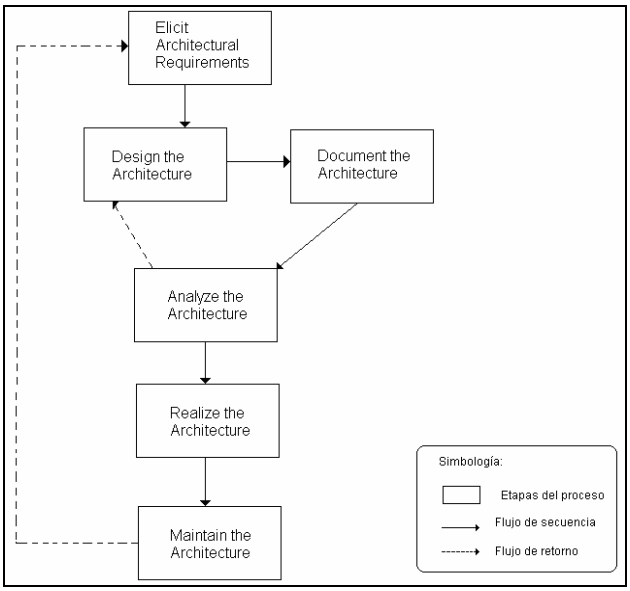

Figura 2-2. Pasos del proceso Basado en Arquitecturas [12]

2.3.3 Proceso de Arquitecturas de Software

De acuerdo con Bredemeyer Consulting [13], el proceso de arquitecturas de software se divide en los siguientes pasos; ver Figura 3-1:

- Inicio/Commit.- En esta fase se forma el equipo de trabajo, se alineas al equipo con la visión y el propósito de la arquitectura y se desarrolla el plan del proyecto.
- Requerimientos de la arquitectura.-Se establecen y documentan los requerimientos arquitectónicos.
- Estructuración del sistema.- El propósito de esta fase es diseñar el sistema a alto nivel y especificar las guías arquitectónicas para los diseñadores.
- Validación de la arquitectura.- En esta fase se valida que la arquitectura conozca los requerimientos e identifica issues y áreas de mejora.
- Deployment de la arquitectura.- Asegura que la arquitectura este siguiéndose en los productos de diseño y en el sistema.

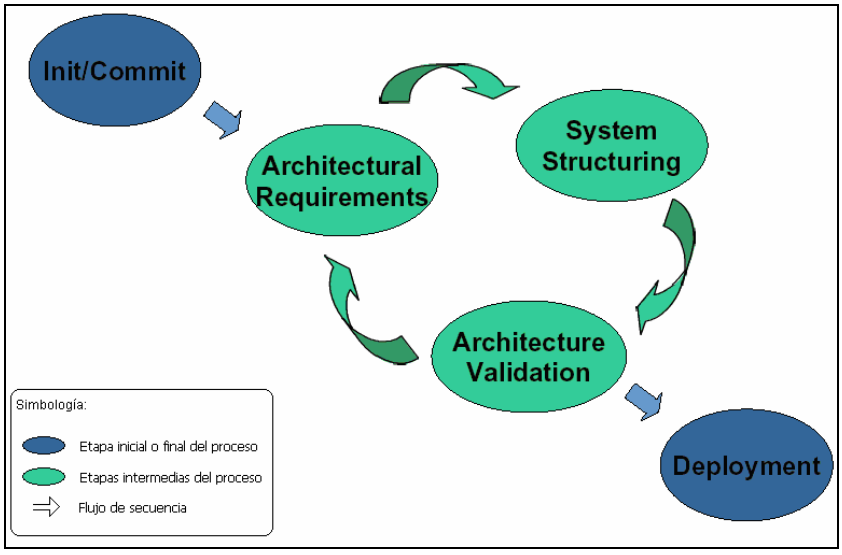

Figura 2-3. Proceso de Arquitecturas de Bredemeyer Consulting [13]

3 Descripción del problema

Uno de los aspectos críticos en el diseño y la construcción de un sistema de software complejo es su arquitectura, actualmente el proceso de diseño de la arquitectura de software no esta del todo formalizado y a menudo es más intuitivo que sistemático. No es si no hasta 1990 cuando el termino de "Arquitectura de Software" comenzó a tomar aceptación tanto en la comunidad de investigadores como en la industria. Profesionales en el diseño de arquitecturas han notado que el diseño y el desarrollo de una buena arquitectura es un factor crítico de éxito y comenzaron a reconocer el valor en la toma de decisiones sobre el estilo arquitectónico, patrones de diseño, frameworks, etc., que utilizarán para el desarrollo de nuevos productos. [5]

Algunos de los problemas más críticos a los que se han enfrentado los arquitectos de software y que han sido tema de estudio de los investigadores son:

- Los procesos de arquitecturas de software no están definidos de una manera detallada y no se encuentran en una mejora continua.
- Quizá la actividad más compleja durante el desarrollo de aplicaciones es la transformación de la especificación de los requerimientos en la implementación del sistema [5] Ver Figura 3-1. Aunque hay que señalar que a pesar de que ésta disciplina de la Ingeniería de Software; Requerimientos, ha sido también un tema de estudio, el presente trabajo usará la especificación de requerimientos como entrada al diseño de la arquitectura.

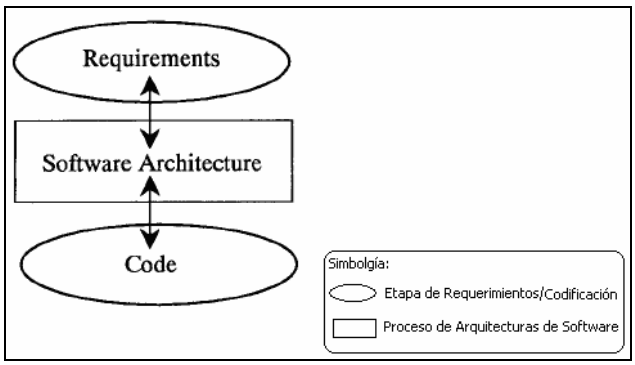

Figura 3-1. La Arquitectura de Software como un puente

- Los requerimientos de calidad (RQs) influencian fuertemente a la arquitectura de software y según la experiencia de [06] éstos son generalmente tratados con un proceso informal durante el diseño de la arquitectura. Siendo el principal foco de atención los requerimientos funcionales del sistema, por lo que no se presta mucha atención en los requerimientos de calidad [6].
- La evaluación de la arquitectura es otro factor crítico, al cual no se le ha dado la valía que representa, debido a que los sistemas son diseñados e implementados sin una evaluación explicita y, sólo después, las pruebas son ejecutadas para determinar si los requerimientos de calidad han sido cubiertos. Y si el sistema no las cumple, entonces se rediseña [7].
- La documentación es otro de los issues al que los arquitectos de software deben poner especial interés debido a que a que la documentación de la estructura del sistema y sus propiedades tiene ventajas para el mantenimiento del mismo. Y como hemos comprobado y leído en algunos casos de estudio, se pierde mucho tiempo tratando de entender el código del sistema [7]. Pero sabemos que no sólo el mantenimiento de software se ve beneficiado ya que muchas veces la

documentación no es efectiva en comunicar la información esencial a los stakeholders de un sistema que esta en su etapa de desarrollo, muchos de los problemas de la comunicación no efectiva tienen que ver con la notación informal que utilizan los arquitectos de sistemas de software. Para esto se han hecho algunos avances en el uso por ejemplo de UML, OCL, etc.

- La existencia de métodos que desacoplan la implementación de código de la arquitectura, lo cual permite tener inconsistencias, violar propiedades arquitectónicas e inhibir la evolución de la arquitectura de software. [9]
- La descripción de la arquitectura de manera formal, a través de los ADL's. [9] [10]

Finalmente y a pesar de los progresos que han surgido en el área de arquitectura de software, éste campo permanece todavía inmaduro. Por lo que el presente trabajo pretende iniciar una investigación en esta disciplina de la Ingeniería de Software junto con una mezcla de la metodología de Seis Sigma llamada DFSS.

4 Propuesta de solución

En esta sección se describe la propuesta de solución del proceso de arquitecturas de software basado en DFSS llamado PASWDFSS. El cual se encuentra basado en el proceso DFSS y las diferentes aportaciones que fueron descritas en la sección anterior.

PASWDFSS es un proceso de arquitecturas de software que permite crear una arquitectura nueva, a partir de los requerimientos arquitectónicos de la misma. Los pasos que el proceso presenta, mismos que son mostrados en la Figura 4-1 son:

1. Identificación de requerimientos de la Arquitectura
2. Caracterización del diseño de la Arquitectura
3. Documentación de la Arquitectura
4. Optimizar el diseño de la Arquitectura
5. Validación del diseño de la Arquitectura

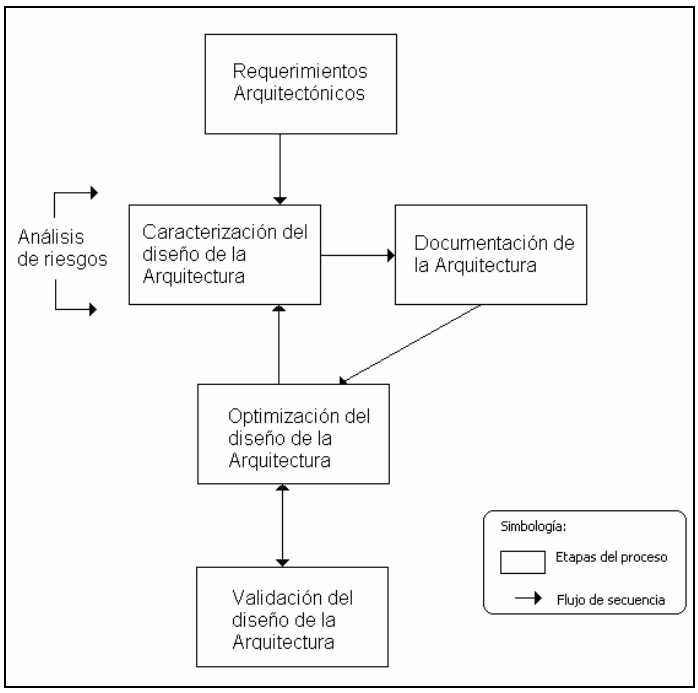

Figura 4-1. Proceso de Arquitecturas de Software basado en DFSS

La Figura 4-1 muestra el modelo general de la solución propuesta para el proceso de arquitecturas de software. Donde como puede verse existe la etapa de requerimientos, la cual tiene por objetivos principales, la creación del equipo de trabajo, selección de los métodos que se utilizarán para la elicitación de requerimientos, la obtención de los mismos, la traducción de los requerimientos a requerimientos funcionales y atributos de calidad medibles y finalmente la categorización de los requerimientos.

En la etapa de Caracterización del diseño de la Arquitectura, el objetivo principal es la creación de alternativas de diseño, es decir la creación de entidades de diseño y sus relaciones, con el objetivo de que cumplan los requerimientos funcionales y atributos de calidad. Dentro de esta fase se encuentra inmerso uno de los pasos de gran importancia: la evaluación de la arquitectura, que como sabemos el principal objetivo de esta etapa es verificar que la arquitectura

de software propuesta este cumpliendo con los requerimientos funcionales y de calidad o bien si existe un conflicto entre ellos que este impidiendo la combinación de los mismos.

La etapa de documentación, tiene como objetivo el representar la arquitectura en prosa, usando una notación estándar y una plantilla estándar según ANSI/IEEE 1471-2000.

Para la etapa de optimización del diseño de la Arquitectura el principal objetivo en base a la evaluación de la misma en el punto anterior, poder hacer los reajustes necesarios en el diseño para poder contemplar en mayor medida todos los atributos de calidad.

Finalmente en la etapa de validación cuyo objetivo es validar que la arquitectura se encuentre de acuerdo con los requerimientos establecidos por los stakeholders. En esta etapa tenemos un flujo doble con la etapa de optimización, esto es debido a que una vez que se valide la arquitectura esta puede sufrir cambios, por lo que deberá pasar por los pasos definidos en la etapa mencionada.

En la siguiente sección se describirán a detalle cada una de las etapas del proceso PASWDFSS siguiendo los siguientes lineamientos: serán descritas las entradas, salidas y los pasos internos de cada una de las etapas del proceso. Se seguirá una notación estándar para los diagramas necesarios en cada etapa así como para la explicación de los mismos.

4.1 Proceso para el desarrollo de Arquitecturas de Software basado en DFSS

El modelo que se esta proponiendo en el presente trabajo esta basado en los procesos descritos en la sección anterior y específicamente en [12] y toma también como referencia el modelo desarrollado por José Hernández [14]. Así como también se ajusta a las etapas definidas el diseño de seis sigma o DFSS.

En la Figura 4-2 se muestra un diagrama en donde puede verse la relación que existe entre PASWDFSS y [12], PASWDFSS y el proceso definido por DFSS.

En cuanto a la relación que existe entre PASWDFSS y el proceso de desarrollo basado en Arquitecturas [12] de Kazman y Bass en la Figura 4-2, podemos ver que el proceso [12] contiene los siguientes pasos: 1) obtención de los requerimientos de arquitecturas, 2) diseño de la arquitectura de software, 3) documentar la arquitectura de software, 4) analizar la arquitectura de software, 5) generar la arquitectura de software y finalmente 6) el mantenimiento de la arquitectura de software. Este proceso coincide con el proceso que estamos proponiendo PASWDFSS en los cuatro primeros pasos, ya que el quinto que es generación de la codificación de la arquitectura quedo fuera del alcance de éste proyecto, aunque hay que señalar que si se han hecho algunos avances como puede verificarse en [09].

En cuanto al paso seis, mantenimiento de la arquitectura, no se tomo en cuenta en nuestro modelo debido a que DFSS se enfoca al diseño de nuevos productos y/o servicios, y por lo tanto no esta definido, pero nosotros consideramos que si es una fase necesaria dentro del proceso, ya que como sabemos todos los sistemas tendrán cambios o mejoras. Ver Figura 4-2.

En cuanto a las mejoras que se proponen en PASWDFSS es que se agrega una etapa de validación de la arquitectura, la cual tiene el objetivo de poder validar que el diseño de la arquitectura de software cumple con las necesidades y expectativas de todos los stakeholders, otra de las fases que se agregó es la optimización de la arquitectura, cuyo objetivo es realizar las adecuaciones ó ajustes necesarios una vez que se ha hecho la verificación de la arquitectura. Y finalmente uno de los pasos de la etapa de Caracterización del diseño de la arquitectura es hacer un análisis de riesgos, donde se analizan los riesgos

potenciales de cumplir o no los atributos de calidad de la arquitectura, así como sus posibles planes de mitigación y contingencia en caso de presentarse alguno de ellos.

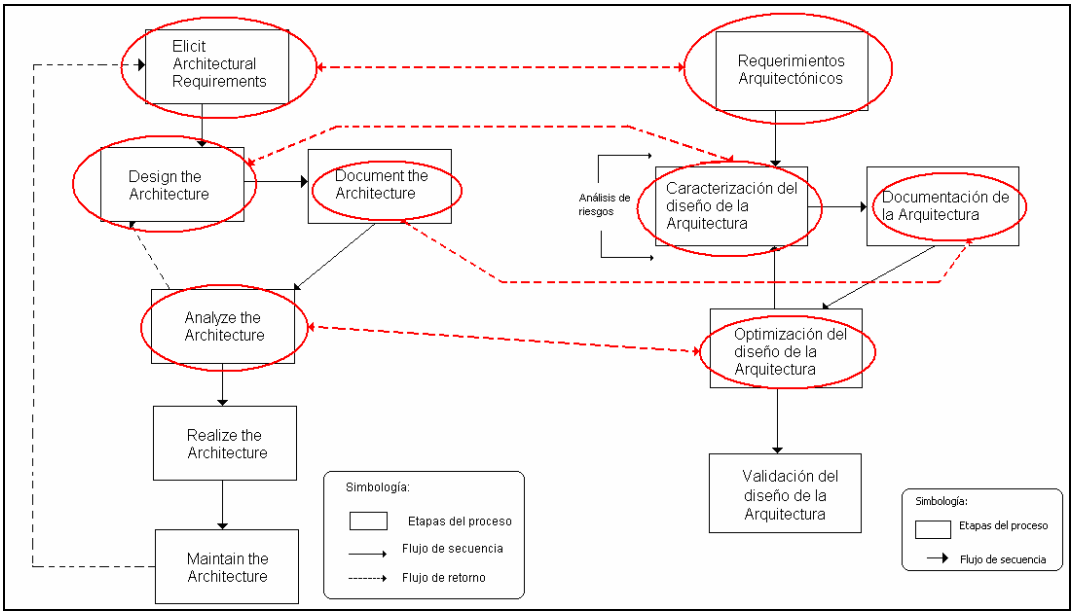

Figura 4-2. Relación entre Architecture-Based Development y PASWDFSS

En cuanto a la relación que existe entre PASWDFSS y el proceso DFSS podemos ver en la Figura 4-3, que realmente guardamos una relación muy estrecha en las etapas propuestas por DFSS aunque hay que señalar que se cambio el nombre de la primera etapa, de “Identify” por “Requerimientos arquitectónicos” por ser un nombre más adecuado en el contexto de arquitecturas de software. En cuanto a las demás etapas realmente se respetaron, aunque hay que señalar que se agrego una etapa llamada documentación de la arquitectura, debido a que como se dijo en la sección de la problemática, existe actualmente en la Ingeniería de Software en el área se Arquitecturas de Software un grave problema en cuanto a documentación se refiere, por lo tanto PASWDFSS al igual que los procesos definidos en la sección anterior la incluyen. Como puede verse en la Figura 4-3, las etapas de los procesos es casi la misma, lo que está adecuado son los pasos en cada una de ellas, así como algunas de las herramientas, que en algunos casos ya se han adecuado a Software y otras quedarán pendientes de dicha adecuación. Este proyecto pretende ser la columna vertebral de un proceso de arquitecturas de software basado en DFSS ya que se considera que una de las ventajas es que algunos como [17], han encontrado aplicando DFSS que es un proceso que ya esta definido y que no solo te dice los pasos que debes seguir en cada una de las etapas, si no que también propone el uso de herramientas que en algunos casos se han adaptado a Software y que en mi opinión son de mucha ayuda, aunque el principal problema es que pocos ingenieros de software ponen en práctica las investigaciones y propuestas que hacen, por lo tanto la inmadurez que se tiene en este campo de la Ingeniería de Software se ira erradicando si se ponen en práctica lo investigado.

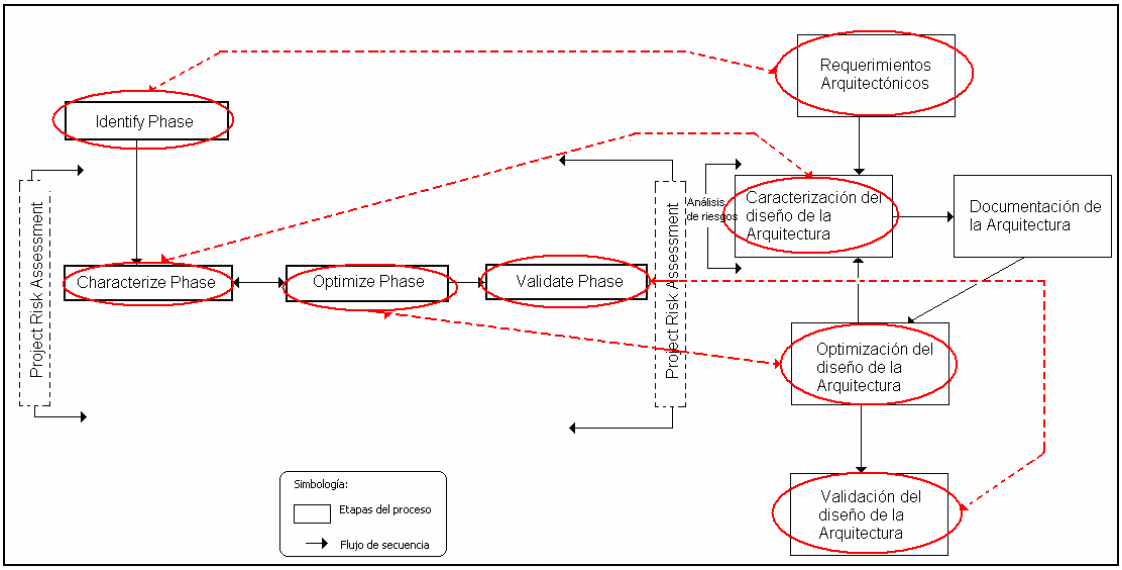

Figura 4-3. Relación entre el proceso definido en DFSS y PASWDFSS

De esta manera y como hemos explicado anteriormente, el modelo PASWDFSS toma lo mejor de [12] y DFSS para especificar sus fases, adicionando pasos que se consideran importantes y que harán más completo el proceso.

Las siguientes secciones describirán cada una de las etapas propuestas en el modelo PASWDFSS tomando en cuanta los siguientes lineamientos:

- La notación para modelar los pasos del proceso está alineados a los propuestos por Ericsson y Penker en [15]. Ver Figura 4-4.

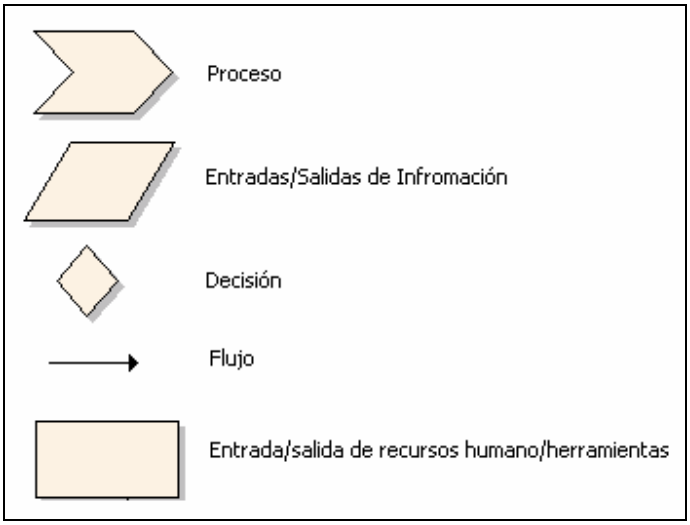

Figura 4-4.Simbología utilizada en el proceso de arquitecturas de software

- Se consideraran los artefactos de entrada a cada una de las etapas del proceso
- Los pasos internos en cada una de las etapas del proceso
- Artefactos de salida de las etapas del proceso
- Se hace mención de las posibles herramientas que servirán de apoyo en las etapas del proceso.

4.1.1 Requerimientos Arquitectónicos

Esta etapa tiene por objetivos principales, la creación del equipo de trabajo, selección de los métodos que se utilizarán para la elicitación de requerimientos, la obtención de los mismos, la traducción de los requerimientos a requerimientos funcionales y atributos de calidad medibles y finalmente la categorización de los requerimientos. Ver Figura 4-5.

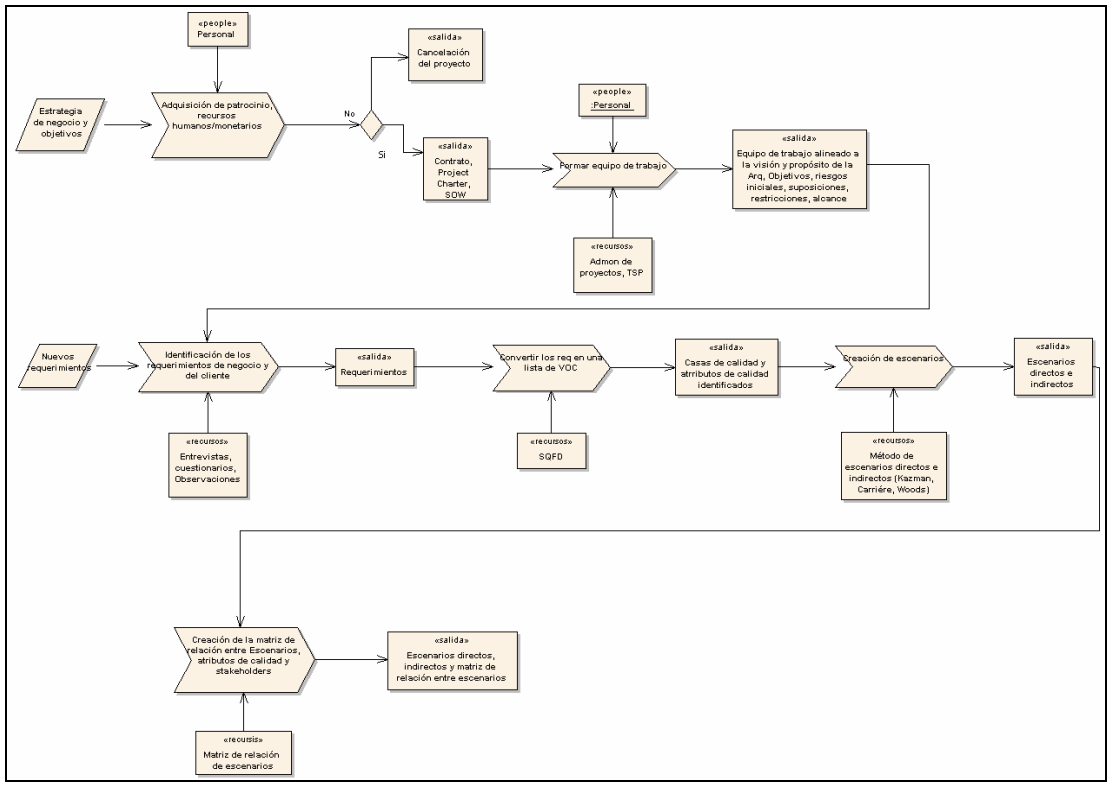

Figura 4-5. Etapa de Requerimientos Arquitectónicos

Dentro de esta etapa se encuentran subetapas que seguirán el mismo estándar de documentación mencionado en la sección anterior. Los pasos a seguir en la etapa de Requerimientos Arquitectónicos son los siguientes:

Paso 1: Inicio.

El paso de inicio, tiene que ver con la creación del equipo de trabajo para el diseño de la Arquitectura, así como de la creación del Project charter y el Plan del proyecto. El acta de constitución del proyecto o Project charter, "es el documento que autoriza formalmente un proyecto. Este documento confiere al director del proyecto la autoridad para aplicar recursos de la

organización a las actividades del proyecto." [16]. El plan del proyecto deberá ser un plan general donde se indiquen los hitos principales del proyecto.

Las actividades a seguir de este paso son:

- Adquisición de patrocinio (si aplica).
- Verificación de la existencia de recursos humanos, no humanos, monetarios (si aplica) y capacitación necesarios para el diseño de la Arquitectura.
- Formar y alinear el equipo de trabajo con la visión y el propósito de la Arquitectura.
- Establecer:
    - Los objetivos principales de la Arquitectura
    - El alcance de la Arquitectura
    - Los criterios de aceptación de la Arquitectura
    - Las suposiciones de la Arquitectura
    - Y finalmente los riesgos iniciales asociados a la Arquitectura
- Creación del plan del proyecto de Arquitecturas

La siguiente tabla resume el paso número uno; Inicio, de la etapa de Requerimientos Arquitectónicos:

Elemento	***Descripción***
Entradas	• Contrato. • SOW (Statement of work). • Equipo de trabajo (personal). • Estrategia de negocios y objetivos.
Salidas	• Plan de trabajo de la Arquitectura. • Equipo de trabajo (roles y responsabilidades). • Project Charter.
Herramientas y/o técnicas	• Administración y planeación de proyectos. • PMBOOK • Graficas de Gantt, Pert • TSP (Creación de equipos de trabajo, roles y responsabilidades, creación de plan de trabajo)
Roles	• Analista. • Arquitecto.

Tabla 4-1. Tabla de elementos del paso 1: Inicio de la fase de Requerimientos Arquitectónicos

Paso 2: Identificación de requerimientos de negocio y del cliente.

El paso de identificación de requerimientos de negocio y del cliente tiene como objetivo el identificar a los principales stakeholders del proyecto, sus necesidades así como analizar dichas necesidades.

Las actividades a seguir en este paso son:

- Identificar los métodos para obtener las necesidades y deseos de los clientes
- Identificación de los principales clientes
- Obtener las necesidades (conocidos como los WHATs en DFSS) y deseos de los clientes en función a requerimientos funcionales y atributos de calidad

- Identificar los atributos de satisfacción del cliente, cada atributo es clasificado acorde a su importancia relativa para el cliente.
- Transformar las necesidades y deseos de los clientes a una lista de la voz del cliente (VOC, voice of customer) a través de la herramienta SQFD [17]. Ver anexo A.
- Identificar los CTSs en: requerimientos críticos de calidad (CTQ, critical-to-quality), los críticos a liberarse (CTD, critical-to-delivery), y así sucesivamente.

Los CTS (critical-to-satisfaction) es un arreglo de características de diseño derivado de las respuestas de los WHATs. El arreglo de CTSs es también llamado el arreglo de los HOWs. Cada WHAT inicial debe tener uno o más HOWs describiendo el como alcanzar la satisfacción del cliente. Por ejemplo si el cliente desea un "carro bonito", este se puede conseguir teniendo en cuenta el diseño de los asientos, el espacio que se tendrá para las piernas, que sea un carro no ruidoso, sin vibraciones, etc. Todos estos son requerimientos del cliente que deben ser medidos y controlados y son lo que conocemos como las características de calidad, que para el caso de Arquitecturas de Software, requerimientos como modificabilidad, performance, portabilidad, etc, son los atributos de calidad y que pueden obtenerse vía la herramienta SQFD. Ver Figura 4-6

Las relaciones entre los CTS técnicos y el arreglo de WHATs son usados a menudo para priorizar las necesidades y deseos de los clientes que fueron plasmados en la matriz de QFD. Debido a que el equipo de trabajo asigna un valor que refleje el alcance para el cual el CTS definido contribuye a conocer los WHATs.

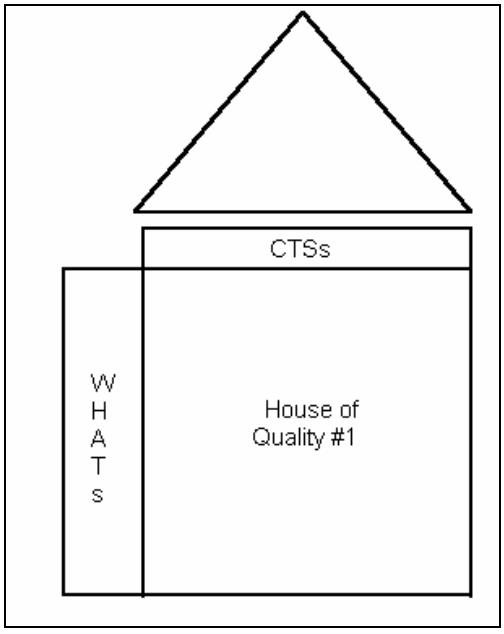

Figura 4-6. Herramienta: QFD

Con los pasos anteriormente mencionados se ha descrito como obtener los requerimientos arquitectónicos por medio de las casa de calidad para software SQFD [17]. El arquitecto de software deberá especificar de manera estándar los atributos de calidad y para esto recomendamos usar el estándar ISO 9126-1.

El siguiente paso y haciendo referencia al proceso definido por José Hernández en [14], en el paso de la especificación de escenarios directos e indirectos, se debe realizar la descripción del comportamiento de la arquitectura en base a escenarios de calidad (directos e indirectos) [21], donde los escenarios directos son enunciados cortos que describen la utilización normal que los

usuarios finales dan al sistema construido en base a la arquitectura de software, los escenarios indirectos son enunciados que representan cambios a la arquitectura de acuerdo a la utilización de nuevas plataformas de hardware o software, reemplazar alguna funcionalidad existente, o un componente, etc. (copiado de [14]).

Los escenarios son acciones relacionadas con los stakeholders y sus intereses en el sistema, mismos que tienen como base los atributos de calidad.
Teniendo los escenarios podemos asegurar que los requerimientos están cubiertos por la arquitectura, pero como sabemos tenemos varios stakeholders interesados de diferente manera en los atributos del sistema, por lo que Kazman, Carrier y Woods, proponen el uso de una matriz tridimensional que relaciona los escenarios con los stakeholders y los atributos de calidad. Ver Figura 4-7.

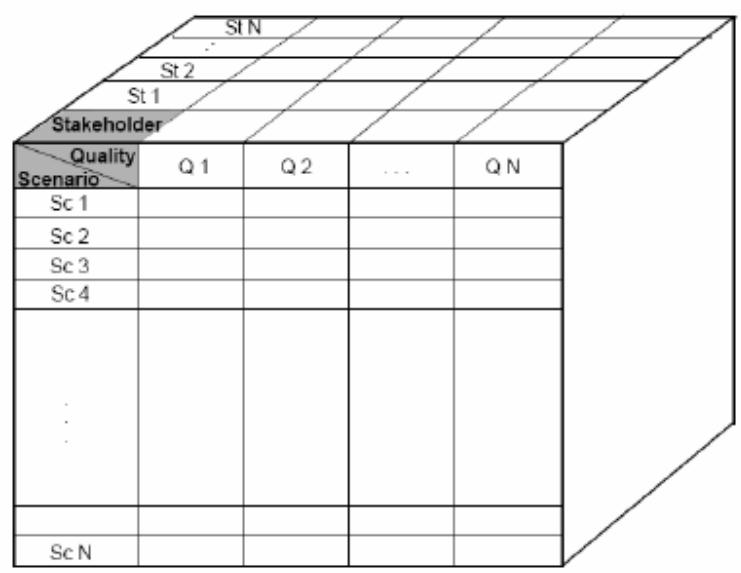

Figura 4-7. Matriz de relación entre escenarios

La siguiente tabla resume el paso número dos; Identificación de requerimientos de negocio y del cliente, de la etapa de Requerimientos Arquitectónicos:

Elemento	***Descripción***
Entradas	• Necesidades y deseos del cliente (Documento de requerimientos del cliente)
Salidas	• Casas de calidad • Escenarios de Calidad • Matriz de relación de escenarios • Requerimientos de calidad
Herramientas y/o técnicas	• SQFD • Investigación de mercado/clientes • Análisis de riesgos • Métodos para recolección de requerimientos: o Entrevistas

	○ Cuestionarios ○ Prototipos ○ Observación • Análisis de Kano • Métodos analíticos (rate monotonic, concurrencia, etc)
Roles	• Analista. • Arquitecto.

Tabla 4-2. Tabla de elementos del paso 2: Identificación de los requerimientos del cliente y negocio

4.1.2 Caracterización del diseño de la Arquitectura

La etapa de caracterización de la arquitectura tiene como principales objetivos el diseño de alternativas de diseño arquitectónico en base a los requerimientos y atributos de calidad del cliente, así como la verificación de dicha arquitectura. Ver Figura 4-8

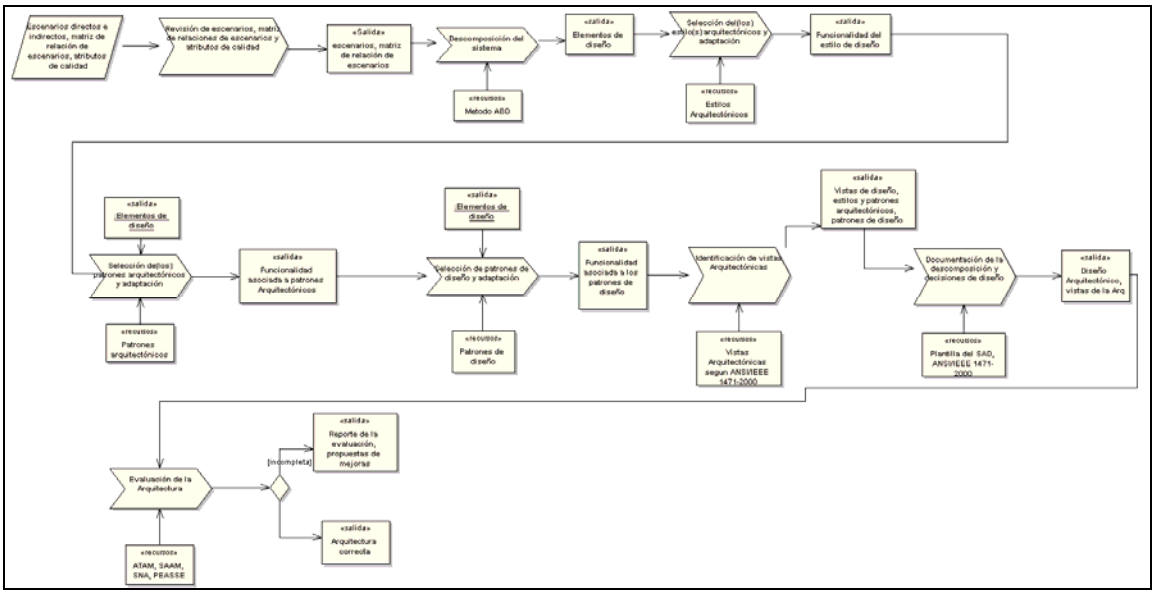

Figura 4-8. Etapa de Diseño de la Arquitectura de Software

Paso 1: Diseño de la Arquitectura

Este paso tiene por objetivo el diseño de la Arquitectura de software en base a los requerimientos del cliente, para esto se hará uso de método ABD [18] el cual provee una simple y poderosa estructura.

Las actividades a seguir en este paso son:

- Revisión de los escenarios, matriz de relación de escenarios y atributos de calidad
- Desde el nivel más alto del sistema, descomponer en subsistemas conceptúales tratando de identificar los estilos arquitectónicos y patrones arquitectónicos y de diseño que puedan ser útiles.
- A nivel subsistema descomponer en componentes conceptuales o módulos con responsabilidades específicas, a este nivel también se identificaran patrones arquitectónicos.
- Una vez que se ha hecho la descomposición y llegado a nivel de componentes conceptuales se podrán descomponer componentes concretos que representan los elementos de diseño en la arquitectura.
- Los elementos de diseño deberán tener una responsabilidad en específico, y conceptualizarse por medio de estilos, patrones y reglas que serán determinadas por los requerimientos y los atributos de calidad.
- Una vez realizada la descomposición de la funcionalidad del sistema, se procederá a la selección de los estilos arquitectónicos en los cuales estará basada la arquitectura, así como su adecuación a las necesidades de la arquitectura.

- Se asignarán en base a la descomposición del sistema, los componentes que estarán implementado un(unos) estilos. Identificando las interfases de cada uno de los componentes.
- Identificación y selección de los patrones arquitectónicos para un conjunto definido de elementos de diseño.
- A nivel más bajo de la descomposición se pueden identificar patrones de diseño que pueden ser útiles y adaptados a las necesidades especificas.
- Cabe mencionar que en base a los escenarios generados en la fase anterior, se podrán generar más de una arquitectura candidata, con lo cual el paso de la evaluación determinará que arquitectura es la adecuada.
- Identificar las vistas arquitectónicas que formarán parte del diseño de la arquitectura.
- Documentación de la descomposición del sistema y decisiones tomadas.

Las vistas que formaran parte del diseño de la arquitectura y parte de la documentación están basadas en las propuestas por el ANSI/IEEE 1471-2000, el cual provee una guía para seleccionar el mayor conjunto de vistas para documentar, que soporte los intereses de los stakeholders. Este estándar describe un conjunto de vistas para satisfacer a la comunidad de stakeholders. Una vista identifica el conjunto de intereses para ser enfocados. Ésta es una representación de un conjunto de elementos de software, sus propiedades, y las relaciones entre ellos. La elección de un conjunto de vistas que muestran la arquitectura completa y todo de las propiedades relevantes.

Las vistas arquitectónicas pueden dividirse en tres grupos principales conocidos como viewtypes [20]. Dependiendo de la naturaleza de los elementos que ellas muestran, estos grupos pueden ser:

- Module viewtype. En este grupo de vistas los elementos son módulos, los cuales son unidades de código que implementan un conjunto de responsabilidades. Un módulo puede ser una clase, una colección de clases, una capa (layer) ó cualquier descomposición de la unidad de código. Cada módulo tiene una colección de propiedades asignadas a él, y éstas intentan expresar la información importante asociada con el módulo, por ejemplo sus restricciones. Los módulos pueden tener relaciones con otros módulos a través de relaciones como: “is part of” ó “inherits from”. Algunos de los estilos arquitectónicos contenidos en ésta categoría son: descomposition, uses, generalization y layered.

- Component-and-connector viewtype. Los estilos en éste tipo de vistas expresan comportamiento en tiempo de ejecución. Éstos están descritos en términos de componentes y conectores. Donde un componente es una de las principales unidades de procesamiento de ejecución de Sistemas; un conector es un mecanismo de interacción entre los componentes. Objetos, procesos o colección de objetos pueden ser componentes y conectores incluyen: tuberías, repositorios, sockets, etc. El Middleware puede ser visto como un conector entre los componentes que lo usan. Algunos de los estilos arquitectónicos contenidos en ésta categoría son: Pipe and Filters, shared data, publish&suscribe, client-server, peer to peer y communicating processes.

- Allocation viewtype. Este grupo de vistas muestra las relaciones entre los elementos de software y los elementos en uno ó más ambientes externos en los que la arquitectura de software es creada y ejecutada (hardware, archivos del sistema, equipo de desarrollo, etc). Algunos de los estilos arquitectónicos contenidos en ésta categoría son: deployment, implementation y work assigment.

La siguiente tabla resume el paso número uno; diseño de la Arquitectura, de la etapa de Caracterización del diseño de la Arquitectura:

Elemento	***Descripción***
Entradas	• Escenarios de calidad • Matriz de relación de escenarios
Salidas	• Diseño arquitectónico • Vistas de la Arquitectura
Herramientas y/o técnicas	• TRIZ • QFD • Axiomatic Design • Diseño Robusto • Simulación • DFMEA • Estilos Arquitectónicos • Patrones de diseño
Roles	• Arquitecto.

Tabla 4-3. Paso 1. Diseño de la Arquitectura; de la etapa dos del proceso

Paso 2: Evaluar las alternativas de diseño

El objetivo de la etapa de evaluación de las alternativas del diseño de arquitecturas es poder verificar que dicho diseño cumple con los requerimientos de calidad definidos.

Las actividades a seguir en este paso son:

- Definir el objetivo de la evaluación.
- Selección del método de evaluación en base a los atributos de calidad a analizar.
- Selección de los métodos analíticos para verificar el cumplimiento de los atributos de calidad.
- Seguir los pasos de la evaluación de la Arquitectura.
- Generación de reporte de evaluación.

En esta etapa podemos tomar en cuenta la recomendación de [18] y evaluar la arquitectura en base a escenarios, ya que en base a su experiencia la evolución basada en escenarios es muy útil. Aunque hay que mencionar que la Simulación, el modelado matemático y/o el razonamiento objetivo pueden ser alternativas en la evaluación[18].

La siguiente tabla resume el paso número dos; Evaluación de la Arquitectura, de la etapa de Caracterización del diseño de la Arquitectura:

Elemento	*Descripción*
Entradas	• Diseño de la Arquitectura, escenarios directos e indirectos.
Salidas	• Evaluación de la Arquitectura • Propuestas de mejoras
Herramientas y/o técnicas	• Revisiones de diseño • ATAM • SAAM • ABAS • SNA
Roles	• Arquitecto

Tabla 4-4. Paso 2: Evaluación de la Arquitectura; de la etapa dos del proceso

4.1.3 Documentación de la Arquitectura

La documentación de la arquitectura es una de las etapas que se ha descuidado por parte los arquitectos de software, pero la experiencia nos ha indicado que es fundamental debido a que nos sirve como: medio de comunicación entre los stakeholders y el equipo de trabajo, plasma de manera textual y gráfica el diseño y las decisiones tomadas para el mismo, así como los análisis analíticos realizados para la comprobación de los atributos de calidad, etc. Ver Figura 4-9.

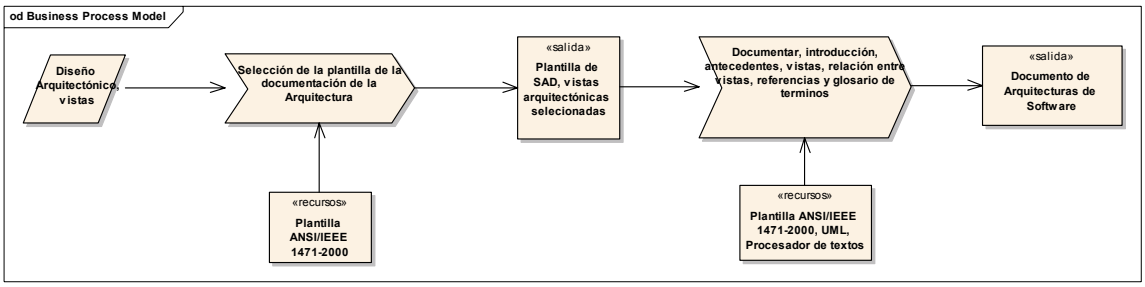

Figura 4-9. Etapa de documentación de la Arquitectura

Algunas de las recomendaciones de esta etapa que tomamos de [12] son:

- La documentación debe ser navegable y completa.
- Se deben especificar las suposiciones y restricciones de diseño que se definieron por parte del cliente.
- La documentación debe estar disponible para todos los stakeholders.

Las actividades a seguir en este paso son:

- Escoger la plantilla que servirá de base para documentar la arquitectura, en este caso hacemos la recomendación de usar la plantilla de ANSI/IEEE 1471-2000.
- Llenar la plantilla siguiendo las indicaciones propuestas, aquí cabe señalar que no es necesario que se llenen todos los puntos de la plantilla, sólo los que a consideración del arquitecto sean necesarios y relevantes.

La siguiente tabla resume la etapa de Documentación de la Arquitectura:

Elemento	***Descripción***
Entradas	• Diseño de la Arquitectura, escenarios directos e indirectos y vistas arquitectónicas
Salidas	• Documento de Arquitecturas de software
Herramientas y/o técnicas	• ANSI/IEEE 1471-2000 • UML
Roles	• Arquitecto • Documentador

Tabla 4-5. Etapa de Documentación de la Arquitectura

4.1.4 Optimización del diseño de la Arquitectura

La optimización de la Arquitectura permite realizar las adecuaciones necesarias al diseño de la Arquitectura. Ver Figura 4-10.

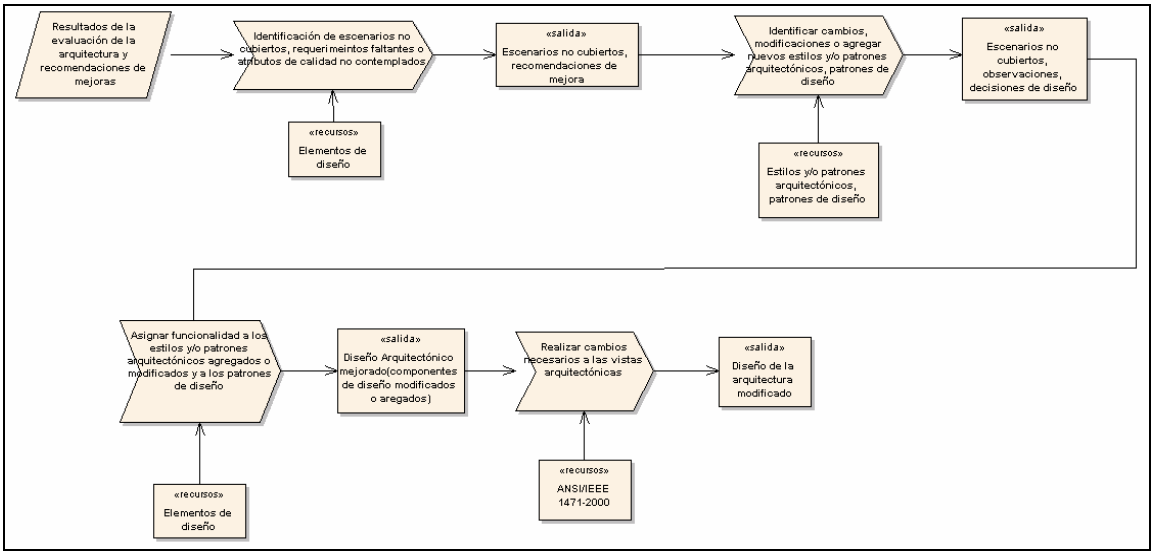

Figura 4-10. Etapa de Optimización de la Arquitectura

Las actividades a seguir en esta etapa son:

- En base a la evaluación realizada y al reporte generado en la misma, definir:
  - Los escenarios no contemplados que pueden ocasionar modificaciones en el diseño de la Arquitectura.
  - Los requerimientos (funcionales como atributos de calidad) no cubiertos en el diseño de la Arquitectura de Software.
  - Identificación de nuevos requerimientos
- El equipo de trabajo junto con el Arquitecto de Software analizaran los issues del paso anterior para realizar las modificaciones necesarias al diseño propuesto o realizar una nueva evaluación si se considera el diseño lo suficientemente robusto como para soportar los issues detectados. Dentro del análisis deberán tomar en cuenta:
  - Cambios, modificaciones o adiciones en los estilos arquitectónicos y patrones de diseño seleccionados.
  - Asignar nueva funcionalidad a los estilos arquitectónicos y patrones de diseño añadidos o modificados.
  - Agregar nuevos componentes o módulos
  - Realizar los cambios necesarios en las vistas arquitectónicas.
  - Ir al paso de evaluación para verificar el cumplimiento de los requerimientos (tanto los que ya se tenían como los nuevos)
- Documentar los cambios realizados

La optimización de la Arquitectura permite realizar las adecuaciones necesarias al diseño de la Arquitectura.

La siguiente tabla resume la etapa de Optimización de la Arquitectura:

Elemento	*Descripción*
Entradas	• Diseño de la Arquitectura • Evaluación de la Arquitectura
Salidas	• Documento de Arquitecturas de Software adaptado a las recomendaciones que surgieron en la evaluación.
Herramientas y/o técnicas	• FMEA • Simulación
Roles	• Arquitecto

Tabla 4-6. Etapa de Optimización de la Arquitectura

4.1.5 Validación del diseño de la Arquitectura

De esta manera y como hemos explicado anteriormente, el modelo PASWDFSS toma lo mejor de [12] y DFSS para especificar sus fases, adicionando pasos que se consideran importantes y que harán más completo el proceso.

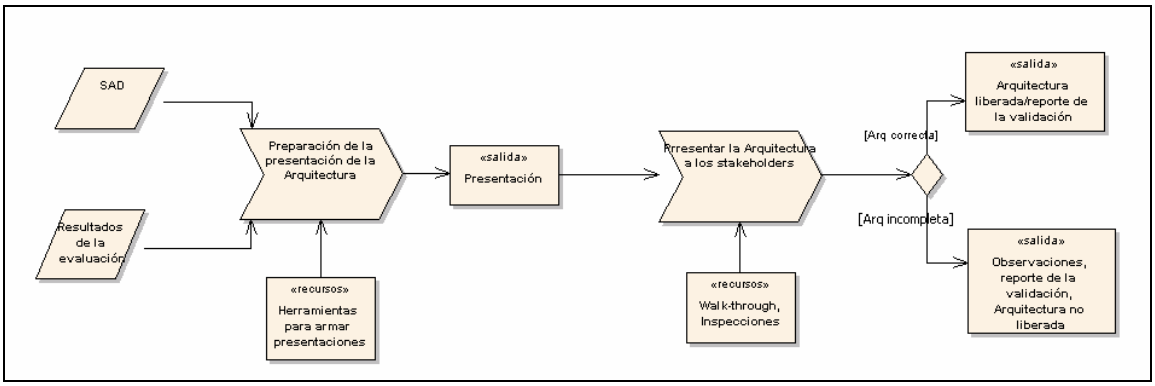

Figura 4-11. Etapa de validación de la Arquitectura

Las actividades a seguir en esta etapa son:

- Preparación de la presentación de la Arquitectura de Software
- Presentación de la Arquitectura de Software y de los resultados de la evaluación, así como las mejorar y adecuaciones realizadas a ésta.
- Generación de observaciones y/o solicitudes de cambio
- Generación de reporte final de la validación de la Arquitectura de Software
- Si el cliente indica que todo esta correcto:
    - Liberación entrega de la Arquitectura de Software
- En caso contrario:
    - La liberación no se realiza

La siguiente tabla resume la etapa de Documentación de la Arquitectura:

Elemento	***Descripción***
Entradas	• Documento de la Arquitectura (con las mejorar propuestas en la evaluación, si aplica), resultados de la evaluación.
Salidas	• Documento de Arquitecturas de software liberado • Documento de Arquitecturas de software no liberado con solicitud de cambios
Herramientas y/o técnicas	• Walk-through • Inspecciones
Roles	• Arquitecto • Stakeholders

Tabla 4-7. Etapa de Validación de la Arquitectura

5 Conclusiones y trabajo futuro

El proceso de Arquitecturas de Software es un paso muy importante en el desarrollo de sistemas de software complejos y grandes, donde intervienen un gran número de personas y cuyo objetivo es definir a alto nivel los componentes y la relación entre ellos, así como sus responsabilidades dentro de la misma. Cabe mencionar que es hasta 1990 que esta área ha tomado interés en la comunidad de investigadores y en la industria, ya que se ha comprobado que la Arquitectura de Software permite controlar y tomar decisiones basadas en las consecuencias (monetarias, de tiempo, recursos, etc.) que implican los cambios de requerimientos o de hardware en los sistemas que están basados en ella.

Por lo tanto la presente investigación se enfoco en definir un proceso de Arquitecturas de Software basado en DFSS, ver Figura 5-1. Este proceso define las entradas y salidas del proceso general, los pasos internos en cada etapa y el uso de herramientas.

Los pasos definidos para el proceso propuesto son los siguientes:

1. Identificación de los requerimientos de la Arquitectura.- Cuyo objetivo es la creación del equipo de trabajo, selección de los métodos de elicitación de requerimientos, obtención de los mismos, traducción de los requerimientos a requerimientos funcionales y de calidad medibles y cuantificables y su categorización.
2. Caracterización del diseño de la Arquitectura.- El objetivo principal es la creación de alternativas de diseño (arquitecturas candidatas) y la evaluación de las mismas.
3. Documentación de la Arquitectura.- En esta etapa se representa la arquitectura en posa, usando una notación estándar y una plantilla (se sugiere el uso de ANSI/IEEE 1471-2000.
4. Optimización del diseño de la Arquitectura.- En esta etapa se realizan los ajustes necesarios en base a la evaluación de la Arquitectura.
5. Validación del diseño de la Arquitectura.- En esta etapa se realiza la validación de la Arquitectura, para que ésta se encuentre alineada a los requerimientos de los stakeholders.

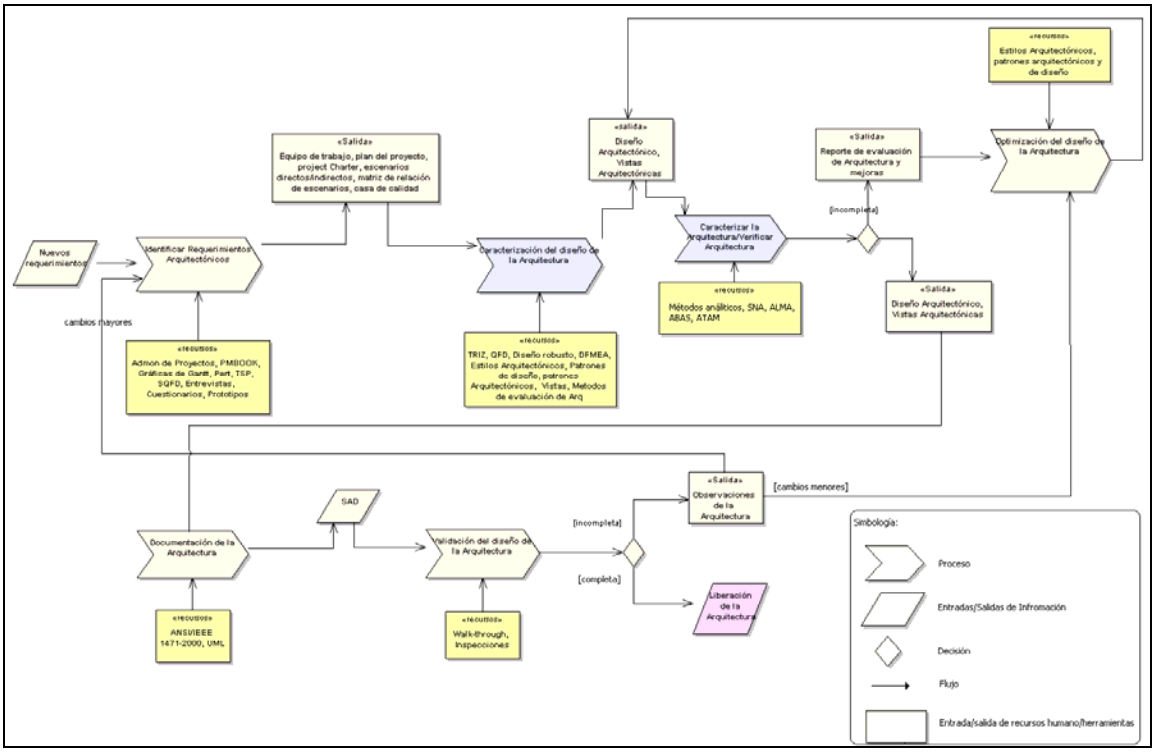

Figura 5-1. Proceso de Arquitecturas de Software

Finalmente algunos de los trabajos a futuro que fueron detectados en la presente investigación son:

- Adecuación de las herramientas que DFSS define en su proceso de desarrollo de nuevos productos y/o servicios, para software, tomando en cuento que no todas ellas pudieran adaptarse.
- Poner en práctica el proceso y hacerle mejoras.
- Incorporarle la etapa de mantenimiento, así como los pasos necesarios para que dicho proceso no solo sirva para el desarrollo de nuevas Arquitecturas, si no también para Arquitecturas que ya están definidas pero no documentadas.
- Incorporar métricas que permitan medir y posteriormente controlar el proceso.
- Formalizar el proceso definido, creando sus scripts y guías para que ayude a los Arquitectos en su uso y seguimiento.

6 Anexo A Quality Function Deployment (SQFD)

La ***Calidad*** en el Software esta sobresaliendo como uno de los principales issues en el ciclo de vida del desarrollo de un producto de Software. Ahora establecer la calidad de un producto es igualmente importante que el desarrollo de la eficiencia. [17]

El presente anexo, presenta un resumen de los resultados de la adaptación del uso de la herramienta QFD (Quality Function Deployment) para Software, por la firma MVS.

La administración total de la calidad (TQM, Total Quality Management) es un aspecto importante en el mejoramiento de la calidad en muchas organizaciones. Donde QFD, es un vehiculo de implementación de TQM, y que la firma MVS adapto a Software con el objetivo de mejorar la calidad en los sistemas de software (SQFD).

SQFD se enfoca en mejorar la calidad del software desarrollando procesos para implementar técnicas de mejoramiento de calidad durante la fase de elicitación de requerimientos del ciclo de vida del desarrollo del sistema. Esta técnica permite incrementar el análisis y productividad del programador, una reducción en el número de errores que pasan de una fase a otra y la satisfacción del cliente.

La firma MVS reconoce que las organizaciones derivan una gran parte de sus fondos a la producción y subsecuentemente a la venta de software, pero también dice que si estas empresas adaptan y usan QFD en el desarrollo de software, podrían tener mayores ingresos. En general la tendencia de la utilización de técnicas de mejora de calidad ha ido incrementándose, ver Figura 6-1.

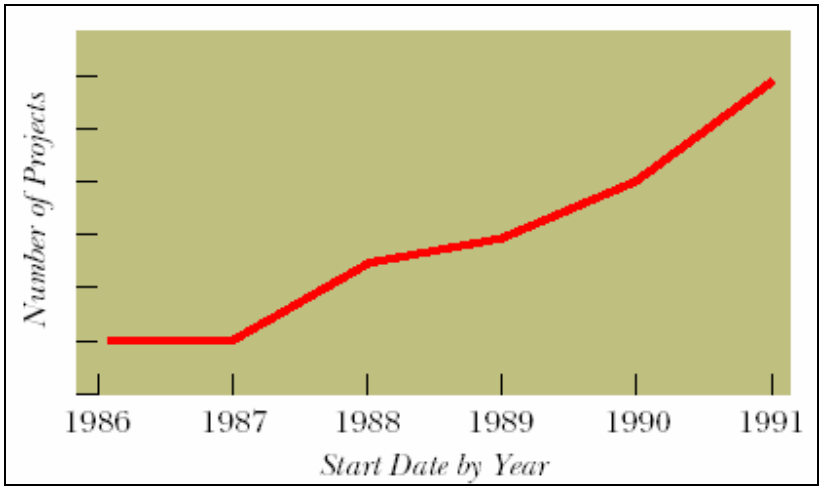

Figura 6-1. Estadística del uso de técnicas de mejoramiento de la calidad [17]

El concepto de SQFD se origino en Japón, tal como QFD. La transferencia metodológica de la tecnología QFD a SQFD se escribió en un libro en 1984, cuando los japoneses empezaron a explorar su uso en el desarrollo de software embebido.

Los beneficios citados por SQFD según [17] son:

- Fomenta una mejor atención a las perspectivas del cliente
- Crea una mejor comunicación entre departamentos
- Permite justificar la toma de decisiones
- Cuantificar los requerimientos del cliente
- Representa los datos para facilitar el uso de métricas

- Evita las perdidas de información
- Puede ser adaptado al desarrollo de productos de software, entre otros

Algunas de las organizaciones que han publicado material con respecto a SQFD son: DEC, AT&T, Hewlett-Packard, IBM y Texas Instruments.

Finalmente describiremos brevemente algunas estadísticas presentadas por los estudios realizados por MVS en cuanto al uso de SQFD.

En la Figura 6-2 podemos ver que SQFD tiene un impacto positivo en cuatro factores: envolvimiento del usuario, soporte en administración y envolvimiento y técnicas para acortar el ciclo de vida del desarrollo de software. La importancia aquí radica que tres de estos factores fueron considerados los más importantes en el estudio de Necco.

	Factor (by rank)*	Mean SQFD Rating
1.	Improved user involvement	4.60
2.	Improved management support and involvement	4.40
3.	Better trained user and management personnel	3.20
4.	Technique to shorten SDLC	4.00
5.	Methods which integrate techniques and tools	2.80
6.	Better trained systems personnel	3.60
7.	Increased use of automated tools	2.80
8.	Improved project development technique	4.40
9.	Improved cost/benefit analysis techniques	3.80
10.	Improved computer hardware technology	3.60

Figura 6-2. Impacto de SQFD en algunos de los factores del desarrollo de SW [17]

En la Figura 6-3 podemos ver que el 51.4% de los proyectos fueron sistemas operativos/utilidad y tipos de software embebidos. Estos son los dos tipos de software cuyos requerimientos pueden ser determinados más fácilmente y definirlos cuantificablemente. Por lo que el uso de SQFD en este tipo de software es muy útil.

Software Type	Percentage
Operating system/utility	28.6
Embedded	22.8
Proprietary	22.8
Management information system	11.4
Decision support system	8.6
Network	2.9
Not applicable	2.9
Transaction processing system	0.0
Executive support system	0.0

Figura 6-3. Proyectos por tipo de Software [17]

En la Figura 6-4 podemos ver que el 82.9% de los proyectos estuvieron en la categoría de desarrollo de nuevas y actualizadas o mejoradas. Estos son lógicamente los tipos de desarrollo en el cual la recolección y transformación de los requerimientos de los clientes puede ser buena.

Development Type	Percentage
New	40.0
Upgrade	34.2
Proprietary	14.3
Enhancement	8.6
Not applicable	2.9
Correction of a deficiency	0.0
Standard maintenance	0.0

Figura 6-4. Beneficios de SQFD [17]

6.1 Proceso de SQFD

Como ya mencionamos SQFD es una técnica front-ent de elicitación de requerimientos que define y cuantifica los requerimientos críticos del cliente. Los pasos que SQFD sigue son los siguientes: Ver Figura 6-5

Paso 1: Los requerimientos del cliente son recolectados y registrados en la parte izquierda del eje y. Los requerimientos son usualmente sentencias cortas registradas en la terminología del cliente y son acompañadas por una definición detallada en los que en SQFD se llama un diccionario de datos.

Paso 2: En cooperación con los clientes, los requerimientos deben transformarse a sentencias técnicas y medibles y registrarse en la parte superior del eje x. Por ejemplo, “fácil de aprender” puede transformarse en “tiempo para completar un tutorial”, “número de iconos”, “Ayuda en línea”. En este punto es importante señalar que algunos requerimientos del cliente pueden convertirse en múltiples especificaciones técnicas.

Paso 3: Los clientes son cuestionados para completar la matriz de correlación identificando las fortalezas de la relación entre los requerimientos de varios clientes y las especificaciones técnicas. Por ejemplo, “fácil de aprender” esta fuertemente relacionado con “tiempo para completar el tutorial”, una correlación fuerte puede recibir un puntaje de 9 en la matriz de correlación.

Paso 4: En base a las encuestas al cliente, las prioridades de los requerimientos deben ser establecidas y son registradas del lado derecho del eje de las y.

Paso 5: Este proceso involucra el desarrollo de especificaciones técnicas del producto (bajo el eje x), sumando el resultado de multiplicar las prioridades por los valores de la correlación.

El resultado final del proceso de SQFD es un conjunto mínimo de especificaciones técnicas y medibles, su porcentaje de importancia y medida objetivo.

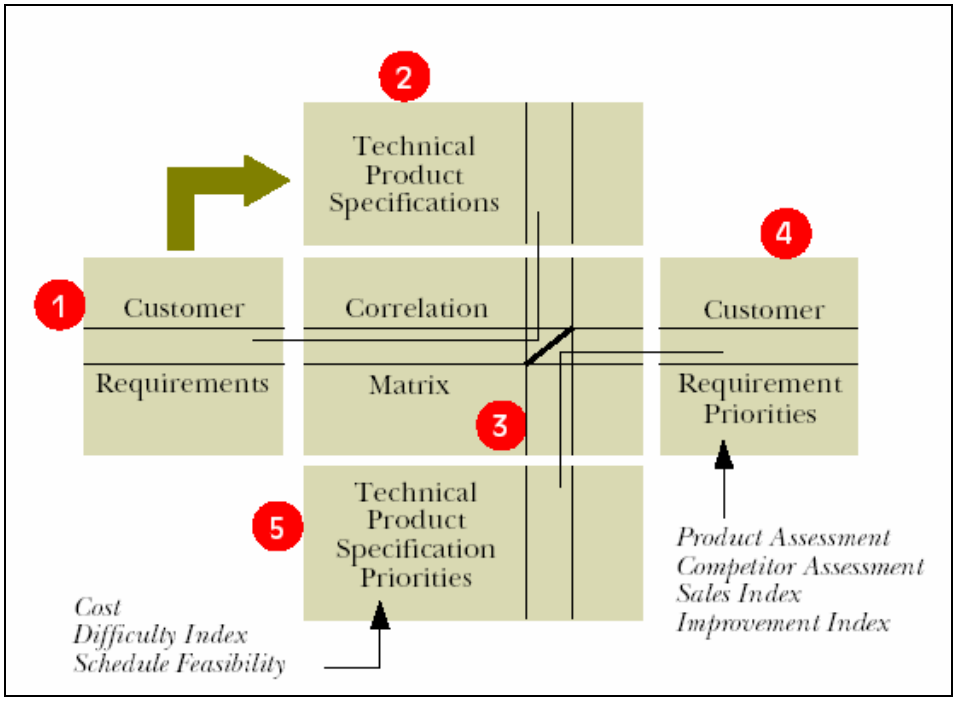

Figura 6-5. Modelo de SQFD

6.2 El futuro de SQFD

Los usos actuales de SQFD se mencionan a continuación:

- Mejorar en envolvimiento del cliente
- Mejorar el apoyo administrativo
- Recortar el tiempo de desarrollo de productos de software
- Mejorar el desarrollo de proyectos
- Es una metodología estructurada
- Soporta la integración del equipo
- Como herramienta preventiva para mejorar la calidad de los productos de software
- Evita la perdida de información

7 Glosario de términos

Término	Descripción
Checklist	Lista de verificación.
Escenario	En el contexto de este documento un escenario es una breve descripción de una simple interacción del usuario con el sistema. De esta manera es posible comprender las relaciones de esta interacción con los componentes de la arquitectura de software.
Sigma	Símbolo griego usado para representar la desviación estándar, o la cantidad de variación en un proceso.
Sistema legado	Sistemas que son heredados en las empresas, comúnmente sistemas viejos.
Trade-off	Para el contexto de este documento un Trade-off es un equilibrio entre varios atributos de calidad.

8 Acrónimos

Acrónimo	Descripción
ADL	*Architecture Description Language*.
CTC	*Critical-to-cost*
CTD	*Critical-to-delivery*
CTQ	*Critical-to-quality*
CTS	*Critical to Satisfaction*
DFSS	*Design for six sigma* (Diseño para seis sigma).
QFD	*Quality function deployment*
VOC	*Voice of Customer*
FMEA	*Failure mode-effect analysis*
DFMEA	*Design failure mode-effect análisis*
PFMEA	*Performance failure mode-effect analysis*
CAD	*Computer-aided design*
CAE	*Conputer-aided engineering*
SOW	*Statement of Work*

9 Referencias

[1] Len Bass, Paul Clements and Rick Kazman. *Software Architecture in Practice*. Edited by SEI Series In Software Engineering. second ed: Addison Wesley, 2003.
[2] Paul Clements, Rick Kazman and Mark Klein. *Evaluating Software Architectures, Methods and case studies*. Edited by SEI Series In Software Engineering. Addison Wesley, 2002.
[3] Mary Shaw and David Garlan. An introduction to Software Architecture. School of Computer Science Carnegie Mellon University Pittsburgh, PA, 1994.
[4] Tennant, Geoff. *Design for Six Sigma.* Edited by Gower House, England, 2002.
[5] Garlan David. Software Architecture: a Roadmap. School of Computer Science Carnegie Mellon University Pittsburgh, PA.
[6] Bosh Jan, Molin Peter. Software Architecture Design: Evaluation and Transformation. University of Karlskrona/Ronneby, Departement of Computer Science.
[7] Paul Clements, Rick Kazman and Mark Klein. *Evaluating Software Architectures*. Edited by The SEI Series in Software Engineering: Addison Wesley, 2002.
[8] Yang, Kai, El-Haik Basem. Design for Six Sigma: A Roadmap for Product Development. Edited by McGraw-Hill, 2003.
[9] Aldrich, Jonathan, Chambers, Craig, Notkin, David. "Archjava: Connecting Software Architecture to Implementation." In: Department of Computer Science and Engineering, University of Washington, 2002.
[10] Shaw, Mary, Garlan, David. "Formulations and Formalism in Software Architecture." In : Carnegie Mellon University, 1995.
[11] Antony, Jiju, and Banuelos Ricardo Coronado. "Design for Six Sigma." Manufacturing Engineer 2002, 24-26.
[12] Bass, Len, Kazman, Rick. "Architecture-Based Development." In : Carnegie Mellon Software Engineering Institute, 1999.
[13] Consulting, Bredemeyer. Software Architecting, 2006. Available from http://www.bredemeyer.com.
[14] Hernández, Juárez José de Jesús. "Modelo Para El Desarrollo De Arquitecturas De Software." Reporte Técnico de Investigación, Centro de Investigación en Matemáticas Aplicadas A.C., 2005.
[15] Eriksson, Hans-Erik, Penker, Magnus. Business Modeling With Uml, Business Patterns at Work: Wiley Computer Publishing, 2000.
[16] Guía De Los Fundamentos De La Dirección De Proyectos. . Edited by Norma NacionalAmericanaANSI/PMI 99/001/2004. Tercera Edición ed: Project Management Institute, 2004.
[17] Haag, Stephen, Raja, M.K., Schkade, L.LHernández. "Quality Function Deployment Usage in Software Development.".Communications of the ACM., 1996.
[19] Bachmann, Felix, Bass, Len, Chastek, Gary. "The Architecture Based Design Method." In : Technical Report CMU/SEI-2000-TR-001, 2000.
[20] Bachmann, F, Bass, L, Carriere, J Clements, P, Garlan, D; Ivers, J, Little, R.; & Nord, R, "Software Documenting Software Architectures: Organization of Documentation Package." In: (CMU/SEI-2001-TN-010).
[21] Kazman, Rick, Carriére, Jeromy, Woods, Steven, "Toward a Discipline of Scenario-based Architectural Engineering" In: Carnegie Mellon University Pittsburgh.