CENTRO DE INVESTIGACION EN MATEMÁTICAS, A.C.

Modelo para el Desarrollo de Arquitecturas de Software

Reporte Técnico de Investigación
que para obtener el grado de

Maestro en Ingeniería de Software

presenta

José de Jesús Hernández Suárez

Director:

Dr. Carlos Montes de Oca Vázquez

Guanajuato, Gto. a 22 de julio de 2005.

Resumen

El desarrollar sistemas de software cada vez más grandes y complejos para cubrir las necesidades y objetivos de las organizaciones actuales, exige prestar atención y especial cuidado al diseño de la arquitectura de dichos sistemas. Se sabe que el proceso para elaborar una arquitectura de software es muy crítico e importante para el desarrollo de un sistema de software de gran escala; requiere de conocimientos, experiencia y creatividad por parte del arquitecto. Son pocas las propuestas de proceso para elaborar una arquitectura de software que existen en la literatura, en su mayoría dejan en un alto nivel los pasos que se deben seguir, y en otros casos sólo atacan una parte del proceso. Por otra parte, las organizaciones y arquitectos de software requieren un proceso que les muestre a detalle las actividades que tienen que seguir, así como los recursos en los que se pueden apoyar para elaborar una arquitectura de software. El presente trabajo se encuentra orientado a la atención de estos problemas, proponiendo un modelo de proceso para el desarrollo de arquitecturas de software, el cual contempla la definición de pasos a cubrir, el flujo de actividades a detalle que se deben seguir, la especificación entradas y salidas para cada una de dichas actividades, la descripción de las herramientas susceptibles de utilizar para apoyar las mismas, la descripción de los roles que intervienen, y la definición de las métricas que se deben registrar para controlar y mejorar el proceso.

Agradecimientos

El presente trabajo representa el fruto del apoyo, aliento, confianza, guía, enseñanzas y aportaciones que recibí de muchas personas. Deseo expresar mi más reconocida gratitud a todas ellas.

Al Dr. Carlos Montes de Oca, mi director de investigación, por sus excelentes y puntuales observaciones, recomendaciones, enseñanzas y guía que orientaron mi camino para llevar a cabo este esfuerzo, así como a lo largo de la maestría.

Al Dr. Cuauhtémoc Lemus Olalde, por todos los conocimientos transmitidos a través de sus enseñanzas, ejemplos, y observaciones como experto en arquitecturas de software, así como todo su apoyo.

Al Dr. Miguél Ángel Serrano, por sus diversas aportaciones que reforzaron mi enseñanza en diferentes temáticas a lo largo de la maestría.

Al Consejo de Ciencia y Tecnología del Estado de Guanajuato (CONCYTEG), por su parcial financiamiento a través del proyecto 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 de alta calidad académica.

A mis amigos y compañeros de generación de la maestría, Karen Cortés, Omar Gómez, Gustavo Herrera, Juan Antonio López y Nayhely Trujillo, por su valiosísimas experiencias transmitidas, por su incondicional y sincero apoyo en todo momento, y por su grandísima paciencia que siempre me tuvieron para avanzar en nuestro camino.

A mi alma mater, la Universidad Nacional Autónoma de México, y especialmente a la Dirección General de Servicios de Cómputo Académico (DGSCA), por forjar mi carrera, conocimiento, habilidades y experiencia, que me han ayudado a lo largo de mi vida académica y profesional, y que sin duda alguna me ayudaron a concluir mi maestría por todo lo que siempre me transmitió.

Al Dr. Juan Voutssás Márquez, a la Mtra. Marcela Peñaloza Báez, al Mtro. Hugo Alonso Reyes Herrera, al Lic. José Luis Medina Flores, y a todos mi excelentes amigos y compañeros de la Dirección de Sistemas de la DGSCA que me dieron su invaluable apoyo, inspiración y aliento para estudiar esta maestría, así como sus excelentes experiencias, enseñanzas y ejemplos transmitidos que han ayudado a mi formación.

Y especialmente a mi gran familia, mis padres José de Jesús y Sofía, mis hermanos, Olivia, Sofía, Verónica, Luis Adrián y Alejandra, por su incondicional apoyo, cariño, impulso, aliento y confianza que siempre he recibido de ellos y me ha ayudado a ser quien soy.

Atentamente, José de J.

Tabla de Contenido

Lista de Figuras

Lista de Tablas

MODAS
Modelo para el desarrollo de arquitecturas de software

1. Introducción

Las necesidades actuales que tiene toda organización para el logro de sus objetivos, demandan la construcción de grandes y complejos sistemas de software, mismos que requieren de la combinación de diferentes tecnologías y plataformas de hardware para alcanzar un buen funcionamiento acorde a dichas necesidades. Esto les sugiere a los ingenieros en software poner especial atención y cuidado en su proceso de diseño de la arquitectura de software bajo la cual está soportado el funcionamiento de dichos sistemas de software. Si una arquitectura de software se encuentra deficiente en su concepto o diseño, o en el peor de lo casos, no contamos con la arquitectura del sistema que nos encontremos desarrollando, tendremos grandes posibilidades de estar construyendo un sistema incorrecto que no alcanzará el total de los objetivos planteados. Esto indudablemente nos podrá llevar a un re-trabajo de mayor inversión tiempo que el planeado, o peor aun, nos podrá llevar al fracaso del sistema de software cuando se encuentre en operación.

En el diseño de un sistema, existen dos fases generales que se pueden distinguir: 1) diseño arquitectónico y 2) diseño detallado. El diseño arquitectónico ocurre inmediatamente después de la especificación de requerimientos de software y antes del diseño detallado. Un diseño arquitectónico se encuentra enfocado a los elementos estructurales, funcionales y de comportamiento de un sistema, tratando de modelar el problema y la propuesta de solución. Los elementos estructurales que influencian las decisiones de diseño incluye: estructura de organización y control; comunicación, sincronización, y protocolos de acceso a datos; distribución física de funciones de elementos de diseño; desarrollo futuro, etc. Por su parte, el diseño detallado describe el control jerárquico, estructura de datos, lógica y empaquetado de elementos establecidos en el diseño estructural (traducido y modificado de) [1].

Bass, Clements y Kazman definen a la arquitectura de software de un programa o sistema de cómputo, como la estructura o estructura del sistema, la cual consta de componentes de software, propiedades visibles externamente de esos componentes y la relación a través de ellos (traducido de) [2].

De acuerdo al concepto de arquitectura de software, podemos decir que el proceso de elaboración de una arquitectura de software se refiere al conjunto de pasos que tenemos que seguir para conceptualizar, diseñar, definir, verificar, documentar y evaluar los diferentes componentes de software y sus propiedades bajo los cuales estará soportado un sistema de cómputo, así como la relación que existirá entre los mismos.

El proceso de elaboración de arquitecturas de software representa uno de las procesos más importantes y a la vez más críticos del ciclo de vida del desarrollo de sistemas, debido a que en esta etapa es dónde se define y especifica el diseño de alto nivel del sistema, el cual fundamenta los componentes, estructura, organización y relaciones entre dichos componentes, mismos que permitirán llevar a cabo las funciones del sistema de acuerdo a los requerimientos e intereses de los usuarios. Para los ingenieros de software, un proceso definido para la elaboración de arquitecturas de software representa contar con una comunicación mutua con los diferentes *stakeholders*, realizar decisiones tempranas de diseño, evaluación e identificación temprana de restricciones de diseño, identificar atributos de calidad en conflicto o *tradeoffs* y reutilizar componentes de software, además de evitar errores, encontrar fallas y reducir costos a consecuencia de re-trabajo.

Actualmente, importantes autores y expertos en la materia de arquitecturas de software, como Bass, Clements, Kazman, Nord, Kruchten, Garlan y Bachmann, entre otros, han realizado muy buenas aportaciones y recomendaciones en la materia, lo cual ha permitido adoptar y adaptar modelos para llevar a cabo un proceso de elaboración de arquitecturas de software, o bien un parte de ella.

El problema radica en que existe la ausencia de la definición concreta de un proceso para la elaboración de una arquitectura de software que permita identificar claramente, las actividades a llevar a cabo en cada uno de los pasos propuestos, la secuencia concreta de las mismas, las entradas y salidas del proceso, así como de cada una de las actividades que presentan los pasos del mismo, y las herramientas susceptibles de utilizar para apoyar dichas actividades.

Por otra parte, sabemos que existe una empresa mexicana dedicada a la consultoría de software cuyo negocio base es el desarrollo de software a la medida, que preocupada por una constante mejora de procesos y control estadístico de sus operaciones, busca perfeccionar su proceso de elaboración de arquitecturas de software. Actualmente esta empresa cuenta con un proceso definido para esta fase del ciclo de vida de un sistema, sin embargo, se puntualiza que dicho proceso sirve como una guía, es decir, es opcional para los ingenieros de software al llevar a cabo la labor del diseño de una arquitectura, ya que no restringe el proceso utilizado en el diseño de la misma, debido a que es la parte más creativa del proceso de desarrollo. De esta manera, la empresa en ningún momento restringe las técnicas o metodologías utilizadas para representar o definir aspectos de diseño de la arquitectura, ni restringe las secciones a cubrir en su documentación, sin embargo, para arquitectos novatos o con poca experiencia, esta actividad puede ser complicada y muy laboriosa sino se cuenta con un proceso que detalle de manera clara las actividades que puede seguir, las entradas y salidas de las mismas, así como la descripción de los recursos o herramientas en los que se puede apoyar para llevar a cabo dichas actividades .

Considerando que: 1) el proceso de elaboración de una arquitectura de software es de suma importancia y crítico para el desarrollo de un sistema de software de gran escala; 2) no existe en la literatura un modelo que permita identificar de manera clara y rápida las actividades a detalle que se deben seguir para la construcción de una arquitectura de software con la secuencia de las mismas, y la descripción de recursos o herramientas que apoyen cada una de dichas actividades; y que 3) el proceso definido por la empresa que hemos mencionado como ejemplo, aun con la correcta definición y control estadístico de otros procesos para el ciclo de vida de un sistema, el que se relaciona con el desarrollo de una arquitectura de software no es muy detallado y se encuentra débil en algunos aspectos; proponemos como solución diseñar un modelo de proceso para el desarrollo de arquitecturas de software que permita: a) identificar claramente las actividades a llevar a cabo en cada uno de los pasos propuestos, b) conocer detalladamente las entradas y salidas del proceso, así como de cada una de las actividades que presentan los pasos del mismo, c) conocer las herramientas o recursos que son posibles utilizar para apoyar dichas actividades, d) identificar los roles que deben de intervenir, y e) registrar métricas dentro de las actividades a llevar a cabo que permitan controlar y mejorar el proceso. De esta manera, el presente reporte de investigación presenta una propuesta de proceso para elaboración de arquitecturas de software definido, que trata de contemplar todos los elementos expuestos hasta el momento. La propuesta se encuentra basada principalmente en varios de los pasos y elementos que toman en cuenta el modelo de Desarrollo Basado en Arquitecturas de Len Bass y Rick Kazman [3].

El presente documento de reporte técnico se encuentra dividido en diez secciones. La primera sección esta representada por la presente introducción; la segunda sección describe la problemática a la que nos enfrentamos; la tercera sección muestra el trabajo previo realizado en la materia (estado del arte); la cuarta sección describe el nuestra propuesta de solución; la quinta sección describe el modelo para el desarrollo de arquitecturas de software propuesto; la sexta sección discute las desventajas, limitaciones, ventajas, recomendaciones, alcance y mejoras que hemos identificado para el modelo para el desarrollo de arquitecturas de software propuesto; la séptima sección da las conclusiones de la propuesta; la octava sección muestra las referencias analizadas para la elaboración del presente reporte; la novena sección presenta un glosario de términos utilizados a lo largo del presente documento, y finalmente, la décima sección presenta bibliografía adicional que apoyó la investigación.

2. Descripción del problema

Dentro del campo de las arquitecturas de software existen diferentes e importantes aportaciones que han realizado diversos expertos en la materia, como son: Bass, Clements, Kazman, Nord, Kruchten, Garlan y Bachmann, entre otros. Estos autores han propuesto distintos conceptos, modelos, técnicas y herramientas para llevar a cabo una arquitectura de software, o bien una parte de la misma, sin embargo son muy pocos los que consideran el proceso de manera completa, y aun un número mucho menor los que especifican el detalle de actividades a seguir en el proceso. Las organizaciones preocupadas por esta importante fase del ciclo de vida de desarrollo de sistemas, han tomando estas propuestas disponibles para esta fase del ciclo de vida de desarrollo de sistemas, y en el mejor de los casos las han adaptado para contar con un proceso mucho más definido para la creación de arquitecturas de software de acuerdo a las exigencias de sus clientes, sin embargo, en muchos de los casos continúa siendo deficiente.

Actualmente existe una compañía mexicana de consultaría y desarrollo de software a la medida, quién preocupada por una constante mejora de procesos, y control y calidad de sus operaciones, cuenta con procesos bien definidos para todas las etapas del ciclo de vida de un sistema de software, incluyendo la elaboración de la arquitectura de software, sin embargo, este último proceso no cuenta con un nivel de detalle que muestre explícitamente a un ingeniero en software las actividades y herramientas que debe hacer uso para llegar al diseño de la arquitectura de una manera sencilla. Esta importante fase no restringe a los ingenieros en software el proceso utilizado, les proporciona una guía debido a que es la parte más creativa del proceso de desarrollo, y por tal situación, se ha detectado que existen debilidades que impiden cubrir varios de los aspectos y recomendaciones que hacen los expertos para la creación de una arquitectura de software.

Sabemos que es muy importante y crítico el proceso de desarrollo de una arquitectura de software, debido a que el resultado de su ejecución nos conducirá al correcto, o bien, débil diseño de un sistema de software, lo cual puede traer como consecuencia re-trabajo en la fase de codificación del sistema, o peor aún en la fase de pruebas. Tomando esto en consideración, comprobando que los procesos o modelos actualmente propuestos por los expertos en arquitecturas de software se encuentran a un alto nivel, y que la industria de software no cuenta con una definición detallada de este proceso, podemos decir que la problemática común a la que se enfrentan los ingenieros de software en la fase de diseño de sistemas es que no cuentan con un recurso accesible que les muestre de manera clara las actividades a detalle de un proceso para elaboración de arquitecturas de software, el flujo que deben de seguir para llevar a cabo dichas actividades, definición clara de entradas y salidas respectivas, los roles quiénes desempeñan dichas actividades, y los recursos o herramientas que apoyen las actividades, además de métricas que permitan controlar y mejorar dicho proceso. Los modelos y propuestas de proceso actuales que recomiendan los expertos, no consideran en conjunto todos estos elementos que son de suma importancia para el diseño que llevan a cabo los ingenieros en software. Esto nos representa la posibilidad de proponer un proceso definido para elaboración de arquitecturas de software.

El contar con un proceso definido de elaboración de arquitecturas de software trae como consecuencia varios beneficios para los ingenieros de software, tales como mejorar la comunicación mutua entre los diferentes *stakeholders*, realizar decisiones tempranas de diseño, evaluar e identificar tempranamente restricciones de diseño, identificar atributos de calidad en conflicto o *tradeoffs* y reutilizar componentes de software, además de evitar errores, encontrar fallas y reducir costos a consecuencia de re-trabajo.

Por el contrario, si no atendemos la relevancia que tiene un proceso para elaboración de arquitecturas de software, nos encontramos en el riesgo de desarrollar software deficiente en su concepto o diseño, y por ende estaremos construyendo un sistema incorrecto que no alcanzara el total de los objetivos planteados. Esto nos llevará a un re-trabajo con mayor inversión de tiempo, esfuerzo y dinero del planeado, o peor aun, nos podrá llevar al fracaso del sistema de software cuando se encuentre en operación.

Este trabajo se encuentre dirigido a la atención de los problemas descritos anteriormente, para lo cual definimos un modelo para la elaboración de arquitectura de software que busca satisfacer la carencia de un proceso que contemple el flujo de actividades, entradas, salidas, roles, herramientas y métricas que permitan desarrollarla.

3. Trabajo previo

Dentro del campo de la arquitectura de software, existen aportaciones importantes para los diferentes aspectos que deben ser considerados en su desarrollo. Distintos autores han realizado propuestas muy interesantes en relación a métodos o procesos para llevar a cabo una arquitectura de software, o bien alguna parte de ella. En la presente sección describimos brevemente algunos conceptos básicos y algunas de estas aportaciones, mismas que hemos estudiado y analizado como parte de la investigación del tema y en las cuales se basa nuestra propuesta de solución.

La presente sección se encuentra dividida en cinco puntos principales. El primer punto describe algunos conceptos básicos de arquitecturas de software; el segundo punto menciona algunas propuestas de proceso para arquitecturas de software; el tercer punto describe técnicas para el diseño de arquitecturas de software; el cuarto punto menciona algunos estándares relacionados con arquitecturas de software, y finalmente el quinto punto hace mención de algunas herramientas adicionales que serán de utilidad para el proceso de desarrollo de arquitectura de software.

3.1. Conceptos básicos

Para comprender con mayor facilidad los elementos que son considerados en una arquitectura de software, mencionaremos a continuación algunos conceptos importantes, los cuales son: estilo arquitectónico, patrón arquitectónico, patrón de diseño, vistas y estructuras arquitectónicas, *view point*, escenarios y atributos de calidad.

Un estilo arquitectónico se define como un vocabulario de componentes y tipos de componentes, así como un conjunto de restricciones sobre cómo estos componentes pueden ser combinados. Algunos de estos estilos arquitectónicos son: procesamiento secuencial, tuberías y filtros, sistemas orientados a objetos, sistemas en capas, sistemas basados en eventos, bases de datos, entre otros (traducido de) [4].

Un patrón arquitectónico generalmente impone una regla (Perry and Wolf, 1992) en la arquitectura, la cual especifica cómo manejará el sistema un aspecto de su funcionalidad. Un patrón arquitectónico trata con algún aspecto del comportamiento del sistema que no está en el dominio de la funcionalidad del mismo, pero el cual maneja alguno de sus dominios de soporte, por ejemplo, la manera en cómo el sistema maneja la concurrencia, la persistencia, la tolerancia a fallas o la distribución (traducido de) [5]. Un patrón arquitectónico es diferente a un estilo arquitectónico en el sentido que no es predominante y puede ser combinado con un estilo arquitectónico sin problemas. También es diferente a un patrón de diseño, toda vez que afecta completamente la arquitectura, o al menos un parte importante de ella (traducido de) [5].

Un patrón de diseño describe una estructura comúnmente recurrente de comunicación entre componentes que resuelve un problema de diseño general dentro de un contexto en particular (traducido de) [6].

Un estilo arquitectónico está compuesto de patrones arquitectónicos y un patrón arquitectónico está compuesto por patrones de diseño. Un mecanismo arquitectónico es una estructura en la cual los objetos colaboran para obtener cierto conocimiento que ayuda a lograr metas de calidad específicas (traducido de) [7].

Una vista es una representación de un conjunto coherente de elementos arquitectónicos. Las vistas consisten de una representación de un conjunto de elementos y relaciones entre ellos; muestran una perspectiva de la arquitectura de software. Bass Clements y Kazman proponen tres grupos de vistas, los cuales son: estructuras de módulos, estructuras de conectores y componentes, estructuras de asignación [2].

Un *viewpoint* es una especificación de las convenciones para construir y utilizar una vista; representa un patrón o plantilla a partir del cual se desarrollan las vistas individuales de la arquitectura mediante el establecimiento de los intereses de los diferentes *stakeholders* para dicha vista y las técnicas para su creación y análisis (traducido y modificado de) [8]. Identifica el conjunto de intereses a cubrir e identifica las técnicas de modelado, de evaluación o de verificación de consistencia empleadas por cualquier vista que corresponda.

Un escenario es una breve descripción de una interacción simple de un *stakeholder* con el sistema. Los casos de uso se enfocan al comportamiento del sistema en tiempo de ejecución, dónde el *stakeholder* es el usuario, sin embargo, los escenarios abarcan otras interacciones con el sistema, tales como el mantenimiento que se le debe dar a un sistema al emerger una modificación. De esta manera, los escenarios tienen otros usos más allá de la obtención y validación de requerimientos de software, como ayudar a comprender y aceptar lo que requiere el *stakeholder*, y obtener diferentes vistas arquitectónicas (traducido y modificado de) [9].

Un atributo de calidad es un requerimiento no funcional de un sistema de software por ejemplo: confiabilidad, facilidad de modificación, rendimiento, facilidad de uso, entre otros. De acuerdo al estándar IEEE Standard 1061-1992 [38], un atributo de calidad representa el grado en el cual un software posee una combinación deseable de estos atributos. Jan Bosch [10] define dos categorías para agruparlos, las cuales son: atributos de desarrollo y atributos de operación. Los atributos de desarrollo son relevantes desde una perspectiva de ingeniería de software, estos atributos pueden ser: facilidad de mantenimiento, reutilizable, flexibilidad, facilidad de modificación. Los atributos operacionales son características del sistema en operación por ejemplo: el rendimiento del sistema, la confiabilidad, robustez, tolerancia a fallas, seguridad. Cualquier arquitectura de software definida de manera apropiada promueve que se satisfagan atributos de calidad.

3.2. Procesos de arquitectura de software

En la presente sección describiremos brevemente algunos procesos de arquitecturas de software que fueron analizados para generar nuestra propuesta de solución. Estos procesos son: Desarrollo Basado en Arquitecturas y Diseño de Arquitectura de Software: Evaluación y Transformación.

Desarrollo Basado en Arquitecturas

Partiendo de la idea que el desarrollo de una arquitectura de software es un paso crítico en el desarrollo de sistemas de software de gran escala, Len Bass y Rick Kazman en 1999 propusieron un proceso para desarrollar arquitecturas de software, conocido como “Desarrollo Basado en Arquitecturas” *(Architecture-Based Development*) [3], el cual facilita la identificación de tareas que son necesarias para generar una arquitectura de software, así como el seguimiento de un método para su construcción. El proceso incluye los siguientes pasos:

a) Obtener los requerimientos arquitectónicos, el cual genera un listado de requerimientos funcionales, un listado de requerimientos arquitectónicos específicos y un listado de escenarios de calidad.

b) Diseñar la arquitectura, el cual estructura las vistas: funcional, de código, de concurrencia, física y de desarrollo; desarrolla subsistemas candidatos y selecciona subsistemas.

c) Documentar la arquitectura, en la cual muestra recomendaciones para realizar la documentación de dicha arquitectura.

d) Analizar la arquitectura, en dónde evalúa la arquitectura mediante el método ATAM (Architecture Tradeoff Analysis Method) [11].

e) Generar la arquitectura, en dónde muestra recomendaciones para comenzar a codificar la arquitectura de software.

f) Mantener la arquitectura, el cual propone volver a ejecutar el método para incorporar nuevos requerimientos a la arquitectura de software.

De manera gráfica, el proceso es representado como se muestra en la figura 3.2.1. Cada uno de estos pasos considera los siguientes elementos:

- Entradas, incluyendo la recolección de la información principal
- Actividades constructivas

- Actividades de validación
- Salidas

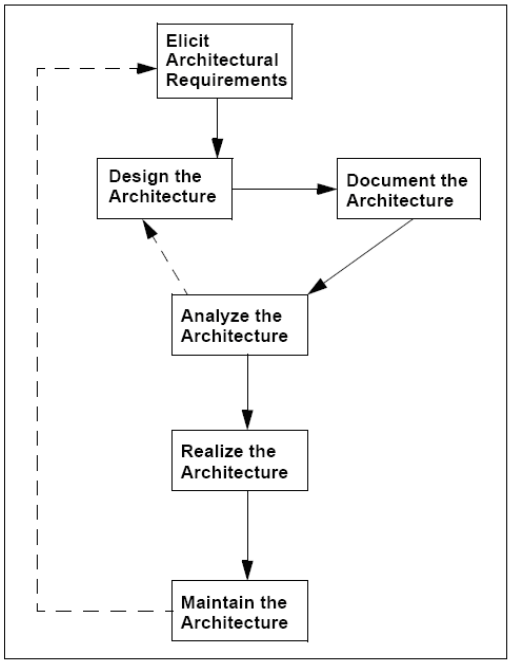


Figura 3.2.1, Desarrollo basado en arquitecturas.

Diseño de Arquitectura de Software: Evaluación y Transformación

Bosh y Molin presentan en su artículo "*Software Architecture Desing: Evaluation and Transformation*" [10] un método de diseño de arquitecturas que provee la base para un proceso de diseño balanceado y optimizado, especialmente para los requerimientos no funcionales. El método evalúa de manera iterativa el grado en el cual la arquitectura soporta cada uno de los requerimientos no funcionales y mejora la arquitectura utilizando la transformación hasta que todos los requerimientos no funcionales están cubiertos.

Para este método, la especificación de requerimientos es considerada como entrada y el diseño de la arquitectura generada es la salida. El proceso se puede resumir de la siguiente manera y es ilustrado con la figura 3.2.2:

1. Se realiza un primer diseño arquitectónico basado en la especificación de los requerimientos funcionales.
2. Este diseño es evaluado con respecto a los requerimientos no funcionales.
3. A cada requerimiento no funcional le es asignada una estimación utilizando una técnica de evaluación cualitativa o cuantitativa.
4. Los valores estimados de los requerimientos no funcionales son comparados con los valores en la especificación del requerimiento.
5. Si todas las estimaciones están cubiertas o mejoradas, el proceso de diseño de la arquitectura es finalizado. De otra manera la segunda fase se ejecutada: transformación de la arquitectura.
6. Durante esta etapa, la arquitectura es mejorada seleccionando el requerimiento no funcional apropiado optimizando su transformación.
7. Cada uno de los conjuntos de resultados de la transformación es una nueva versión del diseño arquitectónico.

8. El diseño es nuevamente evaluado y el mismo proceso es repetido de ser necesario, hasta que todos los requerimientos no funcionales son completados o hasta que el ingeniero en software decida que no es factible la solución existente.

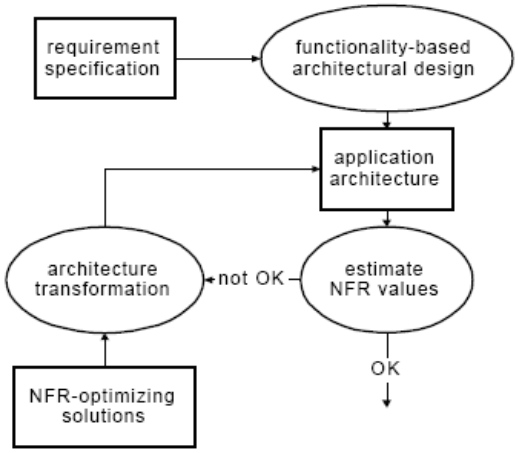


Figura 3.2.2, Esquema de un método de diseño de arquitectura.

3.3. Diseño de arquitecturas

En esta sección describiremos brevemente algunos modelos y técnicas que son utilizadas para diseñar y documentar una arquitectura de software. Dichos modelos son: Diseño Basado en Arquitecturas, Arquitectura Basada en Escenarios, vistas arquitectónicas, Diseño de arquitectura de software efectivo con UML.

Diseño Basado en Arquitecturas

Bachmann, Bass, Chastek, Donohoe y Peruzzi en el año 2000 propusieron un proceso para llevar a cabo el diseño de una arquitectura de software, el cual es conocido como "Método de Diseño Basado en Arquitectura" (*The Architecture Based Design Method*) [12]. El método proporciona una serie de pasos para el diseño conceptual de la arquitectura de software. Satisface los requerimientos funcionales, de negocio y de calidad al nivel de abstracciones que permiten realizar variaciones en la arquitectura cuando se produce una arquitectura en específico. Se basa en la determinación de los elementos que guía o conducen a la arquitectura (*architectural drivers*), lo cual es la combinación de dichos requerimientos funcionales, de negocio y de calidad que moldea la arquitectura.

El método tiene tres fundamentos: descomposición de función (acoplamiento y cohesión), compresión de los requerimientos de calidad y de negocio, a través de la selección de estilos arquitectónicos y el uso de patrones de software. La descomposición es examinada desde la perspectiva de tres vistas: lógica, concurrencia y *deployment*. Se utiliza la vista lógica para registrar las responsabilidades e interfases conceptuales (salida y entrada de datos) para el diseño de elementos; la vista de concurrencia para examinar el sistema desde la perspectiva de múltiples usuarios, recursos, puesta en marcha y otras actividades paralelas, y la vista de *deployment* para representar nodos en una red de computadoras, esto es, la estructura física del sistema.

La entrada del método de diseño basado en arquitectura consiste de:

- Requerimientos funcionales abstractos
- Casos de uso

- Requerimientos de negocio y de calidad abstractos
- Escenarios de calidad
- Opciones de arquitectura
- Reglas

El proceso se fundamenta en la descomposición del sistema en subsistemas conceptuales, éstos a su vez en componentes conceptuales y finalmente éstos últimos en componentes concretos. Este proceso se esquematiza con la figura 3.3.1.

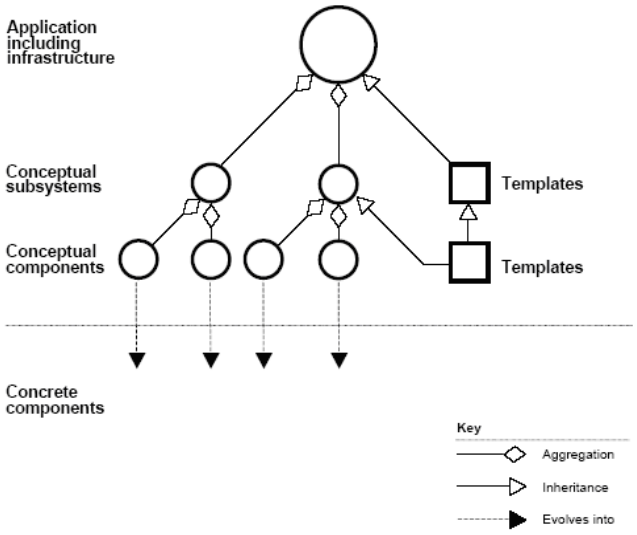


Figura 3.3.1, Descomposición del sistema en elementos de diseño, Método de Diseño Basado en Arquitectura.

Los pasos que considera el método se pueden resumir en los siguientes:

1. Descomposición del sistema en elementos de diseño.
2. Definición de grupos de funcionalidad.
3. Selección del estilo arquitectónico.
4. Refinar plantillas.
5. Verificación de la funcionalidad.
6. Generación de vistas arquitectónicas.
7. Verificación de escenarios y reglas.

Arquitectura Basada en Escenarios

Kazman, Carrière y Woods proponen un proceso que permite relacionar escenarios con los atributos de calidad que se consideran en una arquitectura de software, así como con los *stakeholders* interesados en los mismos [9]. El proceso propone obtener escenarios, corresponder escenarios a las vistas arquitectónicas, y construir modelos analíticos, los cuales ayudan a comprender y documentar las decisiones estructurales que se han hecho sobre la arquitectura. Los escenarios que se manejan en el proceso pueden ser:

- Directos, los cuales representan los usos del sistema construido de la arquitectura de software (estímulo-respuesta), o bien

- Indirectos, quienes representan cambios a la arquitectura existente, y por ende al sistema; incluyen nuevas plataformas de hardware o software, agregar una nueva característica, integrarse con una nueva pieza de software, reemplazar alguna funcionalidad existente, escalar el sistema, etc.

Los autores determinan que cada uno de los intereses de los *stakeholders* debe ser reflejado en los escenarios. Múltiples *stakeholders* pueden estar contemplados en múltiples escenarios de acuerdo a su interés. Para manejar esto, proponen una matriz de escenarios que ayuda a considerar a cada escenario desde cada una de las perspectivas de los *stakeholders* y desde la perspectiva de cada atributo de calidad o interés. Esta matriz se esquematiza por medio de la figura 3.3.2.

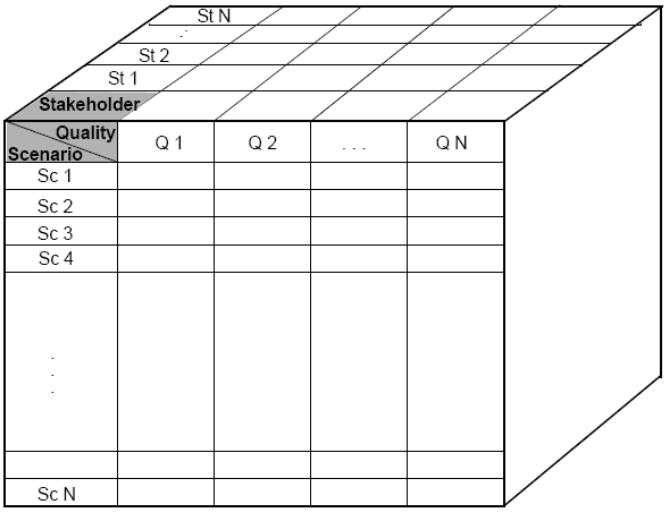


Figura 3.3.2, Matriz de obtención de escenarios genérica.

Los escenarios obtenidos deben corresponder a la representación de la arquitectura de software. Para realizar dicha correspondencia, se debe obtener:

1. Atributos de calidad
2. Escenarios abstractos
3. Escenarios derivados / específicos
4. Mecanismo genérico para responder al escenario
5. Mecanismo específico
6. Modelo analítico, los cuales representan el principio de la ingeniería arquitectónica, pues representan de manera formal cómo serán medidos los atributos de calidad.

Vistas arquitectónicas

Cuando un arquitectura de software se encuentra representada por uno o más estilos arquitectónicos, patrones arquitectónico o por patrones de diseño, es necesario conocer la manera en cómo interactúan los componentes definidos unos con otros, cómo están interconectados dichos componentes entre sí, el tipo de plataforma de software y de hardware dónde se ejecutarán los componentes, entre otros aspectos. En 1995, Kruchten y por su parte Nord, [13] [14] dieron a conocer dos propuestas de vistas arquitectónicas para representar diferentes aspectos de una arquitectura de software.

Kruchten [13] propone cinco vistas:

- Vista lógica. Describe el diseño de los objetos que son modelados cuando es utilizado un método de diseño orientado a objetos. En esta vista también se puede modelar diagramas de entidad relación.
- Vista de proceso. Describe el diseño de los aspectos de concurrencia y sincronización del sistema.
- Vista física. Describe la relación de los componentes de software que serán ejecutados en el hardware y refleja aspectos de distribución.
- Vista de desarrollo. Describe la organización estática de los componentes de software en un ambiente de desarrollo.
- Escenarios. Muestran cómo los componentes de las cuatro vistas anteriores trabajan en conjunto de manera transparente.

Por su parte, Nord [14] propone cuatro vistas, mismas que definiremos en la siguiente técnica.

En la actualidad no existe un estándar que defina las vistas que debe tener una arquitectura de software, cada arquitecto de software puede utilizar y/o crear sus propias vistas, las cuales sin lugar a dudas considerarán estos elementos, sin embargo es posible tomar como referencia las vistas antes mencionadas.

Diseño de Arquitectura de Software Efectivo con UML

Nord, Paulish, Soni y Hofmeister mencionan en su artículo "*Effective Software Architecture Design: From Global Analysis to Uml Descriptions*" [15] que hoy en día es generalmente aceptado que el separar la arquitectura de software en múltiples vistas puede ayudar a reducir la complejidad y la toma de decisiones sobre diseño de *trade-offs*. Proponen la realización de vistas arquitectónicas por medio de UML[1] para analizar los factores que influencian la arquitectura y desarrollar estrategias para adaptar esos factores en el diseño de la arquitectura. En el artículo, los autores presentan los diagramas UML que pueden ser elaborados para representar cada una de las vistas arquitectónicas. Las vistas que consideran para la representación de la arquitectura son [16]:

- Vista conceptual. Describe la arquitectura en términos de elementos del dominio. El arquitecto diseño las características funcionales del sistema.
- Vista de módulo. Describe la descomposición del software y su organización en capas. Una consideración importante aquí es limitar el impacto de un cambio en software y hardware.
- Vista de ejecución. Es la vista de tiempo de ejecución del sistema: es el mapeo de los módulos a entidades en tiempo de ejecución, definiendo la comunicación entre ellos y asignándoles los recursos físicos. El uso de recursos y desempeño son los principales intereses (*concerns*) en la vista de ejecución. Decisiones como el uso de una librería ligada o compartida, o el uso de hilos o procesos, se hacen aquí, aunque esas decisiones pueden retroalimentar a la vista de módulos y requerir cambios.
- Vista de código. Captura cómo los módulos e interfases en la vista de módulos son correspondidas hacia los archivos fuente, y cómo las entidades en tiempo de ejecución en la vista de ejecución son correspondidas a los archivos ejecutables. La partición de esos archivos y cómo están organizados en los directorios afecta la construcción del sistema.

[1] Unified Modeling Language, lenguaje para especificar, visualizar, modelar, construir y documentar artefactos de sistemas de software.

3.4. Estándares

A continuación describiremos brevemente dos importantes estándares, el modelo ISO 9126-1 y el estándar IEEE Std 1471-2000 que son importantes considerar para el diseño y documentación de una arquitectura de software.

Modelo ISO 9126-1

Existe un modelo de la ISO (International Organization for Standardization) que permite identificar las atributos de calidad que pueden caracterizar a una arquitectura de software [7]. El modelo es conocido como modelo de calidad ISO 9126-1. Indica que la calidad está definida por el conjunto de características del producto que reflejan su capacidad para satisfacer su objetivo o cubrir necesidades. Clasifica y define los diferentes atributos de calidad que pueden ser considerados en un sistema de software. Menciona que diferentes atributos de calidad pueden ser considerados por diferentes *stakeholders*, y propone un flujo de actividades para la selección de una arquitectura de software candidata de acuerdo a los atributos de calidad que contempla. Dicho flujo de actividades es el siguiente:

1. Obtener requerimientos funcionales por medio de casos de uso.
2. Obtener requerimientos no funcionales.
   - Especificación de los requerimientos de calidad de acuerdo al dominio del problema.
   - Especificación de las restricciones tecnológicas.
   - Especificación de las características de calidad para cada caso de uso.
3. Priorizar casos de uso.
4. Detallar casos de uso.
5. Identificar la arquitectura candidata, de acuerdo a:
   - Experiencia (arquitectura previamente utilizada y/o patrones existentes)
   - Información clave obtenida de los casos de uso.
   - Conjunto de patrones o estilos arquitectónicos.
   - Acorde a los atributos de calidad.
   - Puede ser especificada gráficamente por medio de un estándar, diagrama de componentes y conectores, o formalmente utilizando lenguajes de descripción de arquitectura o lenguajes formales.
   - Las sub-características y atributos relevantes del dominio del problema, deben ser organizados en una tabla como la que se muestra en la figura 3.4.1.

Characteristics	Sub-Characteristics	Publisher/Subscriber	Repository	Comments and Results
Efficiency	Time behavior (time spent from data reception to data delivery)	Time (Reception)+ Time (store in DB)+ Time (send changes)+ Time (Publisher)+ Time (send changed values)	Time (Client Request)+ Time (Client invoices DB)+ Time (Data Server)+ Time (request)+ Time (DB)+ Time (response Data Server)+ Time (Response to Client)	Publisher/ subscriber is better
	Resource utilization (time)	Browser displays always	Browser displays on request	Repository is better
	Resource utilization (space)	Size (subscription DB) > Size (invoices DB)		Repository is better
Reliability (availability)	Maturity	Maturity (Reception)+ Maturity (DB)+ Maturity (Publisher)+ Maturity (Subscriber)+ Maturity (Subscription DB)	Maturity (Reception)+ Maturity (DB)+ Maturity (Data Server)+ Maturity (Client invoices DB)	Repository is better
	Fault tolerance	=	=	Depends on additional mechanisms
	Recoverability	=	=	Depends on additional mechanisms
Functionality	Security	Mechanism for subscription	Mechanism for each client request	Publisher/ subscriber is better
Portability	Adaptability	=	=	Depends on additional mechanisms
Usability	Learnability	=	=	Depends on the commercial browser used

Figura 3.4.1, Tabla de comparación de arquitecturas de acuerdo a atributos de calidad.

La tabla muestra una relación entre los atributos de calidad con las arquitecturas candidatas, así como observaciones o comentarios de dicha relación que ayudan a tomar la decisión de la arquitectura de software en la que se basará un sistema de información.

- Como se puede observar, las columnas muestran la arquitectura candidata a ser seleccionada. Las medidas son calculadas para cada uno de los atributos de acuerdo a las métricas propuestas por el *framework* mostrado en la figura 3.4.2.
- Cada atributo es priorizado de acuerdo a los requerimientos expresados en el modelo de calidad para el dominio del problema.

6. Aplicar el *framework* de medición arquitectónica basado en el ISO 9126-1 para seleccionar la arquitectura candidata, mostrado en la figura 3.4.2.

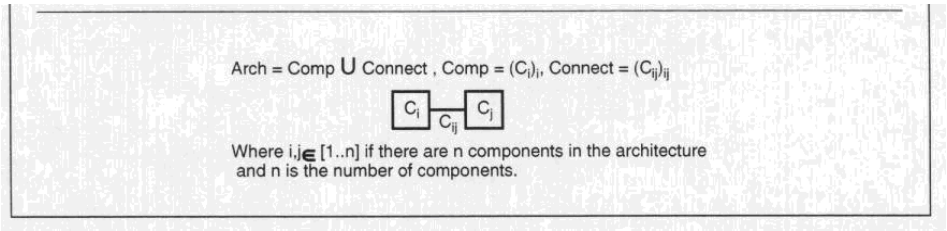


Figura 3.4.2, *Framework* para la medición arquitectónica.

7. Analizar la tabla de evaluación. Los valores de los atributos de calidad relevantes son analizados de acuerdo a los escenarios (posible *trade-offs*).

IEEE Std 1471-2000

El estándar IEEE 1471-2000 [17] recomienda prácticas enfocadas a las actividades de creación, análisis y sustento de una arquitectura de software de un sistema de gran escala, así como la documentación de la misma por medio de descripciones arquitectónicas. Además, el estándar establece un marco conceptual para descripciones arquitectónicas, así como el contenido del mismo.

El estándar considera y describe los siguientes elementos:

- Contexto de las descripciones arquitectónicas.
- Los *stakeholders* y sus roles.
- Las actividades de la arquitectura en el ciclo de vida.
- Los usos de una descripción arquitectónica.
- La documentación arquitectónica.
- La identificación de los *stakeholders* y sus intereses en la arquitectura.
- La selección de *viewpoint* arquitectónicos.
- Consistencia a través de las vistas arquitectónicas.
- El razonamiento arquitectónico.

El arquitecto de software puede utilizar este estándar como referencia para la documentación de una arquitectura de software.

3.5. Herramientas

Quality Function Deployment

Quality Function Deployment (*QFD*) es un método estructurado para planear y desarrollar un producto, de tal manera que se especifiquen claramente los deseos y necesidades del cliente, para después evaluar la capacidad del mismo en términos de su impacto o conocimiento de dichas necesidades (traducido de) [18]. En términos prácticos, *QFD* es una metodología para identificar y traducir necesidades y deseos del cliente a: requerimientos técnicos, características que se puedan medir y características de ingeniería para el desarrollo de un producto.

QFD hace uso de una tabla, llamada casa de calidad, en dónde registra las necesidades del cliente y las traduce a requerimientos técnicos, lo cual se va refinando a través de diferentes niveles, de tal manera que se cubran todos los requerimientos o características del producto. Una tabla *QFD* consiste de seis secciones (ver figura 3.5.1) (traducido de) [19] en donde se especifica textualmente:

- Necesidades y beneficios del cliente.
- Matriz de planeación, la cual registra:
  - Importancia para el cliente
  - Representación de la satisfacción del cliente
  - Representación de la satisfacción de la competitividad
  - Metas
  - Razón de mejora
  - Punto de ventas
  - Valor normalizado
- Respuesta técnica, necesidades del cliente traducidas a un lenguaje técnico.
- Relaciones entre la respuesta técnica y las necesidades del cliente, lo cual se puede medir con: no ligado (0), posiblemente ligado (1), moderadamente ligado (3), fuertemente ligado (9).
- Matriz técnica. Las respuestas técnicas son priorizadas sumando el valor de cada una de las columnas de la respuesta técnica.
- Correlación técnica. Representa la relación entre las respuestas técnicas (grado de impacto).

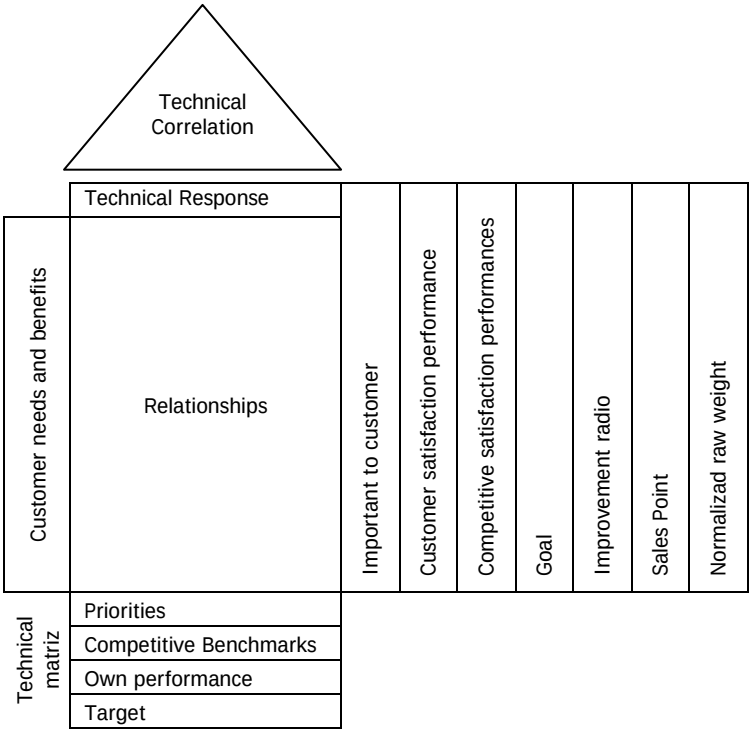


Figura 3.5.1, Casa de calidad de QFD.

Los requerimientos arquitectónicos de una arquitectura de software pueden ser identificados y traducidos a requerimientos técnicos mediante el uso de *QFD*, partiendo de la idea de que dicha arquitectura será utilizada para obtener un producto de software. De esta manera, existe una adaptación de *QFD* para el desarrollo de software, la cual es denominada *Software Quality Function Deployment* (*SQFD*). *SQFD* es una técnica de solicitud de requerimientos, adaptable a cualquier metodología de ingeniería de software, que cuantificablemente, solicita y define requerimientos críticos del cliente (traducido y modificado de) [20]. Los factores que toma en cuenta *QFD* son los mismos considerados por *SQFD*, pero con el enfoque de software.

Con lo anterior, hemos descrito brevemente: 1) algunos conceptos básicos de arquitecturas de software; 2) algunas propuestas de proceso para arquitecturas de software, en dónde explicamos el modelo de Desarrollo Basado en Arquitecturas, que contempla los pasos fundamentales para construir una arquitectura de software, y el modelo Diseño de Arquitectura de Software: Evaluación y Transformación, que propone un método de diseño de arquitecturas de software mediante la evaluación y transformación iterativa hasta obtener los resultados esperados de dicha arquitectura; 3) técnicas para el diseño de arquitecturas de software, en dónde expusimos el Diseño Basado en Arquitecturas, que propone la manera en cómo se puede descomponer el concepto de un sistema de software para llegar a elementos de diseño, el método de Arquitectura Basada en Escenarios, el cual propone el uso de un matriz de obtención de escenarios para relacionarlos con los diferentes atributos de calidad esperados en el sistema y con los diferentes *stakeholders* interesados en el mismo, las vistas arquitectónicas de Kruchten y Nord, y el método de Diseño de Arquitectura de Software Efectivo con UML, dónde se presentan las vistas que recomienda Nord para describir una arquitectura de software, así como los diagrama UML que se proponen para describir cada vista; 4) algunos estándares relacionados con arquitecturas de software, en dónde mencionamos el modelo ISO 9126-1, que especifica los atributos de calidad que pueden tener un sistema de software y un flujo de actividades para seleccionar arquitecturas candidatas de acuerdo a dichos atributos de calidad, y el estándar IEEE 1471-2000, que recomienda prácticas enfocadas a las actividades de creación, análisis y sustento de una arquitectura de software; y finalmente 5) herramientas adicionales, en dónde expusimos la metodología que sigue *QFD* para identificar necesidades de los clientes y convertirlos en requerimientos técnicos de arquitecturas de software.

4. Propuesta de Solución

La propuesta de solución para nuestro problema se encuentra basada en las diferentes aportaciones de importantes autores que fueron descritas en la sección de trabajo previo del presente documento. La esencia de nuestra propuesta consiste en abstraer los elementos relevantes de cada uno de los trabajos previos, con el objetivo de definir un modelo de proceso para elaboración de arquitecturas de software con la consideración de varias de las aportaciones que hacen los expertos en esta materia.

De acuerdo a la problemática expuesta en la sección de la descripción del problema del presente documento, y después de analizar los diferentes conceptos, modelos, técnicas y herramientas que se proponen en la literatura en el campo de arquitecturas de software, presentamos a continuación el Modelo para el Desarrollo de Arquitecturas de Software – MODAS, el cual representa un proceso definido para la elaboración de arquitecturas de software, mismo que permite elaborar una arquitectura nueva, reutilizar una arquitectura existente o dar mantenimiento a una arquitectura en operación. El modelo parte del documento de especificación de requerimientos de software en el caso de una arquitectura nueva, y del documento de arquitecturas de software para la definición de una arquitectura reutilizada o en mantenimiento. Nuestro modelo contempla los siguientes pasos, mismos que se representan la figura 4.1 de acuerdo a sus entradas, salidas y flujos correspondientes:

1. Identificación de los requerimientos arquitectónicos
2. Diseño de la arquitectura
3. Verificación de la arquitectura
4. Adecuación de la arquitectura
5. Documentación de la arquitectura
6. Evaluación de la arquitectura
7. Validación de la arquitectura

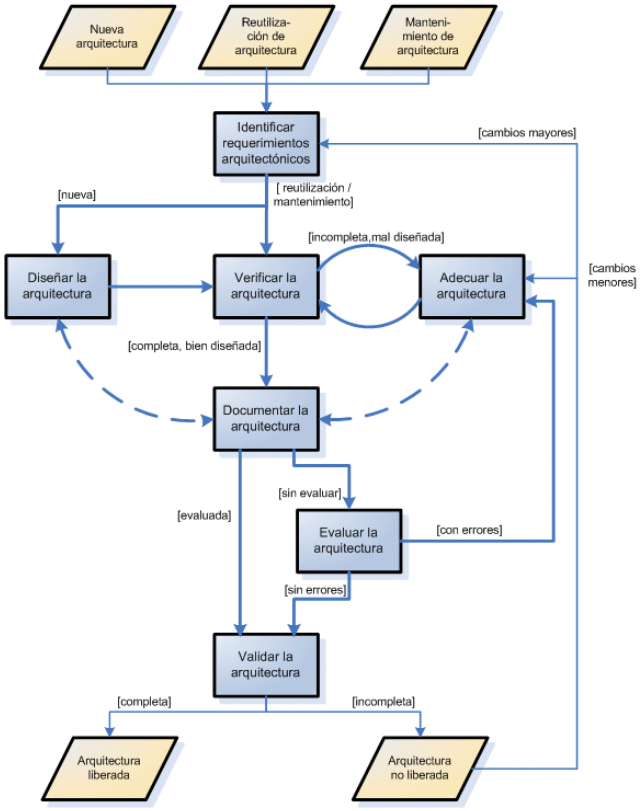


Figura 4.1, Modelo pare el desarrollo de arquitecturas de software – MODAS – Proceso General.

La figura 4.1 representa el panorama general de la solución que muestra nuestro modelo. Como se puede observar, el proceso puede recibir como objetivo la definición de una nueva arquitectura de acuerdo a los requerimientos de software, la reutilización de una arquitectura existente observando que cumple con los requerimientos de software establecidos, o bien, el mantenimiento a una arquitectura de software para incorporación de nuevos requerimientos. La salida del proceso puede ser una arquitectura liberada, o una arquitectura no aprobada, la cual debe ser adecuada en el caso de que existan cambios menores, o volver a generar el proceso de manera completa en el caso de contar con cambios mayores.

El modelo de proceso permite diseñar una arquitectura después de identificar los requerimientos arquitectónicos, o bien, verificar directamente que la arquitectura cumpla con los requerimientos arquitectónicos establecidos si se trata de una arquitectura reutilizada o en mantenimiento. En cualquiera de los dos casos, este último paso verificará la completes y el correcto diseño de la arquitectura, con el objetivo de adecuarla si se observan inconsistencias, o bien documentar el diseño realizado. Es importante señalar que tanto en el momento de realizar el diseño de la arquitectura, como en su adecuación, pueden documentarse ciertas elementos, mismos que se formalizarán en el paso de documentación de la arquitectura.

Para llevar a cabo nuestro paso de evaluación de la arquitectura, haremos uso del proceso para evaluación de arquitecturas de software, propuesto por Omar Gómez [21]. Este proceso recibirá como entrada el documento de arquitectura de software obtenido, y después de realizar ciertos análisis con herramientas como SNA (Survivability Network Analysis), ALMA (Architecture-Level Modifiability Analysis), entre otros, reportará si la arquitectura es adecuada con respecto a los atributos de calidad especificados, si existe algún conflicto entre los mismos, y recomendaciones para su adecuación. A su vez, este proceso mencionará el impacto y el riesgo que correrá el diseño de la arquitectura de software sin la incorporación de ciertos elementos, estilos o patrones. En caso de que la arquitectura no apruebe la evaluación, se llevará a cabo el proceso de adecuación, verificación, documentación y evaluación nuevamente. Si por el contrario la arquitectura aprueba la evaluación, se ejecutara el paso de validación, mismo que se encarga de estudiar el diseño de la arquitectura con los diferentes *stakeholders* para determinar si cubre sus requerimientos establecidos, o bien debe sufrir adecuaciones menores o mayores. Si se descubren cambios menores en la validación, entonces se llevará a cabo el proceso de adecuación, verificación, documentación y validación nuevamente, pero si por el contrario se determinan cambios mayores, como incorporación de nuevos requerimientos, entonces se ejecutará nuevamente todo el proceso.

En la siguiente sección del presente documento definiremos a detalle las actividades a realizar, las entradas y salidas internas, las herramientas que apoyarán dicha actividad y las métricas que ayudarán a controlar y mejorar el proceso para cada uno de los pasos del modelo.

5. Modelo para el desarrollo de arquitecturas de software

El modelo de proceso para el desarrollo de arquitecturas de software que proponemos, consta de siete pasos básicos: identificar los requerimientos de software, diseñar la arquitectura, verificar la arquitectura, adecuar la arquitectura, documentar la arquitectura, evaluar la arquitectura, y validar la arquitectura. Cada uno de estos pasos recibe ciertas entradas, proporciona ciertas salidas, maneja entradas y salidas internas, hace uso de herramientas, y en ciertos casos define una métrica para el control y mejoramiento del mismo proceso.

Nuestro modelo para el desarrollo de arquitecturas de software, se encuentra fundamentado en el modelo que describen Bass y Kazman conocido como: "Desarrollo Basado en Arquitecturas" (*Architecture-Based Development*) [3]. Este modelo consta de seis pasos, los cuales son: obtener los requerimientos arquitectónicos, diseñar la arquitectura, documentar la arquitectura, analizar la arquitectura, generar la arquitectura y mantener la arquitectura. Los autores describen las actividades que se realizarán en cada paso, proponiendo el uso de ciertas herramientas para algunos de ellos, como por ejemplo: una matriz de obtención de escenarios para obtener los requerimientos y el método ATAM (Architecture Tradeoff Análisis Method) para realizar el análisis de la arquitectura. Sin embargo, al analizar este modelo, podemos observar que no se menciona con mucho detalle las tareas específicas a realizar internamente, no son rápidamente identificadas, no se mencionan entradas y salidas internas, ni métricas para controlar y mejorar el proceso. En la figura 5.1 podemos observar la relación que guarda nuestro modelo con el modelo *Architecture-Based Development*.

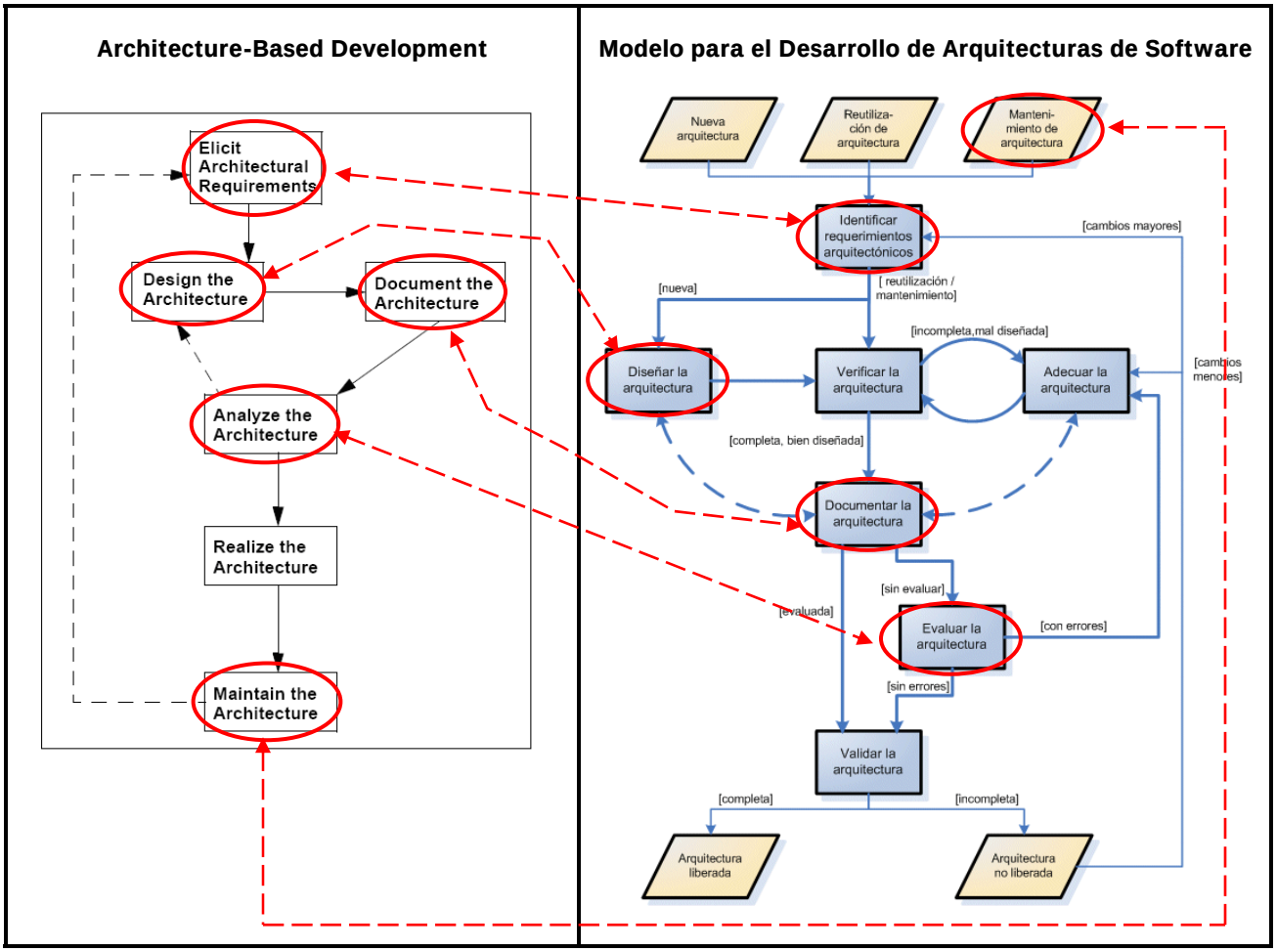


Figura 5.1, Architecture-Based Development en relación con el Modelo para el Desarrollo de Arquitecturas de Software.

Como se puede observar en la figura 5.1, el modelo Architecture-Based Development considera un paso enfocado a generar la arquitectura de software mediante código, el cual no es considerado por el Modelo de Desarrollo de Arquitecturas se Software. A su vez, este primer modelo considera el mantener la arquitectura de software como un paso dentro del proceso, el cual se conecta nueva mente al inicio del mismo. Por su parte, nuestro segundo modelo considera el mantenimiento de la arquitectura como un objetivo de entrada del proceso, el cual, después de identificar los requerimientos arquitectónicos, omitirá el paso de diseñar la arquitectura, para directamente verificarla y adecuarla. Por su parte, el Modelo de Desarrollo de Arquitecturas se Software, considera tres pasos adicionales: verificar la arquitectura, el cual determinará si la arquitectura se encuentra completa y sin errores de diseño antes de ser evaluada; adecuar la arquitectura si requiere de cambios después de verificación o validación, y finalmente validar la arquitectura con los diferentes *stakeholders* una vez que se haya generado el documento de arquitectura de software y haya pasado la evaluación correspondiente.

De esta manera, podemos decir que el Modelo de Desarrollo de Arquitecturas se Software se fundamenta en el modelo Architecture-Based Development, incorporando pasos adicionales que ayudan a su correcto diseño, así como la incorporación de elementos que facilitan la identificación de actividades a realizar. Los elementos que adicionalmente considera el modelo de proceso son:

- Entradas y salidas del proceso general.
- Definición de los pasos internos de cada uno de los pasos generales del proceso.
- Detalle de entradas y salidas de las actividades de los pasos del proceso.
- Uso de herramientas que apoyan cada pasos del proceso.
- Incorporación de métricas que ayudan a controlar y mejorar el proceso.

El modelo de proceso permite desarrollar una nueva arquitectura de software, reutilizar una arquitectura de software existente para cubrir los requerimientos arquitectónicos de un nuevo sistema, o bien, dar mantenimiento a una arquitectura que se encuentre en operación.

Las entradas que presenta el modelo de proceso son:

- Documento de especificación de requerimientos de software para el caso de la elaboración de una nueva arquitectuela.
- Documento de especificación de requerimientos de software y documento de arquitectura de software para el caso de la reutilización de una arquitectura previamente definida.
- Documento de arquitectura de software y definición de nuevos requerimientos para el caso del mantenimiento de una arquitectura de software en funcionamiento.

Las salidas que presenta el modelo de proceso son:

- Documento de arquitectura de software liberado
- Documento de arquitectura de software sin liberar, con observaciones y recomendaciones de adecuación.

Los pasos que considera el Modelo para el Desarrollo de Arquitecturas de Software, son:

1. Identificar requerimientos arquitectónicos, el cual, mediante el documento de especificación de requerimientos, o definición de nuevos requerimientos de software, permite analizar los requerimientos funcionales y requerimientos no funcionales para ser convertidos a requerimientos técnicos que darán las directrices del diseño o adecuación de la arquitectura de software, según se el caso de definir una nueva arquitectura, reutilizar una arquitectura existente o dar mantenimiento a una arquitectura en operación.

2. Diseñar la arquitectura, mediante el cual el sistema se descompone en subsistemas, componentes y elementos de diseño, mismos que al conceptualizarse y asignar funcionalidad representarán nuestro diseño arquitectónico.

3. Verificar la arquitectura, por medio del cual determinaremos la completes de los requerimientos en el diseño, así como el correcto diseño de los componentes de software, dando la posibilidad a que la evaluación se enfoque en la detección de problemáticas profundas y con un mayor grado de especialización, de los atributos de calidad de la arquitectura.

4. Adecuar la arquitectura, el cual permitirá realizar ajustes o adaptaciones al diseño de la arquitectura de software una vez que fue verificada, evaluada o validada y se identificó que no cubre los requerimientos establecidos, o bien se encuentran mal diseñados.

5. Documentar la arquitectura de software, por medio del cual representaremos gráfica y textualmente la arquitectura de software, además de plasmar los requerimientos que cubre, orientadores arquitectónicos que dan la directriz de la arquitectura, el uso de estilos arquitectónicos, patrones arquitectónicos y/o patrones de diseño, decisiones de diseño, intereses de los diferentes *stakeholders*, entre otros aspectos.

6. Evaluar la arquitectura de software, mediante el cual determinaremos si la arquitectura de software cumple con los requerimientos no funcionales o atributos de calidad, si existe algún conflicto entre dichos atributos de calidad, o bien el impacto y riesgo que puede traer como consecuencia la ausencia de ciertos estilos, patrones o componentes en la arquitectura de software.

7. Validar la arquitectura de software, por medio del cual, finalmente determinaremos con los diferentes *stakeholders* que la arquitectura de software se encuentre acorde a los requerimientos establecidos y cubra todas las necesidades. Si al validarla se identifican cambios menores, podrá ejecutarse el paso de adecuación, documentación y validación nuevamente; pero si se identifican cambios mayores, podrá ejecutarse todo el proceso nuevamente.

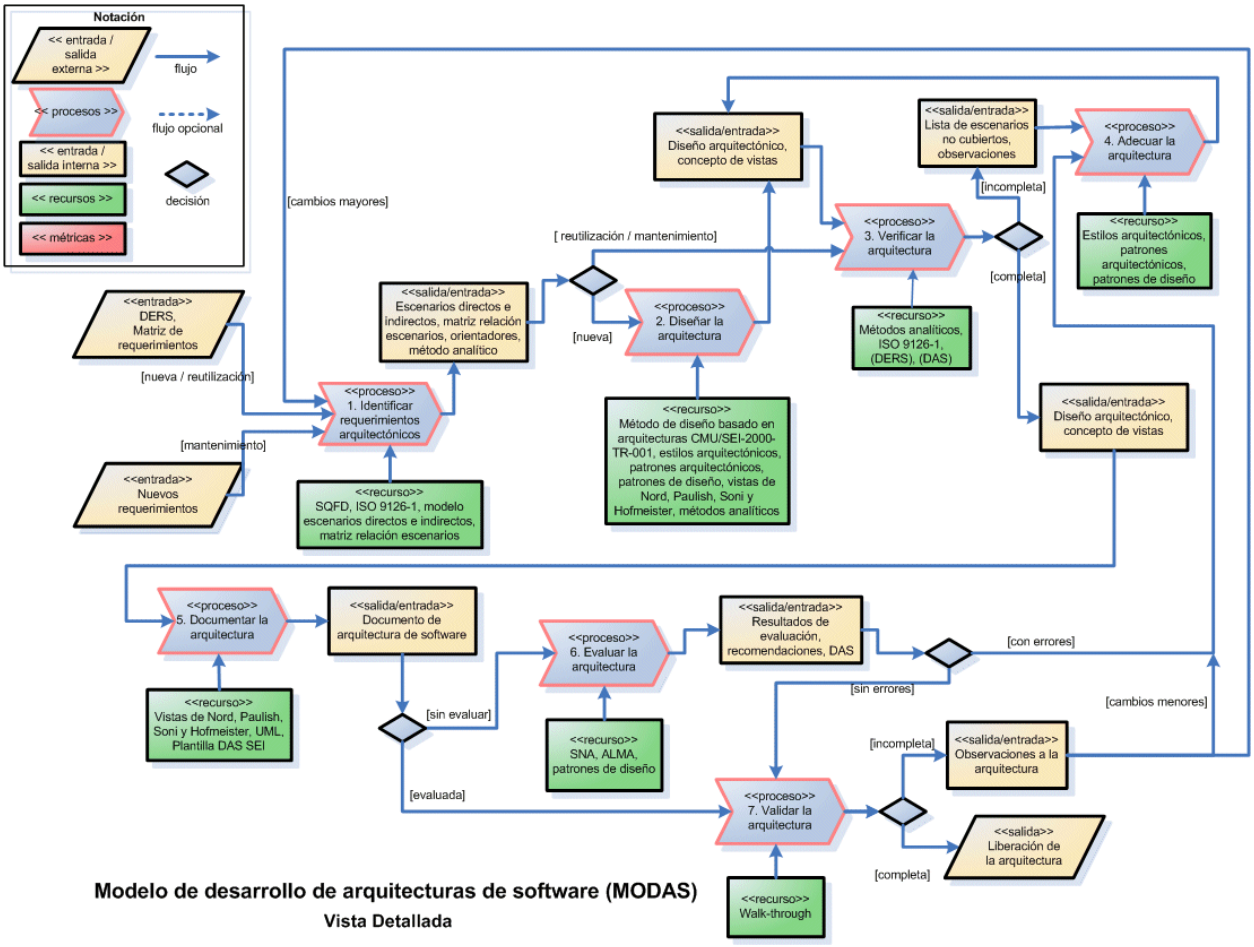


Figura 5.2, Vista detallada del Modelo para el Desarrollo de Arquitecturas de Software.

La figura 5.2 muestra una vista detallada del Modelo para el Desarrollo de Arquitecturas de Software, en dónde se especifican las entradas y salidas de cada paso, así como las herramientas que apoyarán la realización de las actividades de cada uno de ellos. La definición de cada paso con cada uno de sus elementos la describiremos más adelante. Hacemos uso de uno de los *profiles* de UML (Unified Modeling Language) para modelar procesos de negocio, creado por Eriksson y Penker [2]. Por medio de este *profile* de UML describiremos cada paso de nuestro modelo de procesos. La notación que utilizaremos se describe en la tabla 5.3:

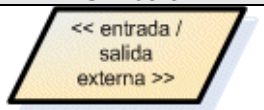
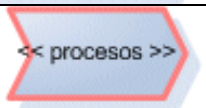
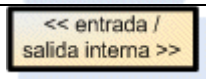
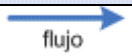
Símbolo	Descripción
	Representa una entrada o salida del proceso general, es decir de manera externa.
	Representa un paso del proceso, ya sea en el proceso general, así como los sub-pasos internos.
	Indica una entrada o salida interna del proceso, es decir, en los sub-pasos.
	Se refieren a las herramientas que se utilizarán como recursos dentro de cada uno de los pasos del modelo
	Indica las métricas que se registrarán en ciertos pasos del proceso para controlarlo y mejorarlo.
	Se refiere a la indicación del seguimiento de un flujo, la entrada la un recurso, o bien el registro de una métrica.
	Se refiere a la indicación de un flujo opcional.
	Representa la decisión sobre un camino que puede seguir el proceso de acuerdo a las salidas obtenidas en el paso previo.

Tabla 5.3, Notación para describir el Modelo para el Desarrollo de Arquitecturas de Software.

A continuación detallaremos el flujo interno que contiene cada uno de los pasos de nuestro modelo representado en la figura 5.2, así como la descripción de los elementos que los conforman.

5.1. Identificar requerimientos arquitectónicos

Este paso permite identificar los requerimientos arquitectónicos del sistema que será desarrollado. El análisis de los mismos se realizará a través del documento de especificación de requerimientos de software y de una matriz de seguimiento a requerimientos, para el caso de una arquitectura nueva o reutilizada, o bien, la definición de los nuevos requerimientos, para el caso del mantenimiento de una arquitectura existente. El objetivo de este paso es analizar los requerimientos funcionales y requerimientos no funcionales del sistema para ser convertidos en requerimientos técnicos, los cuales darán las directrices del diseño o adecuación de la arquitectura de software, según sea el caso del objetivo inicial de entrada del proceso.

[2] Eriksson, Hans-Erik and Penker, Magnus, *Business Modeling with UML, Business Patterns at Work*, Wiley Computer Publishing, 2000.

La figura 5.1.1 muestra las actividades que sigue este paso, especificando sus entradas, salidas, herramientas a utilizar y registro de métricas.

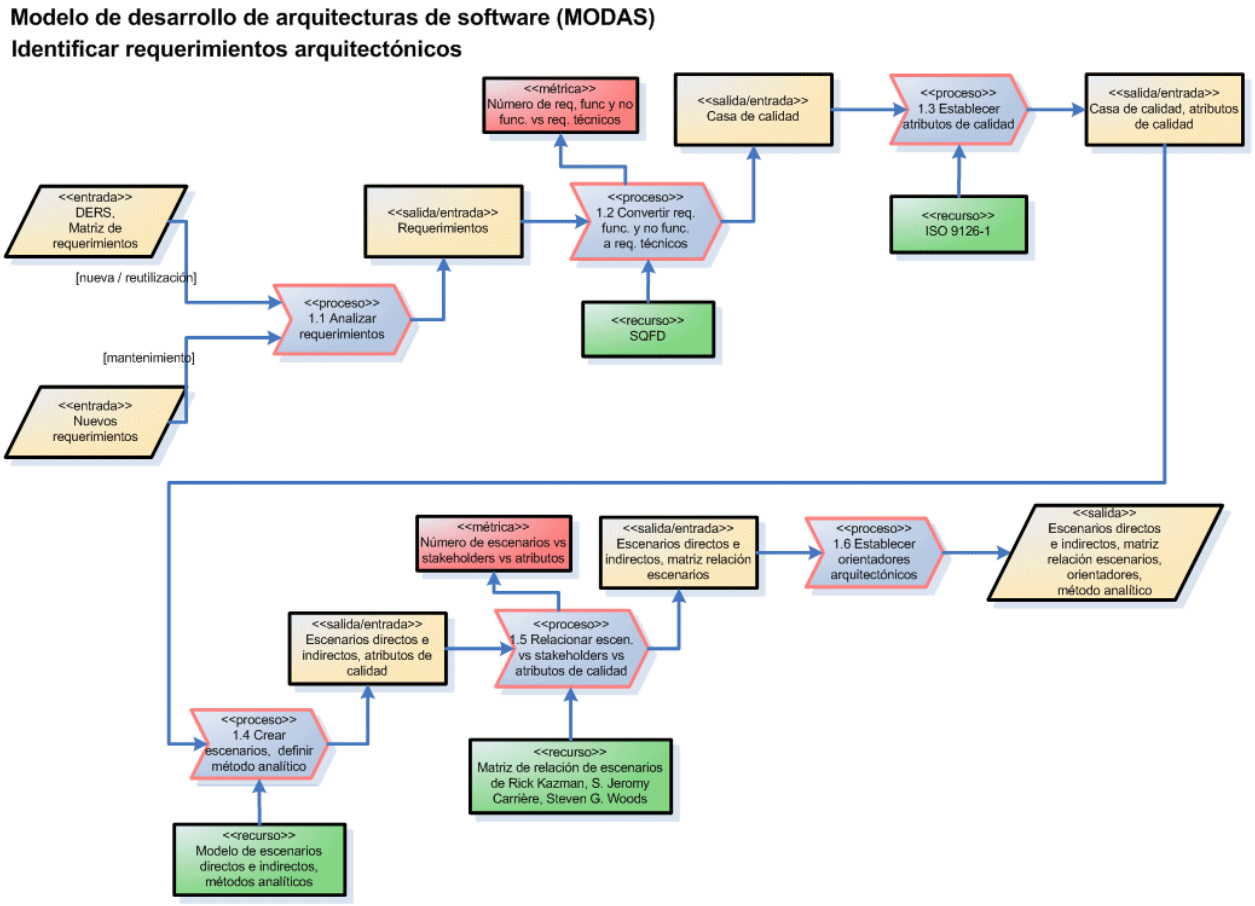


Figura 5.1.1, Paso: Identificar requerimientos arquitectónicos del Modelo para el Desarrollo de Arquitecturas de Software.

Los elementos a considerar para este paso del proceso son:

Elemento	Descripción
Entradas	Documento de especificación de requerimientos de software y matriz de seguimiento a requerimientos, para el caso de una arquitectura nueva o reutilizada, o la definición de los nuevos requerimientos, para el caso del mantenimiento de una arquitectura existente.
Salidas	Escenarios directos e indirectos que afectan la arquitectura, matriz de relación de escenarios, orientadores arquitectónicos y la selección del posible método analítico que podrá medir los requerimientos no funcionales en el diseño arquitectónico.
Recursos	SQFD, el ISO 9126-1, modelo de escenarios directos e indirectos, la matriz de relación de escenarios de Kazman, Carrière y Woods [9], y métodos analíticos para medir atributos de calidad.
Métricas	Número de requerimientos funcionales y no funcionales entre número de requerimientos técnicos definidos, así como número de escenarios definidos entre el número de *stakeholders* y entre número de atributos de calidad establecidos.
Roles	Analista y arquitecto.

Como mencionamos anteriormente, este paso parte con el documento de especificación de requerimientos de software, matriz de seguimiento a requerimientos, o bien un listado de requerimientos funcionales y requerimientos no funcionales del sistema, cuya arquitectura está siendo elaborada. Los requerimientos funcionales pueden estar especificados mediante casos de uso de UML.

Por medio de los requerimientos funcionales y no funcionales del sistema, el arquitecto deberá analizar e identificar los requerimientos que afectan la arquitectura, así como los atributos de calidad que se espera cumpla la misma. Los diferentes *stakeholders*, a través de la especificación de sus diferentes intereses sobre la arquitectura de software, pueden ayudar a determinar los atributos de calidad considerados para dicha arquitectura. El arquitecto puede apoyarse de la ayuda del analista o analistas quiénes especificaron los requerimientos del sistema cuya arquitectura está siendo construida para la identificación de los requerimientos arquitectónicos.

Una vez que contamos con la identificación de requerimientos funcionales y requerimientos no funcionales que afectan la arquitectura, debemos convertirlos a requerimientos técnicos, mismos que reflejan los requerimientos arquitectónicos para nuestro diseño. Para la obtención de estos requerimientos arquitectónicos, se propone el uso de la técnica *SQFD*, con el objetivo de traducir requerimientos funcionales, no funcionales e intereses del cliente a requerimientos técnicos que consideren atributos de calidad para la arquitectura de software.

La secuencia de pasos que se debe seguir de acuerdo a (traducido y modificado de) [19] [20] para la preparación de una casa de calidad *SQFD* es la siguiente, misma que se ilustra con la figura 5.1.2:

1. Registrar los requerimientos funcionales del sistema en la sección de “requerimientos y beneficios del cliente” de la casa de calidad representados por casos de uso. Los casos de uso que se encuentren relacionados deben ser agrupados. Las sentencias de los requerimientos son comúnmente sentencias cortas registradas específicamente en la terminología del cliente (nombre del caso de uso) y son acompañadas de una definición detallada (caso de uso).

2. En conjunto con los clientes, los requerimientos son convertidos a un enunciado técnico y medible del producto de software. Estos enunciados corresponden a los componentes de la arquitectura de software (elementos), mismos que son colocados en la sección de “respuesta técnica” de la casa de calidad.

3. Los clientes participan completando la correlación técnica de la matriz identificando la solidez de las relaciones entre los diferentes requerimientos de usuario y la especificación técnica del producto. Es importante obtener consenso concerniente a las relaciones fuertes.

4. La importancia o prioridad de cada uno de los casos de uso deben ser especificada y registrada en la sección de “importancia para el cliente”. Esta importancia puede ser identificada con 1, 3 y 9, dónde 1 es el de menos importante y el 9 es el más importante.

5. Determinar la relación entre componentes de la arquitectura y los casos de uso. Para cada uno de los casos de uso, la arquitectura de software determina cual responsabilidad de un componente son utilizados por el caso de uso. Posteriormente, buscar el resultado de las relaciones, el cual es utilizado para el indicador de prioridad de los componentes, sumando los resultados de la multiplicación de las prioridades de los requerimientos del cliente por los valores de la correlación entre los requerimientos del cliente y la especificación técnica del producto. Este valor es convertido a porcentaje. El resultado es registrado en la sección de relaciones. El resultado final del proceso de *SQFD* contendrá una especificación técnica del producto y su prioridad.

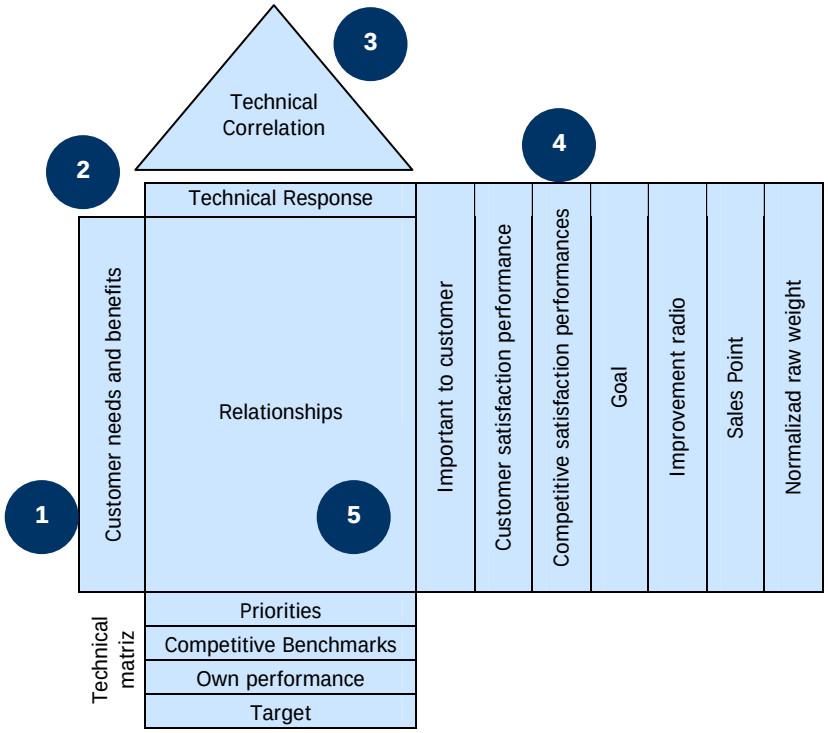


Figura 5.1.2, Casa de calidad de SQFD.

Con lo anterior, hemos descrito la manera en cómo obtener los requerimientos arquitectónicos por medio de una especificación técnica del producto de software. Para obtener información más detallada acerca de esta técnica, se recomienda consultar las referencias: [19] [20].

Para efectos del control de nuestro proceso, en este paso debemos registrar la métrica de número de requerimientos funcionales y no funcionales entre el número de requerimientos técnicos establecidos, con el objetivo de conocer cuántos requerimientos cubrirá la arquitectura de software, así como verificación de esta misma cobertura.

Una vez que hemos identificado los requerimientos arquitectónicos, por medio de la casa de calidad construida, el arquitecto debe establecer claramente los atributos de calidad que darán la directriz del diseño de la arquitectura acorde a los intereses y necesidades de los diferentes *stakeholders*. Para esta actividad, recomendamos analizar y estudiar el modelo de calidad ISO 9126-1, con el objetivo de ayudar a determinar los atributos de calidad específicos para la arquitectura de software, de acuerdo a la clasificación y definición que hace de los mismos.

El siguiente paso es la generación de escenarios directos e indirectos propuestos por Kazman, Carrière y Woods [9] para la descripción del comportamiento de la arquitectura con base en los requerimientos arquitectónicos y atributos de calidad establecidos. Los escenarios directos son enunciados cortos que describen la utilización normal que los usuarios finales dan al sistema construido de la arquitectura de software; los escenarios indirectos son enunciados que representan cambios a la arquitectura de acuerdo a: utilización de nuevas plataformas de hardware o software, agregar una nueva característica, integrarse con una nueva pieza de software, reemplazar alguna funcionalidad existente, escalar el sistema, etc. Los escenarios representan acciones en el sistema que los diferentes *stakeholders* desean realizar de acuerdo a sus intereses, mismos que se verán reflejadas por la respuesta de los diferentes atributos de calidad de la arquitectura.

Al contar con los escenarios directos e indirectos, estamos en la posibilidad de determinar la manera en cómo verificaremos que el diseño de la arquitectura cubre los requerimientos, reflejados en dichos escenarios, de manera correcta y complete. Para llevar a cabo esa verificación existen dos posibilidades, realizarla de manera cuantitativa, solo con apoyo de los mismos escenarios, o de manera cuantitativa, por medio de métodos analíticos. Un método analítico es la definición concreta de los pasos que tenemos que llevar a cabo para verificar de manera formal, con el apoyo de procedimientos matemáticos, que un requerimiento en específico se comporte de acuerdo a los resultados esperados. Regularmente los métodos analíticos se encuentran enfocados a analizar requerimientos no funcionales, como: desempeño, facilidad de mantenimiento, confiabilidad, etc. De esta manera, después de crear nuestros escenarios, debemos determinar si la verificación la haremos de manera cualitativa o cuantitativa, y de seleccionar ésta última, elegir el método analítico correspondiente. Esta decisión nos servirá en el paso de verificar la arquitectura de nuestro modelo para el desarrollo de arquitecturas de software.

Nuestra siguiente actividad es relacionar escenarios con los diferentes *stakeholders* y a su vez con los distintos atributos de calidad determinados. Sabiendo que múltiples *stakeholders* pueden estar contemplados en múltiples escenarios de acuerdo a su interés y que dicho interés se refleja en distintos atributos de calidad, proponemos hacer uso de la matriz de obtención de escenarios de Kazman, Carrière y Woods, con el objetivo de estudiar cada escenario desde la perspectivas de los *stakeholders* y desde la perspectiva de cada atributo de calidad o interés.

La matriz de relación de escenarios de Kazman, Carrière y Woods se ilustra con la figura 5.1.3. Se puede observar que se trata de una matriz tridimensional, en dónde los escenarios deben ser listados en la primera fila del primer cuadrante y los atributos de calidad a manera de nombre de columnas de la tabla. Los diferentes *stakeholders* deben ser listados en la primera fila del segundo cuadrante. De esta manera, podemos relacionar a los escenarios/atributos de calidad/*stakeholders* por medio de la intersección fila/columna de cada cuadrante y poder contar con las diferentes perspectivas de los diferentes *stakeholders* y que sean contempladas dentro de la arquitectura de software.

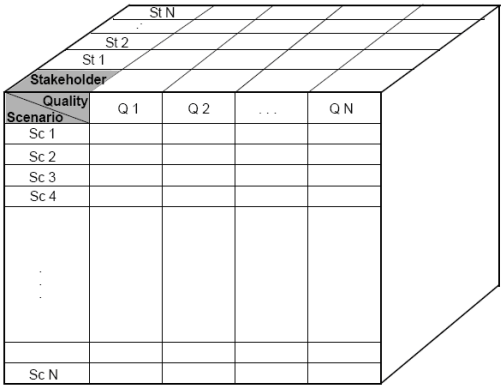


Figura 5.1.3, Matriz de relación de escenarios.

Al construir la matriz de relación de escenarios, podemos analizarla para conocer la relación que tienen los escenarios directos e indirectos de la arquitectura de software con los diferentes *stakeholders*, así como con los distintos atributos de calidad de la arquitectura. Para ayudar al control de nuestro proceso, en este paso debemos registrar la métrica de número de escenarios directos e indirectos entre el número de *stakeholders*, así como el número de atributos de calidad, con el objetivo de contar con elemento cuantitativo de esta relación, misma que servirá para verificar la cobertura de escenarios en el paso de verificar la arquitectura.

El último paso de la identificación de requerimientos arquitectónicos, es establecer los orientadores arquitectónicos (*architectural drivers*). Los orientadores arquitectónicos son el conjunto de necesidades, elementos, decisiones y determinaciones que marcan las directrices hacia dónde tenemos que dirigir el diseño de la arquitectura. Estos orientadores arquitectónicos los podremos obtener mediante los requerimientos arquitectónicos identificados, atributos de calidad concretados, escenarios directos e indirectos generados y la relación identificada de dichos escenarios con los diferentes *stakeholders* y con los distintos atributos de calidad. Todos estos elementos, al ser analizados, representan las directrices u orientadores que deben ser cuidadosamente considerados en el diseño de la arquitectura de software.

Podemos concluir que en el paso de obtención de requerimientos arquitectónicos, hemos visto: como recibiremos los requerimientos funcionales y no funcionales para elaborar una arquitectura de software; cómo haremos uso de la técnica *SQFD* para convertir dichos requerimientos funcionales y no funcionales a requerimientos técnicos; que debemos establecer los atributos de calidad de la arquitectura que esteremos construyendo; cómo podemos dividir los requerimientos provenientes de casos de uso y requerimientos no funcionales en escenarios directos e indirectos; cómo relacionar éstos con los diferentes *stakeholders*, así como con los distintos atributos de calidad de la arquitectura, y finalmente que debemos analizar todos los elementos generados para establecer los orientadores arquitectónicos, mismos que darán la pauta para el diseño de la arquitectura de software.

Nuestros escenarios directos e indirectos, la matriz de relación de escenarios, los orientadores arquitectónicos y el posible método analítico seleccionado, representan las salidas de este paso del proceso, mismos que servirán como entrada para el paso de diseñar la arquitectura o verificar la arquitectura, de acuerdo a los objetivos de la construcción de la arquitectura.

Las métricas obtenidas en este paso del proceso pueden ser utilizadas para determinar la cobertura de requerimientos en la arquitectura de software en el paso de verificación de la arquitectura.

5.2. Diseñar la arquitectura

El paso de diseñar la arquitectura nos permitirá descomponer el sistema en subsistemas, componentes y elementos de diseño, mismos a los que al conceptualizarse les asignaremos funcionalidad para obtener nuestro diseño arquitectónico. La figura 5.2.1 muestra las actividades que sigue este paso, especificando entradas, salidas, herramientas a utilizar y registro de métricas.

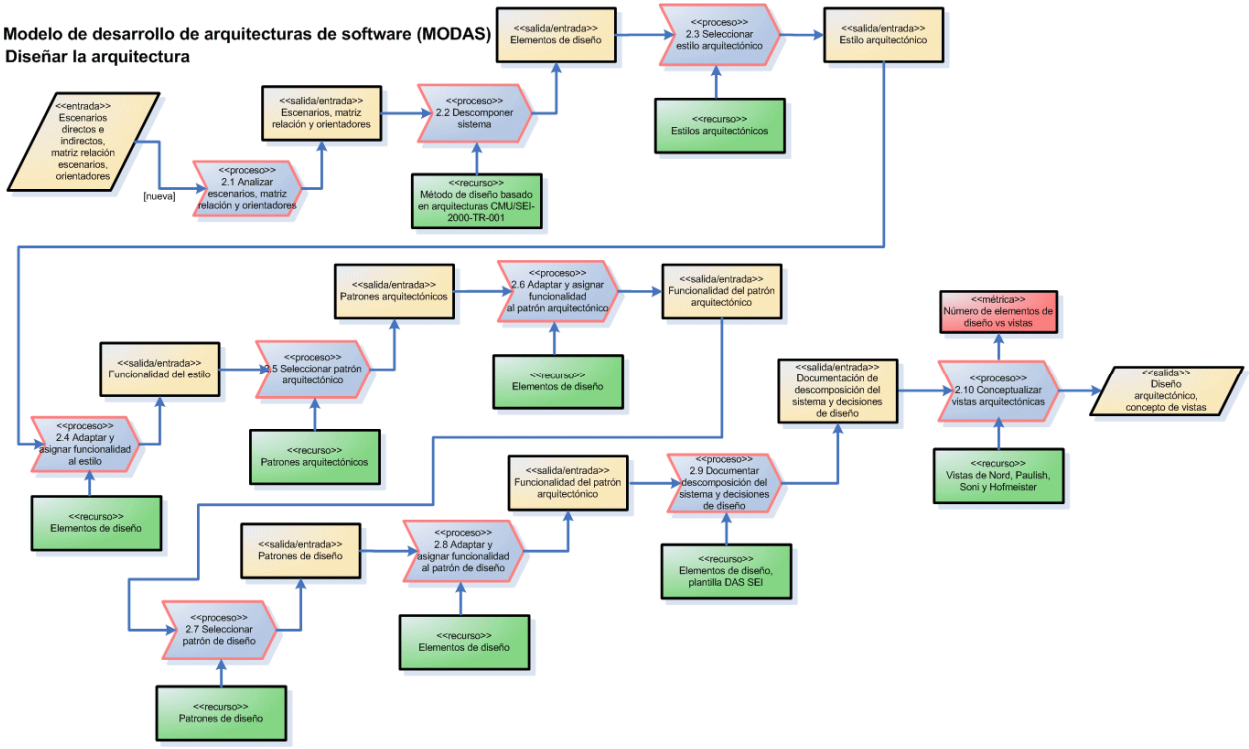


Figura 5.2.1, Paso: Diseñar la arquitectura del Modelo para el Desarrollo de Arquitecturas de Software.

Los elementos a considerar para este paso del proceso son:

Elemento	Descripción
Entradas	Escenarios directos e indirectos, matriz de relación de escenarios y orientadores arquitectónicos.
Salidas	Diseño arquitectónico y el concepto de las vistas arquitectónicas.
Recursos	Método de diseño basado en arquitecturas CMU/SEI-2000-TR-001 [12], estilos arquitectónicos, patrones arquitectónicos, patrones de diseño, plantilla de documento de arquitecturas de software del Instituto de Ingeniería de Software de la Universidad de Carnegie Mellon, y las vistas arquitectónicas de Vistas de Nord, Paulish, Soni y Hofmeister.
Métricas	Número de elementos de diseño entre número de vistas seleccionadas.
Roles	Arquitecto.

Una vez que obtuvimos los requerimientos arquitectónicos, el siguiente paso es diseñar la arquitectura de software. Como primera actividad, deberemos estudiar los diferentes escenarios directos e indirectos obtenidos en nuestro paso anterior, así como la matriz de relación de escenarios para reconocer cada una de las relaciones que tienen dichos escenarios con los diferentes *stakeholders*, y con los distintos atributos de calidad. Al examinar cada uno de estos elementos cuidadosamente, podremos hacer las consideraciones pertinentes para descomponer el sistema adecuadamente y llegar a un diseño arquitectónico que refleje cada una de las funcionalidades y atributos de calidad establecidos.

Para obtener el diseño de la arquitectura de software, utilizaremos el Método de Diseño Basado en Arquitectura propuesto por Bachmann, Bass, Chastek, Donohoe y Peruzzi [12]. Este método se basa en la consideración de elementos orientadores que guían o conducen el diseño de la arquitectura (*architectural drivers*), lo cual está representado por la combinación de los requerimientos funcionales y requerimientos no funcionales a considerar. De esta manera, haremos uso de este método para llevar a cabo nuestra segunda actividad en el diseño de la arquitectura, descomponer el sistema.

Los pasos que se deben seguir para obtener el diseño están basados en: la descomposición de la función del sistema, y en la comprensión de los escenarios directos e indirectos con su relación con los diferentes *stakeholders*, así como con los diferentes atributos de calidad. Esta descomposición y compresión de escenarios tiene como parte de sus objetivos, seleccionar estilos arquitectónicos, patrones arquitectónicos y patrones de diseño para los elementos de diseño generados.

A continuación describiremos la manera en cómo deberemos descomponer el sistema de acuerdo a su funcionalidad para obtener elementos de diseño, de tal manera que nos permita llevar a cabo la representación de la arquitectura de software.

Desde su nivel más alto, el sistema y su infraestructura deben ser descompuestos en subsistemas conceptuales, junto con uno o más estilo y patrones que puedan ser identificados. En el siguiente nivel, los subsistemas conceptuales deben ser descompuestos en componentes conceptuales, junto con uno o más patrones auxiliares. Los autores ilustran esta descomposición por medio de la figura 5.2.2.

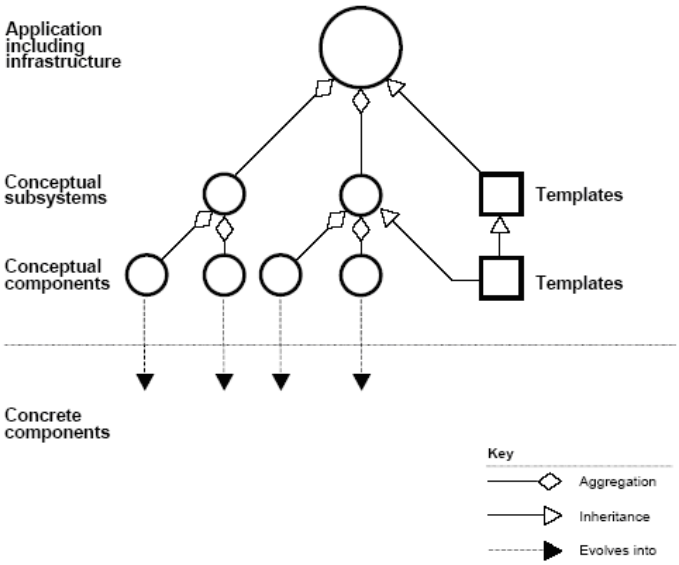


Figura 5.2.2, Descomposición del sistema.

Por medio de los componentes conceptuales, se podrán descomponer componentes concretos que representarán elementos de diseño en la arquitectura de software.

Para realizar esta descomposición y llegar a elementos de diseño, existen dos posibles caminos, los cuales son válidos y quedarán al criterio del arquitecto:

a) Definir todos los subsistemas conceptuales antes de descomponerlos y posteriormente definir todos los componentes conceptuales antes de conceptualizar cualquier componente concreto.

b) Definir un subsistema conceptual en profundidad, es decir con sus componentes conceptuales y componentes concretos, posteriormente definir otro subsistema conceptual en profundidad y así sucesivamente.

Al realizar esta descomposición de arriba hacia abajo, debemos considerar: el conocimiento del dominio del problema, la incorporación de la tecnología que es necesaria y la experiencia del equipo de arquitectos.

Cada uno de los elementos de diseño deberá conceptualizarse por medio estilos, patrones y con un conjunto de reglas determinado por los requerimientos funcionales y los atributos de calidad.

Un elemento de diseño tiene un conjunto de responsabilidades de acuerdo a su funcionalidad, las cuales deben ser divididas en grupos (traducido de) [12]. Esta división debe ser refinada y dependerá de la importancia de un atributo de calidad particular en el elemento de diseño.

El objetivo de la descomposición es producir grupos de funcionalidad con suficiente “granularidad” que permita representar una descomposición de las funcionalidades del elemento de diseño y mantener un número de grupos lógicamente manejables.

Una vez que se haya realizado la descomposición de la funcionalidad del sistema, considerando su infraestructura, la siguiente actividad será estudiarla para seleccionar uno o más estilos arquitectónicos en los cuales estará basada la arquitectura de software. Posteriormente se deberá asignar funcionalidad al estilo o estilos arquitectónicos seleccionados. Este flujo de actividades se representa por medio de la figura 5.2.3.

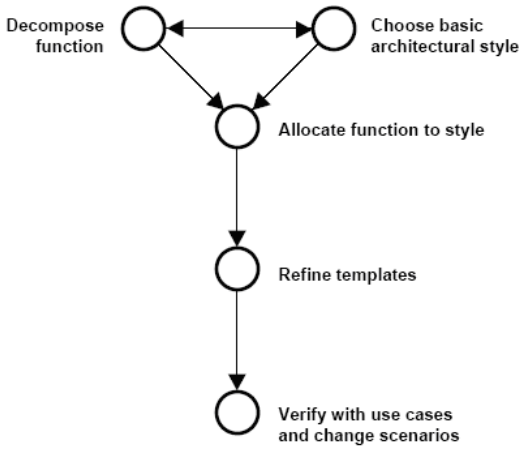


Figura 5.2.3, Selección de estilo arquitectónico.

El conjunto de elementos de diseño tiene un estilo arquitectónico, el cual debe ser modificado o adaptado con base al atributo o atributos de calidad de interés y de esta manera, lograr su objetivo.

El destinar funcionalidad al estilo, como se muestra en la figura 5.2.3, nos indica que debemos determinar la función de cada uno de los tipos de componentes identificados al escoger el estilo arquitectónico. Las interfaces conceptuales para cada elemento de diseño también deben ser identificadas.

Dentro de la iteración de estos tres pasos: dividir la funcionalidad, seleccionar el estilo arquitectónico y destinar funcionalidad al estilo, es dónde se pueden dar *tradeoffs* con respecto a varios atributos de calidad que se desee tener en la arquitectura de software.

Una vez que hayamos seleccionado el estilo arquitectónico y le hayamos asignado funcionalidad, la siguiente actividad es seleccionar uno o varios patrones arquitectónicos para un conjunto de elementos de diseño. Esto nos permitirá hacer uso de una solución ya probada y recomendada por los expertos, sin embargo, podemos adaptarlo a nuestras necesidades para lograr nuestro objetivo de diseño. La siguiente actividad es asignarle funcionalidad al patrón o patrones de diseño seleccionados, de acuerdo a los escenarios definidos.

Dentro del nivel más bajo de nuestra descomposición del sistema, podremos seleccionar un patrón de diseño para un elemento o componente en particular. De la misma manera, podremos adaptarlo a nuestras necesidades para finalmente asignarle la funcionalidad respectiva.

Es importante resaltar que la selección y uso de estilos arquitectónicos, patrones arquitectónicos y patrones de diseño, para diseñar la arquitectura de software, es nuestra recomendación para facilitar la incorporación de soluciones ya probadas, lo que nos puede ayudar a reducir tiempo de diseño, evitar errores de diseño y facilitar la comprensión del mismo. Sin embargo, la selección de dichos estilos y patrones, estará determinada por el conocimiento y experiencia del arquitecto, de tal manera que él decidirá la incorporación u omisión de alguno de ellos, de acuerdo al dominio del problema y orientadores arquitectónicos.

Por otra parte, es posible que al seleccionar los estilos o patrones, detectemos que podemos crear más de un diseño arquitectónico que puede cubrir los escenarios directos e indirectos. El arquitecto deberá evaluar si al presentarse la situación, se justifica el considerar dos arquitecturas candidatas que pueden solucionar las necesidades del sistema. De darse el caso, el paso de evaluación de la arquitectura determinará la arquitectura adecuada, después de analizarla con los métodos pertinentes.

La siguiente actividad del diseño de la arquitectura, es la documentación de la descomposición realizada del sistema, de los estilos arquitectónicos, patrones arquitectónicos y patrones de diseño a utilizar, así como de sus adecuaciones. De la misma manera, también se deben documentar todas las decisiones y/o razonamientos llevados a cabo para realizar el diseño y asignar la funcionalidad esperada.

Una vez llevada a cabo esta documentación, deberemos seleccionar y conceptualizar las vistas arquitectónicas que conformarán el diseño de nuestra arquitectura. Recordemos que una vista arquitectónica nos muestra cierta perspectiva de la arquitectura de software de acuerdo a una estructura en específico. Una estructura es el conjunto de nodos conectados por sus relaciones. Los nodos son nombrados "componentes" y las relaciones "conexiones". Esos componentes y conexiones son relacionados con otra información que conocemos como "propiedades". Las propiedades son utilizadas para distinguir entre diferentes tipos de conectores (traducido y modificado de) [3].

Las vistas que proponemos generar como parte del diseño y documentación de la arquitectura de software, son las vistas establecidas por Nord, Paulish, Soni y Hofmeister [16], además de una vista de datos adicional que proponemos. Estas vistas son:

- **Vista conceptual**, con la cual describiremos la arquitectura en términos de sus elementos de diseño principales de acuerdo su dominio y la relación a través de ellos (traducido de) [16]. Por medio de esta vista, el arquitecto modela las características funcionales del sistema.

- **Vista de módulo**, por medio de la cual describiremos la descomposición del sistema, es decir, cómo los componentes y conectores son divididos en subsistemas y módulos, mismos que están organizados en capas [16]. En esta vista se debe considerar el impacto de un cambio en software y hardware.

- **Vista de ejecución**, con la que describiremos cómo los módulos definidos en la vista de módulo son asignados a elementos de tiempo de ejecución y a componentes de hardware, así como la interacción entre dichos elementos de tiempo de ejecución, es decir, definiendo la comunicación entre ellos y asignándoles los recursos físicos (traducido de) [16]. En esta vista se deciden aspectos de: cómo asignar componentes funcionales a entidades de tiempo de ejecución, cómo manejar la comunicación, coordinación y sincronización a través de ellos y cómo se da el mapeo con el hardware.

- **Vista de código**, por medio de la cual describiremos cómo los módulos e interfases de la vista de módulos son correspondidas hacia librerías, archivos fuente y directorios, y cómo las entidades en tiempo de ejecución en la vista de ejecución son correspondidas a los archivos ejecutables (traducido de) [16]. La organización de esos archivos y cómo se encuentren organizados en los directorios, afecta la construcción del sistema.

- **Vista de datos**, por medio de la cual representaremos las diferentes entidades de datos con sus atributos, así como las relaciones entre las mismas, las cuales conformarán las tablas y campos de la base de datos que pueda contener el sistema.

Para efectos de controlar y mejorar nuestro proceso, en esta actividad registraremos como métrica el número de elementos de diseño generados, entre el número de vistas conceptualizadas. Esta métrica nos puede ayudar cuidar la proporción entre el número de elementos de diseño generados y las vistas conceptualizadas, de tal manera que no se omitan elementos de diseño en una vista.

Las diferentes vistas arquitectónicas solo deberán ser conceptualizadas en este paso de nuestro proceso, ya que la generación y descripción de las mismas se realizarán en el paso de documentación de la arquitectura de software.

Dentro del paso de diseñar la arquitectura de software, hemos visto las diferentes actividades que debemos realizar para llegar a dicho diseño, las cuales son: analizar escenarios, matriz de relación de escenarios y orientadores arquitectónicos; descomponer la funcionalidad del sistema en subsistemas conceptuales, componentes conceptuales, componentes concretos y elementos de diseño; seleccionar, en su caso adaptar, y asignar funcionalidad a estilos arquitectónicos, patrones arquitectónicos y patrones de diseño; documentar la descomposición realizada y la selección de estilos y patrones, y finalmente seleccionar y conceptualizar las vistas arquitectónicas que conformarán la arquitectura de software. Nuestro siguiente paso es verificar la arquitectura.

5.3. Verificar la arquitectura

En este paso de la elaboración de la arquitectura de la arquitectura de software, determinaremos si nuestra arquitectura cubre los requerimientos establecidos en los escenarios directos e indirectos, así como el correcto diseño de los componentes de software de acuerdo a la funcionalidad asignada. Esta verificación nos permitirá detectar si la arquitectura requiere de un mayor grado de refinamiento, o corregir elementos en su diseño, previo a su documentación y evaluación. De esta manera, la labor del paso de evaluación será detectar problemas de mayor fondo en cuanto a conflictos del logro de requerimientos no funcionales, o bien el impacto y riesgo de no incorporar u omitir ciertos estilos, patrones o componentes de software en la arquitectura.

La figura 5.3.1 muestra las actividades a detalle que sigue este paso, mostrando sus entradas, salidas herramientas o recursos de apoyo y el registro de métricas.

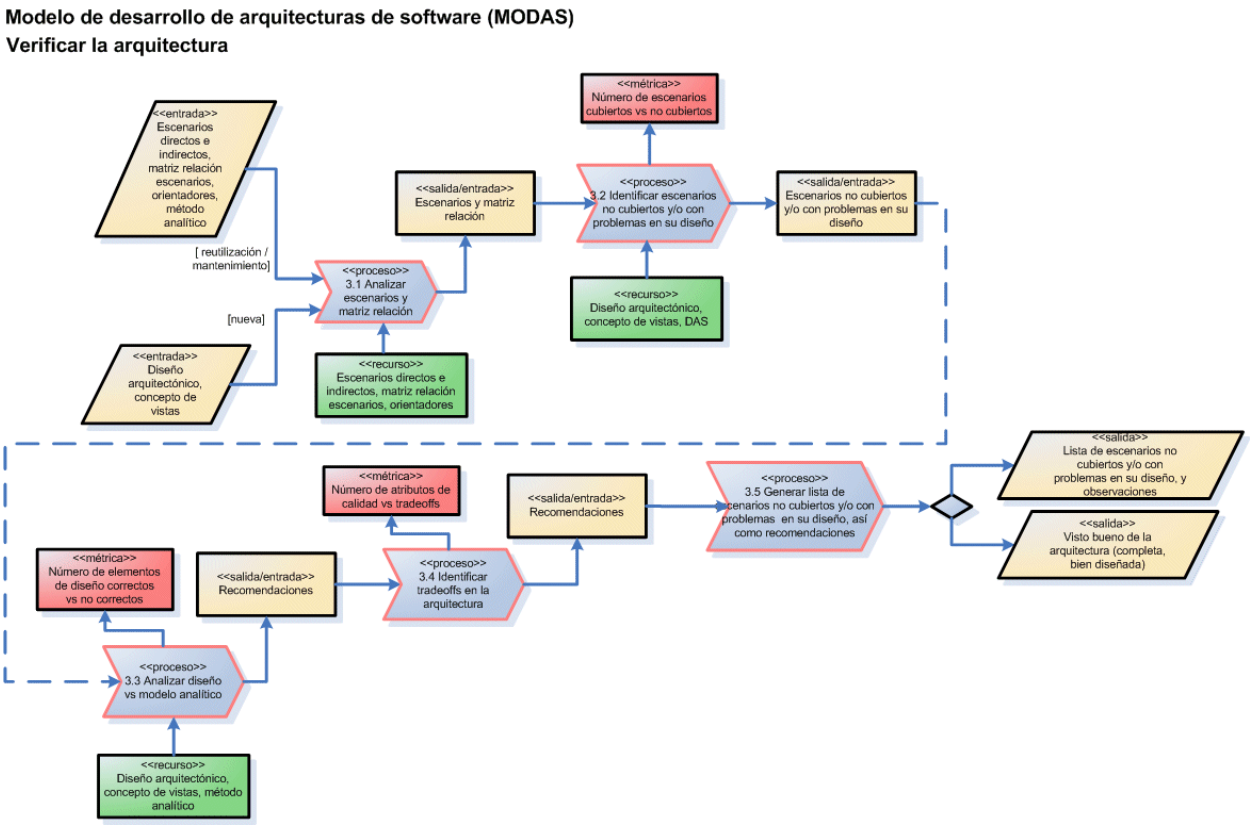


Figura 5.3.1, Paso: Verificar la arquitectura del Modelo para el Desarrollo de Arquitecturas de Software.

Los elementos a considerar para este paso del proceso son:

Elemento	Descripción
Entradas	Diseño arquitectónico y concepto de vistas, para el caso de una arquitectura nueva; o escenarios directos e indirectos, matriz de relación de escenarios, orientadores arquitectónicos y método analítico seleccionado, el caso de reutilizar una arquitectura existente o darle mantenimiento.
Salidas	Pueden ser dos de acuerdo a los resultados de la verificación: lista de escenarios no cubiertos y/o con problemas en su diseño, así como las observaciones correspondientes; o visto bueno de la arquitectura, es decir, completa y bien diseñada.
Recursos	Las mismas entradas del proceso, es decir, diseño arquitectónico, concepto de vistas, escenarios directos e indirectos, matriz de relación de escenarios, orientadores arquitectónicos y método analítico seleccionado, todos ellos para ser analizados y comparadas contra el diseño de la arquitectura. Cabe resaltar que no importando las entradas que tenga el proceso de acuerdo a su objetivo de reutilizar, mantener o construir una nueva arquitectura, debemos contar con todos los recursos listados, ya sea por la ejecución de un paso previo de nuestro modelo de proceso o porque la arquitectura existente se encuentra documentada con la definición concreta de su diseño arquitectónico.
Métricas	Número de escenarios cubiertos y bien diseñados entre número de escenarios no cubiertos o mal diseñados; número de elementos de diseño correctos contra entre número de elementos de diseño incorrectos, y finalmente número de atributos de calidad entre número de *tradeoffs* identificados.
Roles	Arquitecto.

Como primera actividad de este paso del modelo de proceso, tendremos que repasar los escenarios directos e indirectos, así como la matriz de relación de escenarios, con el objetivo de considerar cuidadosamente los requerimientos que debemos de verificar que están cubiertos por el diseño de la arquitectura.

De esta manera, nuestro siguiente actividad será analizar el diseño de la arquitectura y el concepto de vistas, además del documento de arquitectura de software, en el caso de que se trate de la reutilización de una arquitectura o mantenimiento de una arquitectura existente; todo ello con el objetivo de identificar aquellos escenarios no cubiertos por la misma, o bien que cuenten con problemas en su diseño.

En el caso de haber seleccionado un método analítico para verificar el cumplimiento de un requerimiento funcional o atributo de calidad, deberemos analizar el cumplimiento dicho requerimiento no funcional en el diseño de la arquitectura. Para ello deberemos aplicar los procedimientos, técnicas o fórmulas del método analítico en el diseño correspondiente de la arquitectura.

Es probable que observemos que esta y la siguiente actividad de nuestro proceso, se refieran a la evaluación de la arquitectura, y aunque normalmente es así, nosotros proponemos llevarlo a cabo a un nivel junto o acompañado del paso del diseño, con el objetivo de detectar problemas en el diseño de manera temprana y corregirlos durante esta fase de diseño antes de ser evaluados. Esto permite que la fase de evaluación se enfoque en problemáticas relacionadas con atributos de calidad de una manera más específica; a la justificación del impacto y riesgo de la decisión de omisión de cierto estilo, patrón o componente; o bien a la evaluación de dos arquitecturas candidatas para determinar la más adecuada.

La siguiente actividad de nuestro paso es el identificar *tradeoffs* en la arquitectura, es decir, atributos de calidad o requerimientos no funcionales que se encuentren en conflicto, pues el cumplir con uno hace que no se logre otro. Existen casos comunes como el querer lograr un buen desempeño en la arquitectura, además de un alto grado de mantenimiento. Normalmente para que un sistema se de fácil mantenimiento se requiere dividir en varios componentes de software la funcionalidad, lo que puede afectar de manera considerable el desempeño. Si se detectan casos como este, el arquitecto que realice la verificación, deberá hacer las recomendaciones pertinentes. Aunque esta actividad también se realiza en el paso de evaluación de arquitectura, en este momento permite observar *tradeoffs* sencillos de detectar y corregir, en un nivel de abstracción mayor del que se puede analizar en la evaluación.

Una vez realizadas estas actividades, deberemos generar un listado con: escenarios no cubiertos y/o con problemas en su diseño, requerimientos no cubiertos de acuerdo al modelo analítico y *tradeoffs* identificados, así como las recomendaciones necesarias y pertinentes para adecuar y corregir estos aspectos en el diseño de la arquitectura en la medida de lo posible.

A manera de conclusión, podemos decir que dentro del paso de verificar la arquitectura vimos que es necesario: analizar escenarios y matriz de relación de escenarios; identificar escenarios no cubiertos y/o con problemas en su diseño; identificar requerimientos no cubiertos de acuerdo a un modelo analítico, en el caso de que haya sido seleccionado para su verificación; identificar *tradeoffs* en la arquitectura, y finalmente generar una lista con todas las problemáticas encontradas, así como con las recomendaciones necesarias para la adecuación y corrección de la arquitectura de software.

5.4. Adecuar la arquitectura

El paso de adecuar la arquitectura de software permite realizar ajustes o adaptaciones en el diseño de la arquitectura, después de que fue ejecutado el paso de verificar la arquitectura, evaluar la arquitectura, o bien validar la arquitectura y se identificaron escenarios no cubiertos y/o con problemas en su diseño; requerimientos no cubiertos de acuerdo al modelo analítico con el cual fue evaluado; *tradeoffs* en los requerimientos no funcionales o atributos de calidad establecidos; o detección de nuevos requerimientos.

La figura 5.4.1 muestra las actividades que sigue esta actividad, especificando entradas, salidas, herramientas a utilizar y registro de métricas.

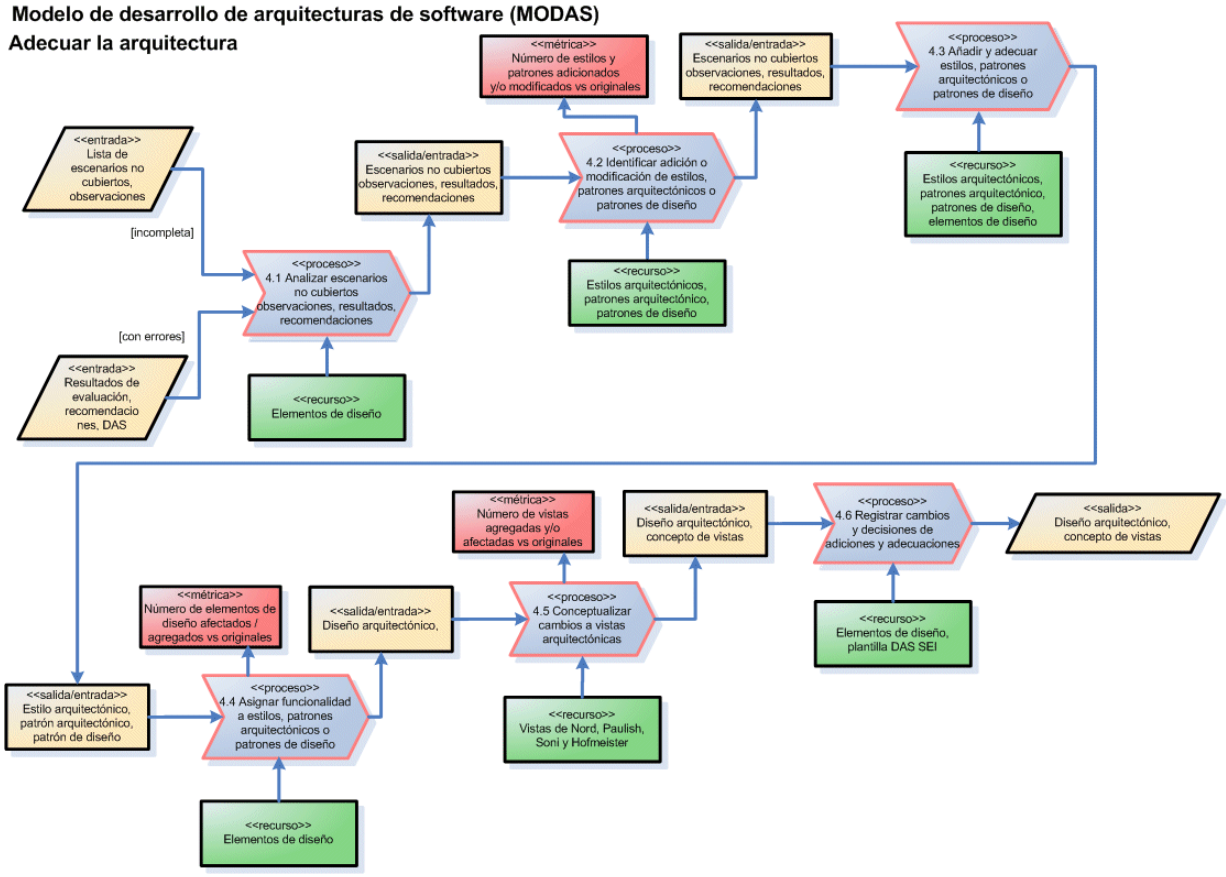


Figura 5.4.1, Paso: Adecuar la arquitectura del Modelo para el Desarrollo de Arquitecturas de Software.

Como se puede apreciar en la figura 5.4.1, esta actividad tiene dos tipos de entrada, lo cual refleja que puede provenir tres pasos diferentes: después de la verificación de la arquitectura, después de evaluación de la arquitectura, o bien después de la validación con los diferentes *stakeholders*. En los tres pasos es posible que se hayan identificado problemas en el diseño de la arquitectura a diferentes niveles. Si proviene de validación con los *stakeholder* de manera directa, indica que se tratan de cambios menores que no requieren volver a identificar los requerimientos arquitectónicos.

Los elementos a considerar para este paso del proceso son:

Elemento	Descripción
Entradas	Lista de escenarios no cubiertos o con problemas en su diseño, así como las observaciones o recomendaciones pertinentes, para el caso de realizarse después de verificación; resultados de evaluación, recomendaciones y el documento de la arquitectura de software, en el caso de que se realice después de la evaluación de la arquitectura; o bien lista de observaciones y documento de la arquitectura de software, en el caso de realizarse después de validación con los diferentes *stakeholders*.
Salidas	Diseño arquitectónico y el concepto de las vistas arquitectónicas ya ajustada o adaptadas con base a las recomendaciones realizadas.
Recursos	Elementos del diseño arquitectónico, estilos arquitectónicos, patrones arquitectónicos, patrones de diseño, vistas arquitectónicas de Nord, Paulish, Soni y Hofmeister y la plantilla de documento de arquitecturas de software del Instituto de Ingeniería de Software de la Universidad de Carnegie Mellon.
Métricas	Número estilos y patrones adicionados y/o modificados entre número de estilos y patrones originalmente seleccionados; número de elementos de diseño afectados y/o agregados entre el número de elementos originales, y el número de vistas agregadas y/o afectadas entre el número de vistas originales. Estas métricas nos ayudarán a detectar defectos inyectados en el paso de diseñar la arquitectura de software.
Roles	Arquitecto.

Como anteriormente mencionamos, este paso para adecuar la arquitectura solo se llevará a cabo si dentro de nuestro proceso para elaboración de arquitecturas de software se ha dado alguna de las siguientes situaciones:

a) Si después de llevar a cabo el proceso de verificación de la arquitectura de software, se determina que el diseño arquitectónico se encuentra incompleto o mal diseñado.

b) Si después de llevar a cabo el proceso para evaluación de arquitecturas de software, el reporte de recomendaciones indica que se deben llevar a adecuaciones a la arquitectura para lograr los objetivos deseados o alcanzar los requerimientos planteados.

c) Si después de llevar a cabo el paso la validación de la arquitectura de software con los *stakeholders* se dan observaciones que representan cambios menores a la arquitectura de software de acuerdo a sus objetivos o intereses.

La primera actividad dentro de este paso es analizar detalladamente los escenarios no cubiertos y/o con problemas en su diseño, observaciones, recomendaciones o resultados de evaluación con relación a los elementos de diseño que representan la arquitectura, dependiendo del paso dónde provenga.

Si las recomendaciones para realizar la adecuación en la arquitectura muestra que uno o más requerimientos no funcionales planteados para el sistema, no son cubiertos por la arquitectura de software, se encuentran mal diseñados o existen conflictos en sus atributos de calidad, entonces ésta debe sufrir cambios para lograr su objetivo. Los cambios deben ser analizados por el arquitecto de acuerdo al reporte de recomendaciones y decidir la mejor solución.

De esta manera, la siguiente actividad de este paso es identificar la adición o modificación de estilos arquitectónicos, patrones arquitectónicos y patrones de diseño, con base en las problemáticas anteriormente analizadas. Con ello podremos decidir si realmente añadimos y/o adecuamos dichos estilos y patrones, para posteriormente asignarles su nueva funcionalidad. Dentro de esta actividad debemos registrar la métrica: número de estilos o patrones que fueron adicionados y/o modificados, entre el número de estilos y patrones inicialmente seleccionados y/o adecuados.

La siguiente actividad será añadir y/o adecuar estilos, patrones arquitectónicos o patrones de diseño a nuestro diseño arquitectónico, de acuerdo a la identificación que realizamos de los mismos. Posteriormente deberemos asignar funcionalidad a dichos estilos y patrones. Dentro de esta actividad debemos registrar la métrica: número de elementos de diseño afectados y/o agradados entre el número de elementos de diseño inicialmente conceptualizados.

Una vez que hemos identificado, añadido y/o adecuado estilos arquitectónicos, patrones arquitectónicos y patrones de diseño, y les hemos asignado nueva funcionalidad, debemos conceptualizar cambios a las vistas arquitectónicas o bien añadir una nueva vista. Con ello, a su vez, debemos registrar la métrica del número de vistas agregadas y/o afectadas, entre el número de vistas originalmente conceptualizadas.

Las métricas que hemos registrado en el paso de adecuar la arquitectura, nos ayudarán a llevar un control sobre los cambios que realizamos en el diseño arquitectónico con base en el original, así como una medida de llevar defectos de dicho diseño, lo que a su vez nos permitiría identificar mejoras en el proceso.

Finalmente, nuestra última actividad de este paso de nuestro modelo de proceso, será registrar en el documento de arquitectura de software todos los cambios o adiciones realizadas adecuaciones en el diseño arquitectónico, así como las decisiones y razonamientos que dieron pauta a dichos cambios.

De acuerdo a lo que comenta J. Bosh y P. Molin en su artículo *Software Architecture Desing: Evaluation and Transformation* [10], así como Bosh en su libro *Design and Use of Software Architectures, Adopting and Evolving a Product-Line Approach* [5], existen cinco categorías de adecuaciones que puede sufrir una arquitectura de software y que son organizadas de manera descendente de acuerdo a su impacto en la misma. Estas categorías son (traducido y modificado de) [10]:

- **Aplicar un estilo arquitectónico**. El estilo más apropiado para el sistema depende principalmente de sus requerimientos no funcionales. Transformar un arquitectura aplicando un estilo arquitectónico, da como resultado una reorganización completa de la arquitectura. Los estilos de arquitectura generalmente no sufren cambios, pero es posible aplicar diferentes estilos en un subsistema.

- **Aplicar un patrón arquitectónico**. Un patrón arquitectónico es diferente a un estilo arquitectónico, pues no es predominante y afecta completamente a la arquitectura o al menos una parte importante. El patrón arquitectónico generalmente impone una regla en la arquitectura que especifica cómo trabajará el sistema con un aspecto de su funcionamiento en particular, ejemplo: concurrencia, persistencia.

- **Aplicar un patrón de diseño**. Esta consideración tiene menos impacto en la transformación. Aplicar un patrón de diseño generalmente afecta solo a un número limitado de clases en la arquitectura. En consecuencia, una clase puede estar envuelta en múltiples patrones de diseño sin crear inconsistencias.

- **Convertir los requerimientos no funcionales a una solución funcionalidad**, que consecuentemente extiende la arquitectura con funcionalidad no relacionada al dominio del problema, pero que es utilizada para cubrir el requerimiento no funcional.

- **Distribuir requerimientos**. Un requerimiento no funcional del sistema puede ser dividido en subsistemas o componentes que toma el sistema. De esta manera, un requerimiento no funcional determinado es distribuido sobre "n" componentes. Un segundo enfoque puede ser distribuir los requerimientos dividiendo los requerimientos no funcionales en dos o más requerimientos no funcionales relacionados.

El arquitecto deberá analizar las recomendaciones realizadas en los pasos previos y determinar una o más soluciones de acuerdo a estas categorías, buscando cumplir con los requerimientos establecidos para el sistema y los atributos de calidad considerados por los intereses de los *stakeholders* en la medida de las posibilidades.

Dentro de este paso hemos vistos las actividades que debemos llevar a cabo para adecuar una arquitectura de software, las cuales son: analizar los escenarios no cubiertos en el diseño arquitectónico y/o con problemáticas en su diseño, observaciones, recomendaciones y resultados de la evaluación, en su caso; identificar la adición o modificación de estilos arquitectónicos, patrones arquitectónicos y/o patrones de diseño; asignar nueva funcionalidad a dichos estilos y patrones adicionados o modificados; conceptualizar cambios a las vistas arquitectónicas, y finalmente registrar cambios realizados y decisiones o justificaciones de adición o modificación de componentes de software.

5.5. Documentar la arquitectura

Este paso de nuestro modelo de proceso nos permite representar gráfica y textualmente la arquitectura de software que hemos diseñado, además de plasmar los requerimientos que cubre, los *viewpoints*, las vistas que contendrá, los orientadores arquitectónicos que dan la directriz de la arquitectura, el uso de estilos arquitectónicos, patrones arquitectónicos y/o patrones de diseño, decisiones o justificaciones de diseño, intereses de los diferentes *stakeholders*, entre otros aspectos.

La figura 5.5.1 muestra las actividades que sigue la documentación de la arquitectura, especificando sus entradas, salidas, herramientas en las que se apoyará y registro de métricas.

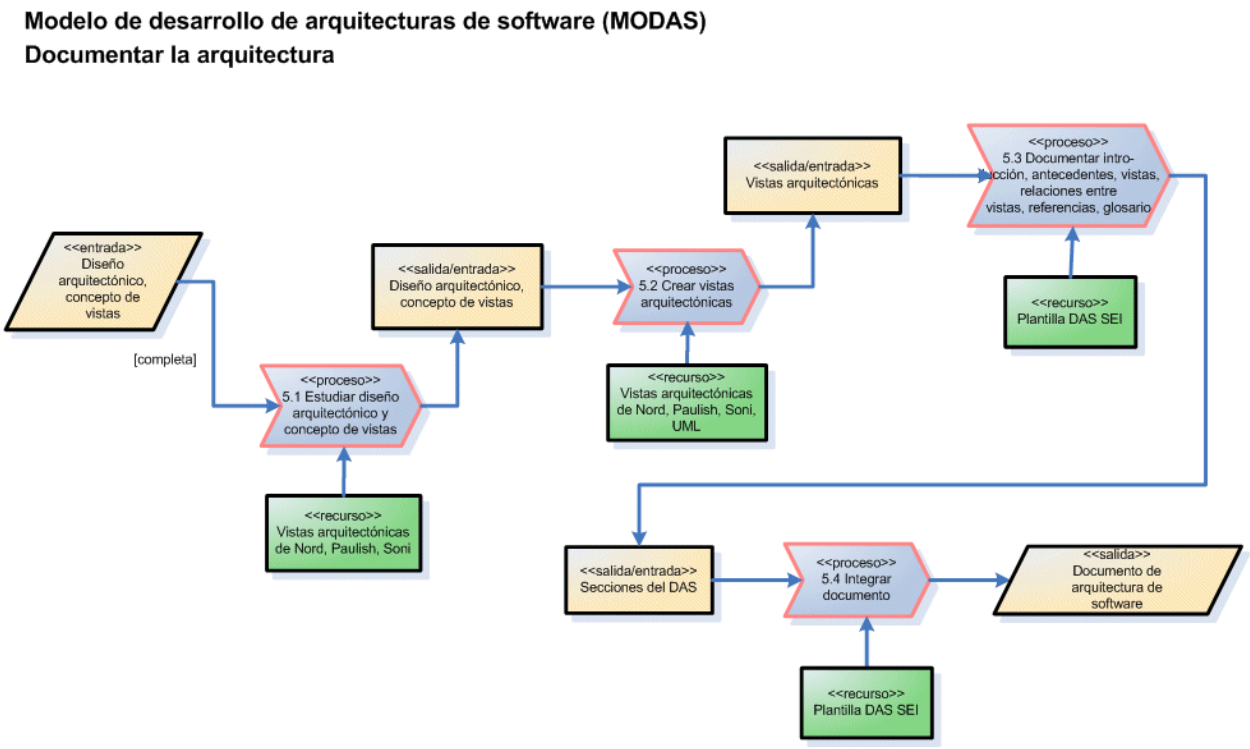


Figura 5.5.1, Paso: Documentar la arquitectura de software del Modelo para el Desarrollo de Arquitecturas de Software.

Los elementos a considerar para este paso del proceso son:

Elemento	Descripción
Entradas	Diseño arquitectónico y la concepción de las vistas.
Salidas	Documento de arquitectura de software.
Recursos	Vistas arquitectónicas de Vistas de Nord, Paulish, Soni y Hofmeister, UML y la plantilla de documento de arquitectura de software del Instituto de Ingeniería de Software de la Universidad de Carnegie Mellon.
Roles	Arquitecto y el documentador.

La documentación de la arquitectura de software debe ser completa y navegable. Debe ser presentada en una representación genérica que haga referencia a varios subsistemas que están conjuntamente conectados, así como mostrando cómo dichos subsistemas están conectados internamente (traducido y modificado de) [3]. Toda la documentación de la arquitectura de software debe ser publicada y disponible a todos los *stakeholders* involucrados.

La infraestructura es el conjunto de mecanismos para la comunicación y coordinación de la arquitectura. Esta infraestructura debe ser documentada, de la misma manera, como una parte integral de la arquitectura.

Como primer paso para documentar la arquitectura de software debemos estudiar el diseño arquitectónico al que se llegó y la concepción de las vistas de acuerdo a las propuestas por Nord, Paulish, Soni y Hofmeister documentar las diferentes perspectivas de la arquitectura [15]. Después de ello generaremos y documentaremos cada de las vistas, las cuales son: vista conceptual, vista de módulo, vista de ejecución, vista de código, y adicionalmente una vista de datos.

Estos mismos autores recomiendan el uso de UML como una manera efectiva de modelar las vistas arquitectónicas, así como la construcción de ciertos diagramas UML que ayudan a representar cada una de las vistas, lo cual retomaremos para generar la representación gráfica de nuestro diseño de arquitectura de software. A continuación describiremos los diagramas UML que recomiendan los autores.

Para la vista conceptual, podemos representar los componentes, puertos y conectores, como clases estereotipadas. De esta manera, podemos construir:

- Diagramas de clase de UML para mostrar la configuración estática.
- Declaraciones del protocolo ROOM para mostrar los protocolos que utilizan los puertos.
- Diagramas de secuencia de UML para mostrar la secuencia de interacciones a través de un grupo de componentes.

En la vista de módulo podemos representar los módulos con una clase estereotipada, y los subsistemas y capas con paquetes estereotipados, por lo que podemos crear:

- Tablas para describir el mapeo entre la vista conceptual y de módulo.
- Diagramas de clase de UML para mostrar subsistemas y descomposición de módulos.
- Diagramas de clase de UML para mostrar las dependencias de uso entre los módulos.
- Diagramas de clase de UML para mostrar las dependencias a través de las capas y la asignación de módulos a las capas.

Para la vista de ejecución podemos representar las entidades en tiempo de ejecución por medio de clases estereotipadas, y sus rutas de comunicación como asociaciones. De esta manera, podemos crear:

- Diagramas de clase de UML para mostrar la configuración de la ejecución del sistema.
- Diagramas de *deployment* de UML para mostrar cómo la configuración de la ejecución es correspondida hacia el hardware.
- Diagramas de secuencia de UML para mostrar el comportamiento dinámico de la configuración o la transición entre las configuraciones.

- Diagramas de estado o de secuencia de UML para mostrar el protocolo de un camino de comunicación.

Para la vista de código podemos representar los componentes fuentes y binarios como componentes estereotipados. Las librerías ejecutables y grupos de código (paquetes) son elementos que existen en UML para su representación. Las relaciones de generación, importación, compilación y ligas, son mostradas como dependencias estereotipadas. De esta manera, podemos construir:

- Ruta de dependencias de UML o tablas para describir el mapeo entre elementos de la vista de módulos y los componentes fuente.
- Dependencias de instancia de UML o tablas para describir el mapeo entre las entidades de tiempo de ejecución y los archivos ejecutables.
- Diagramas de componente de UML o tablas para describir los componentes en la vista de código de la arquitectura, su organización y sus dependencias.

Para el caso de la vista de datos, proponemos crear un diagrama entidad relación por medio de la herramienta CASE Erwin, misma que hace uso de la notación IDEF1x para el modelado de entidades, atributos de las entidades y relaciones entre dichas entidades.

Una vez que tenemos todas las consideraciones para documentar la arquitectura de software, así como las diferentes vistas de Nord, Paulish, Soni y Hofmeister, como siguiente actividad, debemos genera el documento que describa todos nuestros elementos. Para crear el documento de la arquitectura de software, nos basaremos en una plantilla propuesta por el Instituto de Ingeniería de Software de la Universidad Carnegie Mellon. Esta plantilla de documento de arquitectura de software considera los siguientes elementos:

a) Introducción
   a. Propósito
   b. A quién va dirigido
   c. Alcance del documento de arquitectura de software
      i. Requerimientos significativos
   d. Organización del documento de arquitectura de software
   e. Representación del *stakeholder* de acuerdo a sus intereses en la arquitectura de software
   f. Definiciones de *viewpoint*
      i. Nombre del *viewpoint*
      ii. Resumen del *viewpoint*
      iii. *Stakeholder* y la orientación de sus intereses en el *viewpoint*
      iv. Elementos, relaciones, propiedades y restricciones que deben obedecer las vistas que conforman el *viewpoint*
      v. Lenguaje para modelar las vistas arquitectónicas
      vi. Reglas de evaluación y consistencia que aplica a las vistas del *viewpoint*
      vii. Fuente de dónde se basa el *viewpoint*
b) Antecedentes de la arquitectura
   a. Antecedentes del problema
      i. Visión general del sistema
      ii. Objetivo y contexto
   b. Panorama de la solución
      i. Enfoque arquitectónico (razonamiento de las decisiones principales de diseño)
      ii. Análisis de resultados (de cualquier análisis cuantitativo o cualitativo ejecutado en la evaluación de la arquitectura)
      iii. Cobertura de requerimientos (Descripción de escenarios y seguimiento)
c) Vistas
   a. Nombre de la vista
      i. Descripción de la vista
      ii. Visión general del *view packet*
      iii. Antecedentes de la arquitectura aplicable a la vista
      iv. Mecanismos de variabilidad en la vista de la arquitectura
      v. View packet

1. Presentación primaria
2. Catálogo de elementos
    a. Elementos
    b. Relaciones
    c. Interfases
    d. Comportamientos
    e. Restricciones
3. Diagrama de contexto (alcance del *view packet* en el sistema)
4. Mecanismos de variabilidad en el *view packet* de la vista
5. Antecedentes de la arquitectura aplicable al *view packet*
6. Relación con otros *view packet*

d) Relación entre vistas
    a. Relaciones generales entre las vistas
    b. Relaciones vista a vista
e) Materiales referenciados
f) Glosario de términos

Algunos de los puntos que considera esta plantilla para el documento de arquitectura de software serán opcionales de acuerdo al alcance o representación de las vistas arquitectónicas.

Las secciones de nuestro documento de arquitectura de software, pueden documentarse de manera independiente, sin embargo deberá existir concordancia, coherencia y homologación en la descripción de los diferentes elementos del diseño, decisiones y justificación de diseño, ya que varios de los puntos de dicho documento se basan o sustentan en lo descrito en un punto anterior.

Una vez que contamos con la documentación de las diferentes vistas arquitectónicas, así como el resto de las secciones muestra la plantilla, debemos integrar el documento de arquitectura de software de manera formal, mismo que representará nuestra salida para este paso de nuestro modelo.

Con lo anterior, podemos concluir que en el paso de documentar la arquitectura hemos visto algunas consideraciones para documentar la arquitectura de software, los diagramas UML que construiremos para conformar la vista conceptual, la vista de módulos, la vista de ejecución, la vista de código y la vista de datos, así como una plantilla en la cual nos basaremos para generar el documento de arquitectura de software.

5.6. Evaluar la arquitectura

Una vez que contamos con la documentación de la arquitectura de software, debemos evaluar su diseño y definición. Esta evaluación estará determinada de acuerdo a dos objetivos:

a) Evaluar si cubre los atributos de calidad considerados por los diferentes *stakeholders* o bien, si existe algún *tradeoff* en dichos atributos de calidad que puedan impedir el logro de los objetivos o el alcance de uno o varios requerimientos.
b) Establecer el impacto o las desventajas de eliminar o no considerar dentro de la arquitectura el uso de ciertos componentes, patrones de diseño o estilos arquitectónicos.

De acuerdo a uno de estos objetivos, llevaremos a cabo el proceso para evaluación de arquitecturas de software propuesto por Omar Gómez [21]. Este proceso permite determinar si la arquitectura de software cumple con los requerimientos no funcionales o atributos de calidad, si existe algún conflicto entre dichos atributos de calidad, o bien el impacto y riesgo que puede traer como consecuencia la ausencia de ciertos estilos, patrones o componentes en la arquitectura de software.

El proceso de evaluación que se propone, utiliza la identificación de patrones de diseño para asegurar que se logren conectar o relacionar de manera transparente atributos de calidad tales como: facilidad de modificación, rendimiento, seguridad, entre otros. Además de la identificación de patrones de diseño, el proceso hace uso de dos métodos de evaluación: ALMA (Architecture-Level Modifiability Analysis) y SNA (Survivability Network Analysis), dependiendo del atributo de calidad a evaluar. Sin embargo este proceso de evaluación es lo suficientemente flexible, ya que permite hacer uso de otros métodos de evaluación de arquitecturas de software [21].

La entrada del proceso para la evaluación de la arquitectura es el documento de arquitectura de software, mientras que las salidas del mismo son los resultados de la evaluación, y recomendaciones de mejora.

Los recursos que se utilizan para apoyar este paso son: método ALMA (Architecture-Level Modifiability Analysis) y SNA (Survivability Network Analysis).

5.7. Validar la arquitectura

Nuestro último paso del modelo de proceso es la validación de la arquitectura de software con los diferentes *stakeholders*. El objetivo es determinar que la arquitectura de software se encuentre acorde a los requerimientos establecidos y cubra todas las necesidades e intereses de los *stakeholders*. Si después de validarla identificamos cambios menores, podremos ejecutar el paso de adecuación, documentación y validación nuevamente para hacer las modificaciones correspondientes. Sin embargo, si identificamos cambios mayores, como la incorporación de nuevos requerimientos o no correctamente clarificados con anterioridad, podremos ejecutarse todo el proceso nuevamente para agregar al diseño de nuestra arquitectura los componentes correspondientes.

La figura 5.7.1 muestra las actividades a detalle que sigue este paso, mostrando sus entradas, salidas herramientas o recursos de apoyo y el registro de métricas.

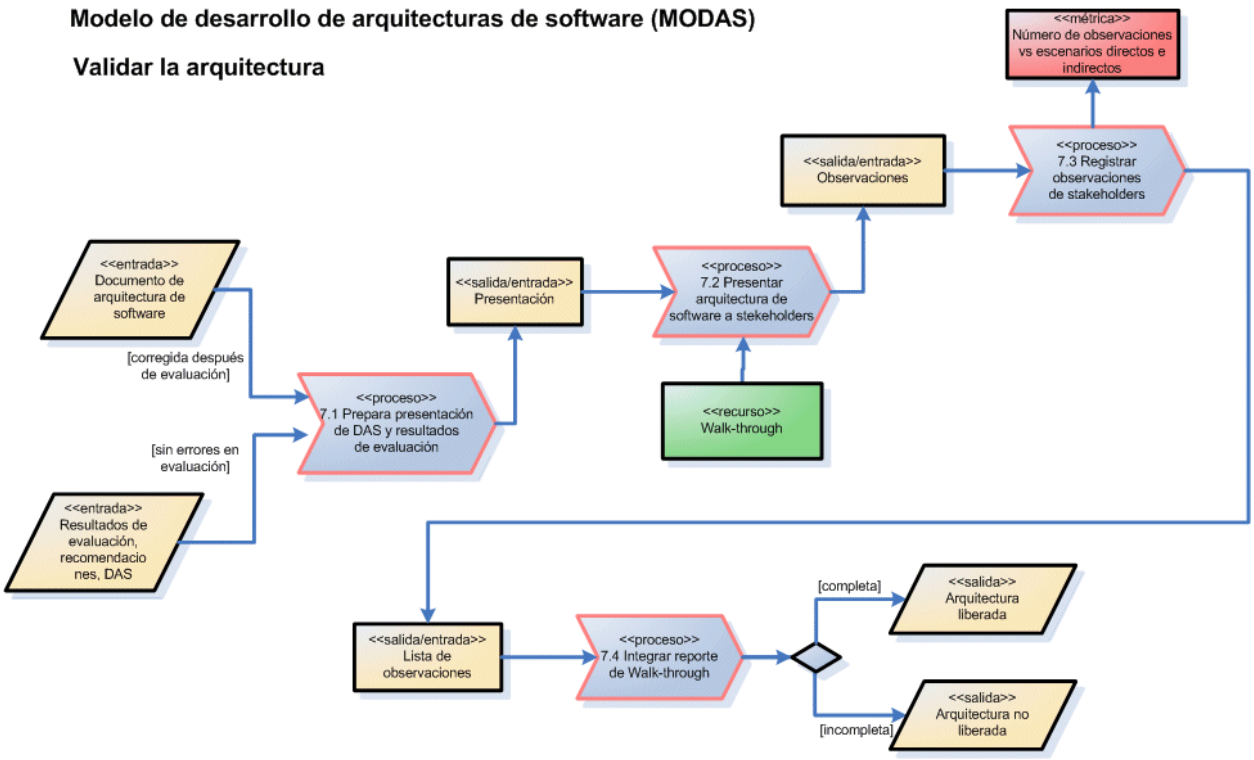


Figura 5.7.1, Paso: Validar la arquitectura del Modelo para el Desarrollo de Arquitecturas de Software.

Los elementos a considerar para este paso del proceso son:

Elemento	Descripción
Entradas	Documento de arquitectura de software y resultados de evaluación que no requieren adecuación, cuando se ejecuta después del paso de evaluar la arquitectura, o bien el documento de arquitectura de software una vez que fue evaluada, adecuada, verificada y documentada nuevamente la arquitectura.
Salidas	Pueden ser dos de acuerdo a los resultados de la validación: arquitectura liberada o arquitectura no liberada con solicitud de cambios.
Recursos	*Walk-through*.
Métricas	Número de observaciones entre número de escenarios directos e indirectos de la arquitectura de software.
Roles	Arquitecto y los diferentes *stakeholders*.

De esta manera, una vez que hemos llevado a cabo el proceso para evaluación de arquitecturas de software, y el reporte emitido muestra que la arquitectura cumple con todos los requerimientos y atributos de calidad establecidos, deberemos llevar a cabo el proceso de validación de la arquitectura, en dónde necesitaremos el documento de arquitectura de software como entrada.

Para llevar a cabo esta validación haremos uso de la técnica de *walk-through*. Un *walk-through* es un proceso estructurado por un equipo para llevar a cabo la revisión de un artefacto de software, el cual sigue el formato de una presentación con los arquitectos, desarrolladores y *stakeholders*, como: cliente, personal de marketing, sistemas y pruebas[3]. Aunque es estructurado, este proceso no es tan formal como otros métodos de revisión, como las inspecciones.

Por lo tanto, la primera actividad de este paso es preparar la presentación del documento de arquitectura de software y de los resultados de la evaluación. Posteriormente, el arquitecto presenta el documento de arquitectura de software a los distintos *stakeholders* y recorre sus diferentes vistas, mientras que la audiencia identifica problemas y hace preguntas, mismas que se van registrando para su control.

Cuando los arquitectos y *stakeholders* se reúnen esté método puede ser efectivo para descubrir omisiones, confusiones y/o diferencias entre lo que requiere el cliente y lo que cubre la arquitectura. Un *walk-through* puede ser utilizado también para capacitar a los *stakeholders* correspondientes sobre el diseño de la arquitectura.

Una vez finalizado el *walk-through*, el equipo que presente la arquitectura deberá integrar y emitir un reporte en dónde se especifiquen las observaciones de los *stakeholders* y decidir si dicha arquitectura de software requiere adecuaciones. En el caso de que la arquitectura requiera cambios menores, se deberá ejecutar el paso de adecuación de arquitectura, documentación y validación nuevamente para hacer las modificaciones correspondientes. En el caso de que la arquitectura deba sufrir cambios significativos que impactan el diseño de uno o más nuevos componentes, como la incorporación de nuevos requerimientos, debemos ejecutar todo el proceso nuevamente para agregar al diseño de nuestra arquitectura los elementos correspondientes. Para ambos casos la salida del *walk-through* será el documento de la arquitectura de software y el reporte de observaciones.

De esta manera, nuestro proceso se ejecutará hasta que el diseño de la arquitectura de software se encuentre completamente validado por los diferentes *stakeholders* y el documento correspondiente a dicha arquitectura de software se encuentre liberado.

[3] De acuerdo a Watts S. Humphrey, A Discipline for Software Engineering, SEI Series in Software Engineering, Addison-Wesley, 2004

Siendo la validación de la arquitectura de software nuestro último paso del proceso propuesto, hemos finalizado con la descripción de nuestro modelo. Vimos cómo debemos identificar los requerimientos arquitectónicos; cómo diseñar la arquitectura de software con base a dichos requerimientos; qué actividades realizaremos para verificar que una arquitectura que está siendo diseñada por primera vez, reutilizada con una arquitectura existente, o bien se le dará mantenimiento cuando esté en operación, cumpla con los requerimientos arquitectónicos y no existan problemas con su diseño; qué actividades debemos llevar a cabo para adecuar una arquitectura de software; cómo debemos documentar una arquitectura y qué elementos debe cubrir dicha documentación; qué método de evaluación de arquitectura llevaremos a cabo, y finalmente, cómo validaremos con los diferentes *stakeholders* que la arquitectura cumple con los distintos requerimientos funcionales y atributos de calidad.

6. Discusión del modelo para el desarrollo de arquitecturas de software

En esta sección del presente documento, discutiremos algunas desventajas, limitaciones y ventajas que hemos detectado en el modelo de desarrollo de arquitectura de software propuesto. Además, daremos algunas recomendaciones para su utilización, y concluiremos el alcance que tiene el mismo, así como los puntos de mejora a los que es susceptible.

6.1. Desventajas que presenta el modelo

Nuestro modelo de proceso para el desarrollo de arquitecturas de software, presenta algunas desventajas que pueden encontrarse al momento de su ejecución. Básicamente, hemos detectado tres desventajas principales.

La primera de ellas se presenta al reutilizar o mantener una arquitectura. Cuando deseemos reutilizar una arquitectura de software existente para el cumplimiento de los requerimientos arquitectónicos, o bien, deseemos dar mantenimiento a una arquitectura en operación, el proceso considera que existe un documento de arquitectura de software previo que contiene los elementos analizados en el paso de documentar la arquitectura de nuestro modelo de proceso. Se toma en cuenta este documento de arquitectura de software, ya que el paso de verificar la arquitectura requiere del mismo para determinar si la arquitectura que está siendo considerada, cumple con los requerimientos funcionales y no funcionales establecidos. Al no contar con este documento, el paso de verificar la arquitectura no tendrá contra qué elementos comparar la solicitud de requerimientos arquitectónicos. Para ello, se deberán definir otras actividades o mecanismos que permitan dar una alternativa al presentarse este caso.

Otras de las desventajas que presenta el modelo, es que dentro del paso de diseñar la arquitectura de software, no se describen específicamente, ni recomiendan el conjunto de estilos arquitectónicos, patrones arquitectónicos y patrones de diseño existentes en la literatura para ser utilizados por el arquitecto. Esta actividad deberá ser realizada de manera independiente y requiere de un esfuerzo adicional por parte del arquitecto para documentarse.

Finalmente, una desventaja más del modelo del proceso, es que no describe los métodos analíticos existentes en la literatura, el cual permite verificar cuantitativamente el cumplimiento de un requerimiento no funcional, y que recomendamos su utilización en el paso obtener los requerimientos arquitectónicos para la selección del mismo, y en el paso de verificar la arquitectura de software, para aplicar dicho método. De la misma manera, esto representa un esfuerzo adicional por parte del arquitecto para documentarse, seleccionar, y aplicar el método más adecuado a los atributos de calidad establecidos.

6.2. Limitaciones que presenta el modelo

El modelo de proceso de desarrollo de arquitecturas de software también presenta algunas limitaciones con su definición. Principalmente hemos detectado tres limitaciones.

La primera limitación del modelo, es que no considera actividades o recomendaciones en el caso de que se desee reutilizar una arquitectura existente, o bien dar mantenimiento a una arquitectura en operación, y que no se cuente con el documento de arquitectura de software para realizar la verificación y adecuación de la misma. El proceso da por hecho que este documento existe y se encuentra listo para utilizarse como recurso.

Otra limitación importante, es que el proceso no considera pasos y actividades específicas para la generación u obtención de arquitecturas de software de sistemas legados, ya que para este caso se deben de llevar a cabo mecanismos de ingeniería inversa, los cuales hacen uso de técnicas y herramientas especializadas.

Por otra parte, podemos decir que el modelo de proceso que proponemos, podrá ser aplicado de manera concreta para generar arquitecturas de sistemas de software en general, que no requieran el diseño de elementos de hardware para llevar a cabo las funciones del sistema.

Finalmente, el modelo de proceso de desarrollo de arquitecturas de software, no considera el control de cambios para las diferentes versiones de diseño arquitectónico que puedan ser realizados con las distintas verificaciones, evaluaciones y validaciones llevadas a cabo. Para ello, consideramos que se haga uso del proceso de administración de la configuración definido por la organización que ejecute nuestro modelo de proceso.

6.3. Ventajas que presenta el modelo

Por otra parte, nuestro modelo de proceso de desarrollo de arquitecturas de software, presenta algunas importantes ventajas que resulta importante mencionar.

La primara ventaja de nuestro modelo, es que muestra el orden concreto para la ejecución del proceso, así como la especificación detallada de las actividades a llevar a cabo en cada paso, detallando el flujo que pueden seguir dependiendo del objetivo o intención de la ejecución del mismo, es decir, diseñar una nueva arquitectura, reutilizar una arquitectura existente o dar mantenimiento a una arquitectura en operación.

Otra ventaja que presenta el modelo, es que este orden concreto para la ejecución del proceso, se muestra de manera tanto gráfica como textual, lo que facilita la comprensión del mismo al seguirlo visualmente, y apoyándose del texto para reafirmar y detallar las actividades a seguir en cada uno de los pasos.

Por otra parte, el modelo de proceso especifica elementos importantes que comúnmente no detallan otros modelos para la elaboración de arquitecturas de software. Estos elementos son:

- Entradas y salidas del proceso general, así como entradas y salidas de los pasos internos.
- Recursos o herramientas a utilizar dentro de los pasos internos del proceso.
- Registro de diferentes métricas en varios de los pasos del proceso, para el control y el mejoramiento del mismo.

6.4. Recomendaciones de utilización del modelo

Con el objetivo de maximizar la eficiencia de la operación del proceso de desarrollo de arquitecturas de software propuesto, debemos considerar algunas recomendaciones para su utilización, mismas que deben ser planificadas antes de la implementación y puesta en marcha del mismo.

La primera recomendación es que el arquitecto de software debe documentarse en profundidad sobre los recursos o herramientas propuestas que apoyan las actividades de cada una de las actividades del proceso. Las herramientas que recomendamos deben ser analizadas antes de la ejecución, son:

- Software Quality Function Deployment [18] [19] [20].
- ISO 9126-1 [7].
- Método de diseño basado en arquitecturas [12].
- Estilos arquitectónicos [2] [5].
- Patrones arquitectónicos [5] [6] [22] [32].
- Patrones de diseño [6] [22] [32].
- Métodos analíticos [11].
- Proceso de Evaluación para Arquitecturas de Software Usadas en el Sector Empresarial [21].
- Método Survivability Network Analysis [21].
- Método Architecture-Level Modifiability Analysis [21].

Es importante recalcar que el arquitecto de software debe conocer y dominar los conceptos esenciales de arquitecturas de software; estar relacionado con alguna de las taxonomías de atributos de calidad; conocer los estilos arquitectónicos principales, y manejar herramientas de modelado de artefactos de software.

Otra importante recomendación que hacemos para el arquitecto de software, es que debe conocer las ventajas, desventajas y particularidades entre los principales lenguajes de programación y tecnologías disponibles, así como el conocimiento del hardware y sus capacidades, ya que es conveniente que el arquitecto domine la mayor cantidad de tecnologías de software para ser capaz de ofrecer las mejores recomendaciones tecnológicas para el diseño arquitectónico y que otorguen beneficio al proyecto. Esto también es importante ya que las características que definen los atributos de calidad del sistema, como son: el desempeño, robustez, portabilidad, facilidad de mantenimiento, flexibilidad, escalabilidad, entre otros, dependen en gran en medida de las decisiones de diseño que tome el arquitecto.

6.5. Alcance y mejoras

El modelo de proceso para el desarrollo de arquitecturas de software considera como alcance:

- La identificación de requerimientos arquitectónicos.
- El diseño de arquitectura de software.
- La verificación de la arquitectura de software en relación a los requerimientos arquitectónicos planteados.
- La adecuación de la arquitectura de software en el caso de detectarse inconsistencias en dicha arquitectura.
- La documentación de la arquitectura de software.
- La evaluación de la arquitectura de software, para determina el cumplimiento o conflicto en los atributos de calidad determinados para la arquitectura, así como el impacto y riesgo que puede traer como consecuencia la ausencia de ciertos estilos, patrones o componentes en dicha arquitectura.
- La validación de la arquitectura de software con los diferentes *stakeholders* para determinar el cumplimiento de todos los requerimientos y objetivos planteados.
- La especificación y descripción de las actividades a detalle que se deben seguir en cada uno de los pasos del proceso de desarrollo de la arquitectura de software.
- La especificación de las entradas y salidas en cada una de las actividades llevadas a cabo el pasos del proceso de desarrollo de la arquitectura de software, así como en cada uno de los pasos de dicho proceso.
- El uso de diferentes herramientas para apoyar las actividades de los distintos pasos que se deben de llevar para el desarrollo de la arquitectura de software.
- El registro de ciertas métricas en diferentes actividades realizadas durante el proceso de desarrollo de la arquitectura de software, para permitir el control y proponer mejoras en dicho proceso.

Este modelo de proceso puede ser mejorado en varios aspectos. Dentro de los más importantes que podemos mencionar, se encuentran:

- Expresar el modelo como un proceso formal que incluya las especificaciones de guías, *scripts* y *checklists* que permitan al arquitecto de software contar con recurso fácil de manejar y seguir en su trabajo diario.

- Proponer y modificar el proceso para incorporar actividades que permitan manejar la reutilización de una arquitectura existente, o bien, el mantenimiento de una arquitectura en operación, cuando no existe documento de arquitecturas de software.

- Incorporar en la descripción textual del proceso, la descripción de los diferentes estilos arquitectónicos, patrones arquitectónicos, patrones de diseño y métodos analíticos existentes en la literatura, con el objetivo de dar al arquitecto de software como posibilidad, el analizar y seleccionar los más adecuados para la propuesta de solución del diseño de la arquitectura de software.

- Proponer pasos, actividades, entradas, salidas y registro de métricas, cuando se debe construir u obtener una arquitectura de software de un sistema legado.

De esta manera, en la presente sección hemos discutido algunas desventajas que presenta el modelo para el desarrollo de arquitecturas de software, algunas limitaciones que hemos identificado en el mismo, algunas ventajas que sabemos tiene respecto a otros modelos, algunas recomendaciones para la utilización del modelo de proceso, y el alcance que tiene, así como alguna de las mejoras que puede ser susceptible.

7. Conclusiones y trabajo futuro

A lo largo del presente documento, hemos expuesto la importancia que tiene la elaboración de una arquitectura de software para la necesidad actual del desarrollo de sistemas de gran escala y cada vez más complejos, así como la ausencia de un proceso que especifique a detalle las actividades que se deben seguir para llegar al correcto diseño de dicha arquitectura. Los modelos actuales que existen, y que hemos presentado como trabajo previo, no muestran de una manera rápidamente identificable, la secuencia de actividades que se deben llevar a cabo en cada uno de los pasos que proponen. Por otro parte, estos modelos no especifican las entradas y salidas que tendrán, tanto lo pasos del proceso, como las actividades internas que se llevan a cabo. De la misma manera, no son muy claros en la especificación de las herramientas que pueden servir de apoyo para llevar a cabo las actividades planteadas, y ninguno considera el registro de métricas que permitan controlar y mejorar el proceso.

Considerando esta importancia que hoy en día tiene el desarrollo de una arquitectura de software, y la necesidad de contar con un proceso sencillo de seguir para llevar a cabo su elaboración, hemos propuesto en el presente documento un modelo de proceso para el desarrollo de arquitectura de software, el cual contiene como elementos:

- Entradas y salidas del proceso general.
- Definición de los pasos internos de cada uno de los pasos generales del proceso.
- Detalle de entradas y salidas de las actividades de los pasos del proceso.
- Uso de herramientas que apoyan cada pasos del proceso.
- Incorporación de métricas que ayudan a controlar y mejorar el proceso.

Este modelo de proceso permite desarrollar una nueva arquitectura de software, reutilizar una arquitectura de software existente para cubrir los requerimientos arquitectónicos de un nuevo sistema, o bien, dar mantenimiento a una arquitectura que se encuentre en operación.

Los pasos generales que considera el modelo de proceso de desarrollo de arquitecturas de software son:

1. Identificación de requerimientos arquitectónicos mediante el documento de especificación de requerimientos, o definición de nuevos requerimientos de software.

2. Diseño de la arquitectura, en dónde el sistema se descompone en subsistemas, componentes y elementos de diseño para asignar la funcionalidad correspondiente, misma que representará nuestro diseño arquitectónico.

3. Verificación de la arquitectura, la cual determinará la completes de los requerimientos en el diseño, así como el correcto diseño de los componentes de software.

4. Adecuación de la arquitectura una vez que fue verificada, evaluada o validada y se identificó que no cubre los requerimientos establecidos, o bien se encuentran mal diseñados.

5. Documentación de la arquitectura de software, por medio del cual se describe de manera gráfica y textual la arquitectura.

6. Evaluación de la arquitectura de software, mediante el cual determinaremos si la arquitectura cumple con los requerimientos no funcionales, si existe algún conflicto entre dichos requerimientos, o bien el impacto y riesgo que puede traer como consecuencia la ausencia de ciertos estilos, patrones o componentes en la arquitectura de software.

7. Validación de la arquitectura de software con los diferentes *stakeholders* para determinar que la arquitectura de software se encuentre acorde a los requerimientos establecidos y cubra todas las necesidades.

Finalmente, hemos discutido algunas desventajas que presenta el modelo para el desarrollo de arquitecturas de software, algunas limitaciones que hemos identificado en el mismo, algunas ventajas que sabemos tiene respecto a otros modelos, algunas recomendaciones para la utilización del modelo de proceso, y el alcance que tiene, así como alguna de las mejoras que puede ser susceptible.

El trabajo futuro para el modelo de proceso de desarrollo arquitecturas de software se encuentra enfocado principalmente a expresar dicho modelo como un proceso formal que incluya las especificaciones de guías, *scripts* y *checklists* que permitan al arquitecto de software contar con recurso fácil de manejar y seguir en su trabajo diario

Por otra parte, deberemos determinar de los pasos adicionales y actividades, definición de entradas y salidas, especificación de recursos o herramientas que pueden apoyar las actividades a seguir, y registro de métricas que permitan controlar y mejorar el proceso, para la obtención de arquitecturas de software de sistemas legados.

Adicionalmente, deberemos modificar el proceso para incorporar actividades que permitan manejar la incorporación o reutilización de una arquitectura existente, o bien, el mantenimiento de una arquitectura en operación, cuando no existe documento de arquitecturas de software disponible.

Por último, resultaría conveniente que incorporemos en la descripción textual del proceso, la descripción de los diferentes estilos arquitectónicos, patrones arquitectónicos, patrones de diseño y métodos analíticos existentes en la literatura, con el objetivo de dar al arquitecto de software como posibilidad, el analizar y seleccionar los más adecuados para la propuesta de solución del diseño de la arquitectura de software.

8. Bibliografía

[1] Sharon A. White, Cuauhtémoc Lemus-Olalde, *The Software Architecture Process*, 1997.

[2] P. Clements, L. Bass, R. Kazman, *Software Architecture in Practice*, Edited by SEI Series, First edition, SEI Series in Software Engineering: Addison Wesley, 2003.

[3] Len Bass, Rick Kazman, *Architecture-Based Development*, CMU/SEI-99-TR-07, 1999, pag: 1-36.

[4] Shaw M. and Garlan D., *Software Architectures: Perspectives on an Emerging Discipline*, Prentice Hall, 1996.

[5] Jan Bosh, *Design and Use of Software Architectures, Adopting and Evolving a Product-Line Approach*, first ed. London: Addison Wesley, 2000.

[6] R. Helm, E. Gamma, R. Johnson, J. Vlissides, *Design Patterns, Elements of Reusable Object-Oriented Software*, Edited by Addison Wesley, Professional Computing Series, 1997.

[7] Francisca Losavio, Ledis Chirinos, Alfredo Matteo, Nicole Levy, Amar Ramdane-Cherif, *Designing Quality Architecture; Incorporating Iso Standards Into the Unified Process Information Systems Management*. Boston: Winter 2004.Vol.21, Iss. 1; pg. 27, 18 pgs. Available from http://0-proquest.umi.com.millenium.itesm.mx:80/pqdweb?did=689834071&sid=1&Fmt=4&clientId=23693&RQT=309&VName=PQD.

[8] Society, IEEE Computer, *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems, Std 1471-2000*, IEEE Computer Society Software EngineeringStandards Committee, 1-23. The Institute of Electrical and Electronics Engineers, Inc., 2000.

[9] Rick Kazman, S. Jeromy Carrière, Steven G. Woods, *Toward a Discipline of Scenario-Based Architectural Engineering,* Software Engineering Institute, Carnegie Mellon University: 1-26.

[10] J. Bosh, P. Molin, *Software Architecture Desing: Evaluation and Transformation*.

[11] Kazman, Klein, Clements, *ATAM: Method for Architecture Evaluation*, CMU/SEI-2000-TR004, 2000, pag: 1-69.

[12] Felix Bachmann, Len Bass, Gary Chastek, Patrick Donohoe, Fabio Peruzzi, *The Architecture Based Design Method*, CMU/SEI-2000-TR-01, 2000, pag: 1-50.

[13] Kruchten, P. "Architectural Blueprints - the "4+1" View, Model of Software Architecture*"*, *IEEE Software*, 1995, pag: 1-15.

[14] Soni D., Nord R. L. and Hofmeister C., *Software Architecture in Industrial Applications*, Washington, USA, 1995.

[15] Robert L. Nord, Daniel J. Paulish, Dilip Soni, Christine Hofmeister, "Effective Software Architecture Design: From Global Analysis to UML Descriptions", *IEEE 0-7695-1050-7/01*, 2001, pag: 741-42.

[16] R. Nord, Ch. Hofmeinster, D. Soni, *Applied Software Architecture,* Edited by Addison Wesley, Second edition, Object Technology Series: Addison Wesley, 2000.

[17] *IEEE Std 1471-2000*, *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*, Software Engineering Standards Committee, IEEE Computer Society, September 2000.

[18] Lau Cohen, *Quality Function Deployment: How to Make QFD Work for You*, Addison-Wesley publishing company, 1995.

[19] Walter P. Bond, Turky N. Al-Otaiby, Mohsen N. AlSharif, *Software Architecture Assessment Using Quality Function Deployment*.

[20] Stephen Haag, M.K. Raja, and L.L. Schkade, "Quality Function Deployment, Usage in Software Development", *Communications of the ACM* 39, No. 1, 1996, pag: 42-49.

[21] Gómez Gómez, Omar, *Proceso de evaluación para arquitecturas de software usadas en el sector empresarial*, Centro de Investigación en Matemáticas, A.C., julio 2005.

[22] R. Meunier, F. Buschamann, H. Rohnert, P. Sommerlad and M. Stal, *Pattern-Oriented Software Architecture, a System of Patterns*, Edited by Wiley, 3 vols. Vol. 1, Wiley Series in Software Design Patterns: John Wiley & Sons, 2001.

[23] Felix Bachmann, Len Bass, Mark Klein, *Deriving Architectural Tactics: A Step Toward Methodical Architectural Design*, CMU/SEI CMU/SEI-2003-TR-004, (2003), pag: 1-53.

[38] *IEEE Standard 1061-1992*, *Standard for Software Quality Metrics Methodolgy*, New York: Institute of Electrical and Electronics Engineers, 1992.

9. Glosario

Arquitecto. Es la persona, equipo u organización responsable de la arquitectura de un sistema (traducido de) [8].

Arquitectura de Software. La arquitectura de software de un programa o sistema de cómputo es la estructura o estructura del sistema, la cual consta de componentes de software, propiedades visibles externamente de esos componentes y la relación a través de ellos [2]. Organización fundamental de un sistema que incorpora, en sus componentes, las relaciones entre cada uno de ellos, el ambiente y guías principales, su diseño y evolución (traducido de) [8].

Atributos de calidad. Requerimiento no funcional de un sistema de software que representa el grado en el cual un software posee una combinación deseable de dichos requerimientos no funcionales. Puede ser por ejemplo, por ejemplo: confiabilidad, facilidad de modificación, rendimiento, facilidad de uso, entre otros (traducido de) [38].

Acoplamiento. Es la medida que indica qué tan fuertemente un elemento está conectado a, tiene conocimiento de, o confía en otro elemento (traducido de) [24].

Cohesión. Es la medida que indica qué tan fuertemente están relacionadas y enfocadas las responsabilidades de un elemento (traducido de) [24].

Componente. Dentro del campo de la arquitectura de software, un componente es cualquier elemento o estructura que tienen asignada una funcionalidad dentro de una arquitectura.

Descripción de la arquitectura. Colección de productos que respaldan la documentación de una arquitectura (traducido de) [8].

Escenario arquitectónico. Son enunciados cortos que describen la utilización normal que los usuarios finales dan al sistema construido de la arquitectura de software, o bien, representan cambios en la misma. Los escenarios representan acciones en el sistema que los diferentes *stakeholders* desean realizar de acuerdo a sus intereses.

Estilo arquitectónico. Consiste de una colección de tipos de componentes con una descripción del patrón o interacción a través de ellos (traducido de) [2].

Hilo de ejecución. Un hilo virtual es un camino simple de ejecución dentro de un programa, modelo dinámico o alguna otra representación de control de flujo (traducido de) [37].

Interfaz. Se refiere a la especificación de un punto de acceso hacia un componente de software.

Orientador arquitectónico (driver). Los orientadores arquitectónicos son el conjunto de necesidades, elementos, decisiones y determinaciones que marcan las directrices hacia dónde tenemos que dirigir el diseño de la arquitectura.

Patrón arquitectónico. Un patrón arquitectónico trata con algún aspecto del comportamiento del sistema que no está en el dominio de la funcionalidad del mismo, pero el cual maneja alguno de sus dominios de soporte, por ejemplo, la manera en cómo el sistema maneja la concurrencia, la persistencia, la tolerancia a fallas o la distribución (traducido de) [5].

Patrón de diseño. Un patrón de diseño describe una estructura comúnmente recurrente de comunicación entre componentes que resuelve un problema de diseño general dentro de un contexto en particular (traducido de) [6].

Proceso de arquitectura. Se refiere a las actividades de definición, documentación, mantenimiento, mejora y certificación de l correcta implementación de una arquitectura (traducido de) [8].

QFD. Quality Function Deployment, es un método para planear y desarrollar un producto o servicio.

Sistema. Colección de componentes organizados para lograr una función específica o un conjunto de funciones (traducido de) [8].

Sistema legado. Se refiere a un sistema de información que se encuentra en operación, cuyo mantenimiento ha sido heredado a un grupo de trabajo que no contribuyó en el desarrollo del mismo, y del cual comúnmente no se cuenta con la documentación que respalda su especificación de requerimientos y diseño arquitectónico.

Stakeholder. Individuo, equipo u organización con interés relativos al sistema. Puede ser un cliente, un usuario, el arquitecto, desarrolladores, evaluadores, etc. (traducido de) [8].

Tradeoff. Se refiere al conjunto de conflictos que puede causar la combinación de dos o más atributos de calidad que se encuentran considerados para una arquitectura de software.

Vista. Se refiere a una presentación del sistema completo desde la perspectiva de un conjunto de intereses relacionados (traducido de) [8].

View packet. Se refiere al paquete más pequeño de documentación arquitectural que puede ser útil a *stakeholder*. La documentación de una vista está compuesta por uno o más *view packets* (traducido de) [8].

Viewpoint. Se refiere a las convenciones para construir y utilizar una vista. Puede ser un patrón o plantilla , por medio de la cual se desarrolla la vista individual para establecer el propósito y técnicas para su creación y análisis (traducido de) [8].

Walk-through. Es un proceso estructurado por un equipo para llevar a cabo la revisión de un artefacto de software, el cual sigue el formato de una presentación con los arquitectos, desarrolladores y *stakeholders*.

10. Apéndice A. Bibliografía Adicional

Libros

[24] Craig Larman, *Applying UML and Patterns*, Prentice Hall, PTR, 2002.

[25] L. Hohmann, *Beyond Software Architecture, Creating and Sustaining Winning Solutions*, Edited by Addison Wesley, 3rd edition, Addison Wesley Signature Series, 2004.

Artículos

[26] Lawrence Chung, Brian A. Nixon, Eric Yu, *Using Non-Functional Requirements to Systematically Select Among Alternatives in Architectural Design*.

[27] Osterwelll, L, *Architecting Processes Are Key to Software Quality*.

[28] Kari Smolander, Kimmo Hoikka, Jari Isokallio, Mika Kataikko, Teemu Mäkelä, "What Is Included in Software Architecture?, a Case Study in Three Software Organizations"*, IEEE Computer Society*, 2002.

[29] P. Orozco, J. I. Asensio, P. García, Y. A. Dimitriadis, and C. Pairot, "A Decoupled Architecture for Action-Oriented Coordination and Awareness Management in Cscl/w Frameworks", *Springer-Verlag Berlin Heidelberg*, 2004, pag: 246-61.

[30] Jung-Eun Cha, Chul-Hong Kim, and Young-Jong Yang, "Architecture Based Software Reengineering Approach for Transforming From Legacy System to Component Based System Through Applying Design Patterns", *Springer-Verlag Berlin Heidelberg*, 2004.

[31] Felix Bachmann, Len Bass, *Managing Variability in Software Architectures*.

[32] Ashish Srivastavaand and Dr. Sanjay Gupta, *Design Patterns in Real Life Object-Oriented Software*.

[33] Tariq Al-Naeem, Ian Gorton, Muhammed Ali Babar, Fethi Rabhi, and Boualem Benatallah, "A Quality-Driven Systematic Approach for Architecting Distributed Software Applications", *ACM 1-58113-963-2/05/000*, 2005, pag: 244-53.

[34] Weihang Wu, Tim Kelly, "Safety Tactics for Software Architecture Design", *IEEE, Computer Society* 0730-3157/04, 2004.

Recursos Electrónicos

[35] Demers, Rich., "A Software Architecture Process"*,* Cincom Smalltalk, 2004/03/13, Available from http://www.cincomsmalltalk.com/userblogs/rademers/blogView?showComments=true&entry=3254118257.

Documentos

[36] Shillito, M. Larry, "Increasing Product Innovation Effectiveness Through QFD and Value Engineering".

[37] Object Management Group, "OMG Unified Modeling Language Specificacion", Version 1.3, alpha R2, Framinngham, MA: January 1999.