ARQUITECTURA DE SOFTWARE PARA LA RENOVACIÓN DE LA E-BUSINESS SUITE DE CARVAJAL TECNOLOGÍA Y SERVICIOS

GUSTAVO ADOLFO RODRÍGUEZ LIBREROS

UNIVERSIDAD DEL VALLE
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
PROGRAMA ACADÉMICO DE INGENIERÍA DE SISTEMAS
SANTIAGO DE CALI
2013

ARQUITECTURA DE SOFTWARE PARA LA RENOVACIÓN DE LA E-BUSINESS SUITE DE CARVAJAL TECNOLOGÍA Y SERVICIOS

Documento Final de Trabajo de Grado
Modalidad Pasantía

GUSTAVO ADOLFO RODRÍGUEZ LIBREROS
0932979-3743
Gustavo.Adolfo.Rodriguez.L@gmail.com

DIRECTOR
Dairo Antonio Cortes Gaviria
Jefe de Servicios Comerciales – E-Business - Capacidad de Aplicaciones
Carvajal Tecnología y Servicios
Dairo.Cortes@carvajal.com

CODIRECTOR
John Alexander Sanabria Ordoñez PhD. D
Docente del Programa Académico de Ingeniería de Sistemas
Universidad del Valle
John.Sanabria@correounivalle.edu.co

Facultada De Ingeniería
Escuela De Ingeniería De Sistemas Y Computación
Programa Académico De Ingeniería De Sistemas
Cali, Noviembre 29 de 2013

Dedicatoria

Este trabajo se lo quiero dedicar especialmente a mis padres, mis hermanos, mi abuela y mi novia, quienes fueron un apoyo fundamental en este proceso.

Gustavo Adolfo Rodríguez Libreros.

AGRADECIMIENTOS

Quiero agradecer inicialmente a Carvajal Tecnología y Servicios, por permitirme tener la oportunidad de ejecutar mi proyecto de grado y hacer parte de mi proceso formativo.

También quiero expresar mis más sinceros agradecimientos por la dedicación, tiempo y energía a cada una de las personas que hicieron posible llevar a cabo este proyecto; sin embargo, me gustaría destacar algunas personas con un reconocimiento especial. Dairo Antonio Cortez, Jefe Familia Servicios Comerciales; por su gran apoyo y orientación constante; Adrián Collazos, ex Coordinador Familia Servicios Comerciales; por su tiempo para debatir ideas; Alejandro Muñoz, Arquitecto de Software Familia Servicios Comerciales, por el acompañamiento, dedicación y conocimientos compartidos; Javier Llanos, Ingeniero de desarrollo Familia Servicios Comerciales; por enseñarme las los aspectos internos de la e-Business Suite; Alejandra Murillo, Consultora Global de Procesos; por el enorme y valioso tiempo que invirtió enseñándome las necesidades del negocio; Juan Camilo Maya, Técnico especializado; por permitirme conocer la e-Business Suite desde la perspectiva de los clientes; María Gómez y Diego Suarez, Ingenieros de Análisis Funcional Familia Servicios Comerciales; por enseñarme la importancia que tiene la correcta especificación de requerimientos; ICOLTRANS, clientes de e-Business Suite; por abrir sus puertas para conocer la e-Business Suite en un ambiente de producción; Carlos Llano, ex Arquitecto de Software Familia Servicios Comerciales, por brindarme las pautas iniciales para desarrollar el proyecto.

Finamente quiero agradecer a la Universidad por permitirnos desarrollar nuestros trabajos de grado en modalidad de pasantía y especialmente a mi maestro, John Alexander Sanabria PhD, por su apoyo y comprensión, durante el desarrollo de este proyecto.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

RESUMEN

El presente trabajo de grado se realizó en modalidad pasantía en la empresa Carvajal Tecnología y Servicios, en el área de e-Business; con una duración de 8 meses. La finalidad de este proyecto fue la definición de una Arquitectura de Software, para la renovación de un producto de la empresa, llamado e-Business Suite. Éste aplicativo desempeña un rol esencial en una de las líneas de servicios más importantes del área de e-Business en Carvajal T&S; el Intercambio Electrónico de Datos. Por lo tanto, la empresa ha determinado que se requiere renovarlo, puesto que, se ha ido desalineado de la estrategia de negocio, y de las necesidades de los clientes. Este proyecto se realizó bajo los lineamientos de una de las mejores prácticas actuales para el diseño de Arquitecturas de Software; donde se evidencia la importancia de una definición deliberada de la Arquitectura de Software, para construir aplicaciones exitosas, efectivas, de alta calidad y sin mayores riesgos. Este documento, permite al lector, conocer cómo se desarrolla el proceso de Arquitectura de Software, aplicado en un contexto específico.

Palabras clave: Carvajal Tecnología y Servicios, Arquitectura de Software, Negocio Electrónico, Intercambio Electrónico de Datos, *Business-to-Business*

ABSTRACT

This graduation work was performed in the modality of internship at the company Carvajal IT and Services, in the area of e-Business; with a duration of eight months. The purpose of this project was to define a Software Architecture, for the renovation of a company's software product called e-Business Suite. This application plays an essential role in one of the most important lines of services in the area of e-Business at Carvajal IT&S; the Electronic Data Interchange. Therefore, the company has determined that is required to renew it, since, this product has been misaligned of the business strategy and customer's needs. This project was developed under the guidance of one of the current best practices for the Software Architecture design; where the importance of a deliberate definition of software architecture is evidenced to build effective, successful and high-quality applications, without assuming higher risks. This work allows to the interested party to know how the software architecture process is developed applied to a specific context

Keywords: Carvajal IT and Services, Software Architecture, E-Business, Electronic Data Interchange (EDI), Business-to-Business (B2B)

1. INTRODUCCIÓN

Este documento contiene un conjunto de conocimientos generales, principios y herramientas sobre la Arquitectura de Software, que han sido adoptados de profesionales muy destacados a nivel mundial en el área de Ingeniería de Software; como lo son: el Arquitecto Len Bass, uno de los miembros principales del Instituto de Ingeniería de Software, y quien ha escrito dos libros galardonados en Arquitectura de Software; y Olive Vogerl, Arquitecto Empresarial y de Tecnologías de Información certificado por IBM Suiza, y quien lidera a nivel mundial la educación en Arquitecturas Empresariales de IBM.

El propósito de este documento es presentar la descripción del trabajo realizado sobre la definición de la Arquitectura de Software para la renovación del aplicativo de e-Business Suite de Carvajal Tecnología y Servicios. Principalmente, la presentación se enfoca en la descripción del procedimiento de diseño de la Arquitectura de Software, y los conceptos sobre cuales se fundamentan cada una de las actividades.

Carvajal Tecnología y Servicios es una empresa multinacional Colombiana del grupo Carvajal S.A, fundada en la ciudad de Santiago de Cali, que cuenta con presencia en 15 países de Latinoamérica. Esta empresa ofrece un amplio portafolio de productos y servicios divididos en unidades estratégicas de negocios, entre las que se encuentran: la Tercerización de Procesos de Negocio (BPO), la Tercerización de Tecnología (ITO) y la Tercerización de Aplicaciones. Las cuales, son soportadas por un equipo de más de 10.000 colaboradores.

La e-Business Suite, es un aplicativo que se emplea en el contexto de los Negocios Electrónicos; principalmente, en el medio del Intercambio Electrónico de Datos, para la gestión de algunos documentos electrónicos que intervienen dentro del ciclo comercial de la cadena de abastecimiento. Se encuentra diseñado especialmente para atender las necesidades de las empresas de tipo fabricante u operador logístico. Este sistema se comercializa desde hace más de diez años en varios países de Latinoamérica, con la mayor parte de clientes concentrados en Colombia, Perú y Venezuela.

El mundo de los Negocios Electrónicos tiene una tendencia a cambiar constantemente, a razón de las nuevas aplicaciones que se desarrollan sobre las tecnologías de información, para mejorar la comunicación y la colaboración entre los socios comerciales. De ahí que, los aplicativos existentes deben evolucionar al ritmo de las exigencias del mercado si realmente desean sostenerse competitivamente durante un periodo de tiempo considerable.

La e-Business Suite es un producto que no fue concebido para evolucionar y por lo tanto genera significativas dificultades en el momento de implementar un cambio; que se traducen en costos y tiempos significativamente elevados. Por esta razón, la empresa, al ser consciente de dicha situación, ha decidido iniciar un proyecto donde se diseñe una alternativa que contenga las características funcionales de e-Business Suite, pero alineadas a las nuevas tendencias y necesidades del mercado, y lo más importarte que permita modificarse rentablemente.

La definición de una Arquitectura de Software es el medio más adecuado para definir una solución estructurada que satisfaga todos los requerimientos técnicos y funcionales de las partes interesadas. Adicionalmente, una correcta definición garantiza la construcción del sistema sobre unas bases sólidas y un producto final efectivo y exitoso que no se expone a riesgos considerables.

En este documento, de acuerdo a las políticas de confidencialidad establecidas por Carvajal Tecnología y Servicios para la protección de los resultados de este proyecto, de tal forma que no se afecten los intereses de la empresa; no se presentan en detalle los resultados obtenidos, y en compensación a esto se expone la investigación primaria realizada para la ejecución de trabajo realizado.

2. PLANTEAMIENTO Y FORMULACIÓN DEL PROBLEMA

La e-Business Suite es una aplicación de Software para Sistema Operativo Windows que se emplea en el sector empresarial por fabricantes y proveedores logísticos, cuya interacción dentro de la cadena de abastecimiento se realiza apoyada sobre el Intercambio Electrónico de Datos. Básicamente, la e-Business Suite permite la generación y visualización de algunos documentos electrónicos, bajo el estándar internacional EANCOM. A grandes rasgos la e-Business Suite permite: generar Catálogos de Precios (PRICAT), generar Avisos de Despacho (DESADV), generar etiquetas de premarcación para el sector textil, visualizar órdenes de compra, visualizar reporte de ventas e inventarios, visualizar avisos de recibo, y visualizar y generar facturas. [1]

El ámbito del Intercambio Electrónico de Datos, dentro de la cadena de abastecimiento, es un entorno que presenta cambios relativamente constantes, derivados de los cambios en las condiciones y necesidades del mercado; por lo tanto, la capacidad de responder efectivamente al cambio, se convierte en un factor clave en aplicaciones de software que hacen parte de éste medio, a la hora de competir en el mercado.

Del mismo modo, es de suma importancia que las aplicaciones puedan ofrecer constantemente nuevas características a sus usuarios; no sólo las necesidades particulares de un cliente deben motivar al cambio; éste, también debe ser motivado por las mismas empresas que las desarrollan, en búsqueda de optimizar el desempeño de sus clientes y transformar sus negocios; y por consiguiente, generándoles ventajas significativas ante sus competidores en el mercado, de una manera sostenible.

A pesar del amplio número de participantes, dentro de la cadena de abastecimiento, que actualmente utilizan el Intercambio Electrónico de Datos, éste número se queda pequeño comparado con la cantidad de socios comerciales que en ciertas regiones geográficas continúan realizado sus procesos mediantes mecanismos de poca eficiencia. Ésta situación se presenta a causa del elevado número de exigencias, y el nivel de capacidades que debe

tener una empresa para participar en éste medio. Por eso se evidencia que otro de los factores clave para los aplicaciones, o servicios de software es poder ser implementados o adquiridos a menores costos.

Por otra parte, un aspecto muy importante en una aplicación de software, es el nivel de exigencia que el sistema impone al usuario para interactuar con él, y la satisfacción que le brinda. Actualmente, no todas las empresas; y en particular las de menor tamaño, pueden disponer de un recurso especializado para ejecutar las tareas que requieren la interacción con el sistema. Por lo tanto, diseñar sistemas altamente intuitivos, prácticos, pero sin dejar de ser robustos, es un elemento fundamental para que más empresas accedan al medio del Intercambio Electrónico de Datos.

Con respecto a las características altamente deseadas en los aplicativos que se emplean en este medio, también se encuentra la versatilidad que ofrezca el sistema para interactuar con él desde diversos tipos de interfaces y puntos geográficos. Por ejemplo, el acceso mediante dispositivos móviles es cada vez más importante, puesto que, resulta más eficiente interactuar con el sistema en los mismos puntos donde se interactúa con el flujo de la mercancía, el elemento de mayor importancia dentro de la cadena de abastecimiento.

Siendo conscientes de las dinámicas y tendencias del mercado, de las futuras exigencias, la empresa ha detectado que la e-Business Suite, como se mencionaba inicialmente, se encuentra en una posición de alto riesgo para afrontar de manera sostenible el futuro próximo. Lo anterior, sustentado en los argumentos serán descritos a continuación.

e-Business Suite, es un sistema que nace de la iniciativa de integrar en un solo aplicativo todas las herramientas que necesitaban, especialmente, las medianas y pequeñas empresas fabricantes, dentro de la cadena de abastecimiento; pero también para todas aquellas cuyo sistema interno no estuviera en la capacidad de generar los documentos de intercambio electrónico de datos propios de las geografías donde la e-Business Suite opera. Así pues, el propósito era ofrecer una solución completa a las empresas que les

permitiera interactuar eficientemente dentro del mundo de los negocios electrónicos.

Las características que los creadores de e-Business Suite querían integran se encontraban en su momento dispersas en varios sistemas de software, propiedad de la empresa, conocidos como: Módulo de Aviso de Despacho, Módulo Gestión al Fabricante, Sistema de Administración de Catálogos y Módulo de Factura Electrónica. A pesar de los esfuerzos, la iniciativa de integración que dio el nacimiento de e-Business Suite, sólo resultó siendo una interfaz gráfica que unificaba el acceso a los módulos mencionados anteriormente, sin mayores cambios, nuevas características o mejoras significativas.

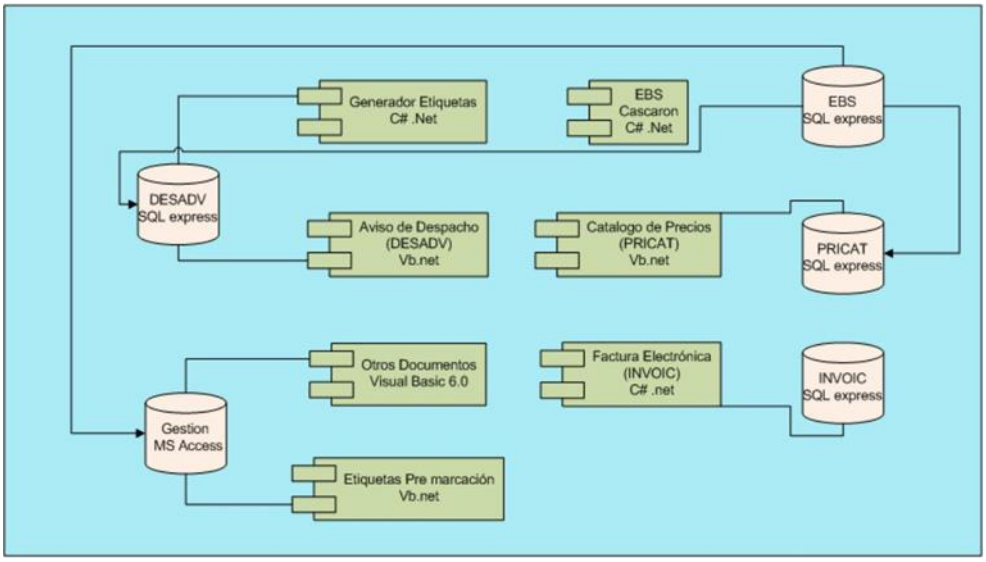

Ilustración 1. Diagrama de componentes del sistema e-Business Suite.

En un primer momento la integración fue una solución aceptable, de hecho, permitía que el cliente adquiriera todas o sólo un subconjunto de las funcionalidades del sistema, mediante un amplio, pero complejo esquema de licenciamiento. No obstante, no transcurrió mucho tiempo para detectarse que la integración, como se había realizado, no satisfaría a cabalidad con las exigencias futura, y por el contrario estaba induciendo a nuevas problemáticas, algunas de las cuales empeoraban o empeorarían con el transcurso del tiempo.

Dentro de los inconvenientes más relevantes de la e-Business Suite, se encuentran las tecnologías en las cuales está desarrollada; por una parte, son bastante obsoletas y por otra son más de una. En su momento, el uso de estas tecnologías implicaba un estilo y paradigma de programación diferente a las prácticas actuales; lo que sumado a, la poca o nula documentación con la que cuenta la e-Business Suite, y el escaso número de ingenieros que tienen la experiencia al respecto, hacen que la implementación de cualquier cambio, cuando resulta siendo viable, sea bastante complejo, que tome mucho tiempo y en últimas, que sea extremadamente costoso.

Por otra parte, al no haberse unificado la persistencia cuando se integraron los módulos, como se puede apreciar en el diagrama de componentes, la información quedo redundante y duplicada, y el sistema de sincronización que se construyó para mantener la información sincronizada implicó significativas pérdidas de rendimiento en el sistema. Adicionalmente, este aspecto implicó numerosas insatisfacciones de los clientes; cuando, por ejemplo, adquirían inicialmente las funcionalidades propias para la gestión del documento Aviso de Despacho, y posteriormente deseaban adquirir las funcionalidades para la gestión de los Catálogos de Precios; en ese escenario, se daban cuenta que la e-Business Suite realmente no integraba dichas funciones, y por lo tanto debían adecuar y cargar nuevamente al sistema toda la información necesaria para poder utilizar la nueva característica.

En relación con el grado de facilidad con el que los usuarios pueden utilizar e-Business Suite, encontramos numerosos casos en el área de soporte técnico, donde los usuarios solicitan apoyo constante para realizar tareas básicas, a pesar de que los colaboradores de las empresas de los clientes reciben capacitaciones cada vez que lo requieren; esto evidencia el alto grado de complejidad que los usuarios experimentan con la herramienta. Así mismo, los clientes cuyo volumen de información es elevado, como por ejemplo, las empresas textiles; que suelen manejar demasiadas referencias para sus productos, tienen que tomarse tiempos extremadamente largos en tareas de mantenimiento de la información como adicionar una nueva colección de

productos o actualizar los precios de una colección existente en el sistema; esto se genera puesto que la herramienta no estaba concebida para el manejo de volúmenes de información tan grandes.

Sumado a lo anterior, la e-Business Suite carece de características versátiles que le permitan integrarse con otros sistemas. Analizando con atención, observamos que el objetivo principal de la gestión que se realiza en e-Business Suite, es básicamente para generar algunos documentos electrónicos; sin embargo, dicha información, en la mayoría de los casos, proviene o necesita regresar a los sistemas internos de cada una de las empresas; por ejemplo a los ERP, una tipología de sistemas de información muy populares en las empresas, que permiten la gestión integral de todos sus recursos. Este aspecto, actualmente, es una de los grandes limitantes en el desempeño de los clientes de e-Business Suite; y el cual, tampoco ha permitido que la herramienta sea implementada en ciertos clientes que la requieren, pero que les resulta inviable transferir los volúmenes de información que manejan, con los limitados mecanismos de integración que ofrece e-Business Suite.

Actualmente, uno de los inconvenientes que presenta mayor ocurrencia, y que resulta cada vez más complejo es la actualización de los clientes hacia nuevos sistemas operativos Windows. Se prevé que de continuar esta situación, es muy probable que en una futura actualización del sistema operativo, no se pueda realizar rentablemente la migración de la e-Business Suite para que sea compatible; lo cual, se constituye como el aspecto que más genera preocupación para la empresa.

Finalmente, es importante resaltar que en Latinoamérica son las cadenas de distribución al detal, las que impulsan, motivan y en ocasiones imponen a las empresas fabricantes que ingresan al Negocio Electrónico, mediante el Intercambio Electrónico de Datos; puesto que para ellos es de vital importancia, porque de otra forma, la gestión de la cadena de abastecimiento sería un caos, teniendo en que los volúmenes de información que pueden llegar a manejar son mil veces mayores en comparación con mayoría de empresas de tipo fabricante.

Por lo anterior, y de acuerdo a la premura que presenta esta situación, la empresa ha decidido ejecutar, con este proyecto, un plan estratégico donde se priorizan las necesidades mencionadas, y por lo tanto, se seleccionan la generación de Catálogos de Precios y Avisos de Despacho, como las funcionalidades que deben ser inicialmente renovadas en una nueva solución. De esta forma, se limita el alcance de este proyecto para agilizar la comercialización (*time to market)* de la nueva solución, y así eliminar rápidamente las limitantes que están surgiendo entre las relaciones comerciales de cadenas y fabricantes, que afectan el desarrollo del mercado.

2.1. Justificación del Problema

Los argumentos que sustentan este proyecto se pueden presentar agrupados desde las siguientes perspectivas:

2.1.1. Justificación Económica

Es de suma importancia para la sostenibilidad de Carvajal Tecnología y Servicios dentro del negocio del Intercambio Electrónico de Datos, entre socios comerciales de la cadena de abastecimiento, contar con una herramienta que contenga las funcionalidades de la e-Business Suite actual, pero que a su vez permita ampliar el mercado y ajustarse de una manera rentable a los constantes cambios que surgen en este medio, por un periodo de tiempo considerable. De no hacerlo, existe una alta probabilidad que la insatisfacción de los clientes, por las limitaciones, incompatibilidades e ineficiencia que día a día impondrá la herramienta frente a las nuevas necesidades, conlleve a pérdidas significativas de clientes; lo que en última instancia, implicará también pérdidas económicas considerables. Además, considerando la acelerada dinámica del mundo de las tecnologías de información; existe también la posibilidad que surjan soluciones sustitutas a ésta por parte de los competidores del sector, que de no estar preparados, absorberán por completo el mercado y obligaran a la e-Business Suite a abandonar la competencia.

2.1.2. Justificación Social

La visión que se tiene para el nuevo sistema, como hemos podido observar, está orientada a ofrecer significativas mejoras en las características actuales de éste; como también, a ofrecer nuevas características, que beneficiarán a la comunidad de cliente actuales, permitiendo que se transformen sus negocios mediante la mejora constante de su desempeño. Pero también, se generará una alternativa muy atractiva para todos clientes potenciales; como lo son, en su mayoría, las pequeñas y medianas empresas, de acceder al mundo de los negocios electrónicos sin tener que realizar grandes inversiones.

Si analizamos un poco más las implicaciones de tener una cadena de abastecimiento cada vez más eficiente, podemos observar que todos estos impactos se trasladarán, de igual forma, a los consumidores finales. Por lo tanto, si se logran mejorar los procesos, desarrollar nuevos productos o servicios, o disminuir los costos, el consumidor final tendrá más y mejores opciones a la hora de realizar sus compras.

2.1.3. Justificación Académica

Este proyecto puede ser de gran utilidad para todos los estudiantes de Ingeniería de Sistemas, que deseen conocer una de las prácticas mayormente aceptadas, en la actualidad, para la elaboración de una arquitectura de software, junto con sus principales elementos relacionados, aplicada en un contexto en particular.

2.2. Objetivos

2.2.1. Objetivo General

Diseñar una arquitectura de software para un sistema que contenga las funcionalidades de los módulos de Catálogo de Precios y Aviso de despacho de la e-Business Suite de Carvajal Tecnología y Servicios, ajustadas a las exigencias actuales del mercado.

2.2.2. Objetivos Específicos

1. Crear la visión general del sistema para identificar el contexto del negocio y el contexto del sistema mediante la especificación de los requerimientos funcionales y los atributos de calidad.

2. Determinar los requerimientos de mayor influencia arquitectónica para establecer sus respectivas prioridades, mediante una clasificación basada en el beneficio que representan para el cliente y los riesgos que representa implementarlo.

3. Determinar los medios arquitectónicos sobre los cuales se va a fundamentar el diseño de la arquitectura, mediante una disertación de los medios utilizados comúnmente en este dominio del problema. [2, pp. 115-239]

4. Diseñar la arquitectura de software del sistema, realizando su especificación mediante el modelo de representación arquitectónico 4+1 de Philippe Kruchten. [3, p. 415]

5. Evaluar la arquitectura de software mediante el uso de la versión ligera del método de análisis de la compensación de la arquitectura ATAM, para determinar si el diseño es apto para el propósito que se requiere. [4]

3. MARCO DE REFERENCIA

3.1. Arquitectura de Software

Aunque existen numerosas definiciones para La Arquitectura de Software, una aproximación que es ampliamente aceptada en el área de la Ingeniería de Software, es la que establece la definición desde dos perspectivas:

Desde el punto de vista de la estructura de un sistema; la Arquitectura de Software de un sistema es la estructura o estructuras del sistema, lo que comprende a sus componentes, las propiedades externas de dichos componentes y la relación que existe entre ellos y con su ambiente.

Desde el punto de vista de la disciplina; la arquitectura de software cubre las actividades de arquitectura y las decisiones relacionadas sobre el diseño e implementación de una arquitectura de software. [2, pp. 43-53]

La Arquitectura de Software, entonces, es un proceso en el cuál se define una solución estructurada que satisface todos los requerimientos técnicos y funcionales previamente establecidos, y dónde adicionalmente se busca optimizar los atributos de calidad, como lo son: el desempeño, la modificabilidad y la seguridad, entre otros. El proceso de definición de una arquitectura implica tomar una serie de decisiones basados en un amplio rango de factores y donde cada decisión pude tener un alto impacto en el éxito de la aplicación.

Dentro de las decisiones generales de mayor importancia para el proceso de definición de la arquitectura se encuentra: la selección de los elementos estructurales que componen el sistema y sus interfaces; el comportamiento del sistema, especificado como colaboración entre sus elementos; la composición de los elementos del sistema; y el estilo arquitectural que guía la organización del sistema.

La definición deliberada de una Arquitectura de Software es de suma importancia puesto que permite minimizar todos los riesgos a los cuales puede terminar expuesto un sistema si se omite algún detalle clave, situación que puede ser muy probable cuando se construyen estructuras tan complejas como el software sin unas bases sólidas. Adicionalmente, este proceso permite articular adecuadamente a todos los actores involucrados, la infraestructura y los objetivos del negocio. En otras palabras, la arquitectura determina finalmente el nivel de calidad del sistema y sus funcionalidades.

Generalmente, las arquitecturas, son creadas basadas en los requerimientos del sistema y los medios disponibles para implementarlos, por lo tanto su definición es el escenario ideal para debatir y compensar los requerimientos contradictorios o competitivos. Como también para construir una unión entre los requerimientos del negocio y los requerimientos técnicos. [5]

3.2. Niveles de Arquitectura

Las Arquitecturas de Software deben exponer la estructura del sistema sin presentar los detalles de implementación; sin embargo, los diseños arquitectónicos se pueden ubicar en diferentes categorías, dependiendo del nivel de abstracción con el que se observe el sistema. Mediante el siguiente gráfico se ilustra la relación jerárquica de los niveles básicos de arquitectura.

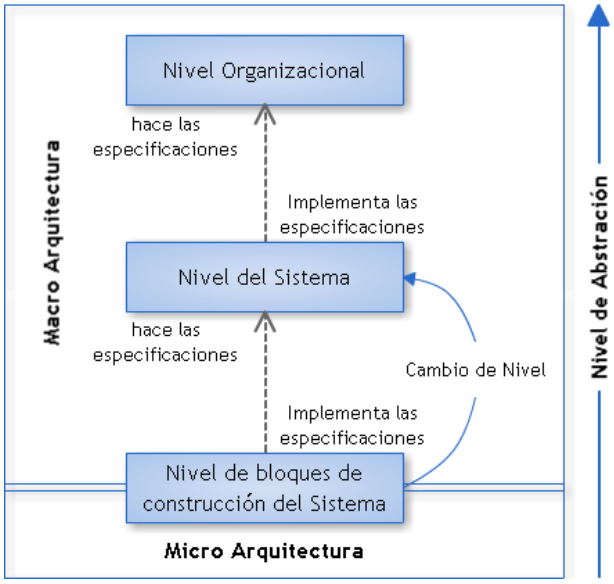

Ilustración 2. Modelo de los niveles arquitectónicos básicos [2, p. 68]

El modelo de niveles de arquitectura permite diferenciar y categorizar cada uno de los requerimientos y decisiones en su nivel correspondiente; lo cual, tiene un impacto positivo para la calidad de la arquitectura, porque permite identificar los orígenes de cada una de las especificaciones que la influencian, y de esta manera orientar los problemas y preguntas que surgen cuando se está creado la arquitectura, evitando mezclar diferentes aspectos, que en última instancia, conlleva a diseños más uniformes y consistentes.

Un Arquitecto de Software que diseña la arquitectura por niveles podrá lidiar con los problemas de una manera más fácil, puesto que contará con los medios más adecuados para tratar cada situación en su nivel correspondiente.

Un ejemplo que permite ilustrar los asuntos que son particulares a cada nivel de arquitectura es el siguiente: los procesos de negocio, son propios del nivel organizacional; los casos de uso del sistema, son propios al nivel del sistema; y la forma como se implementa un caso de uso corresponde al nivel de los componentes que construyen el sistema. [2, pp. 66-76]

3.3. Principios Generales de la Arquitectura de Software

Existen algunos principios generales que pueden ser empleados para guiar el diseño de una Arquitectura de Software, puesto que, han sido desarrollados a lo largo del tiempo basados en soluciones previas que han sido efectivas y exitosas en sus propósitos.

Dentro de los principios generales más importantes, existe uno que establece que todos los diseños deben ser concebidos para evolucionar con el transcurso del tiempo, y no ser concebidos para perdurar. Por lo tanto hay ciertas cuestiones que se deben resolver durante el diseño, si se desea aplicar este principio. Por ejemplo, identificar cuáles son las partes de la arquitectura que representa el mayor riesgo si se laboran erróneamente o determinar cuáles son las partes de la arquitectura que tienen mayor probabilidad de cambiar.

Por otra parte, otro principio general establece que sólo se debe modelar cuando se requiera analizar y reducir riesgos, un modelo muy formalizado puede perder la capacidad de adaptarse o modificarse fácilmente. De igual forma, no se puede esperar que un diseño quede bien elaborado en un primer momento, siempre se debe diseñar tanto como se pueda probar con respecto a las suposiciones y requerimientos.

Igualmente, existe un principio general que determina que todo diseño arquitectónico debe tener muy claramente definido las suposiciones sobre las cuales se ha realizado, los requerimientos que satisface el diseño, los riegos claves en la alternativa del diseño, las medidas empleadas para mitigar dichos riesgos e indicadores que evidencien las mejoras del diseño respecto al último diseño candidato.

Finalmente, existe un conjunto de principios particulares que promueven la minimización de la complejidad del software mediante la separación del diseño en áreas de interés. Entre éstos encontramos: El principio de separación de aspectos.

3.4. Atributos de Calidad

Los requerimientos de un sistema, independientemente de la forma como se especifiquen, se pueden agrupar en tres categorías: funcionales; los cuales determinan como se debe comportar o reaccionar el sistema ante estímulos específicos, atributos de calidad; que representan las cualificaciones de las funcionalidades o de todo el sistema, y restricciones; que corresponden a las decisiones que ya han sido tomadas.

Sin embargo, durante el desarrollo de un sistema, las funcionalidades no deberían ser el centro de atención exclusivo, puesto que las cualidades de un sistema se encuentras reflejadas más allá de sus funcionalidades. Las funcionalidades tienen una relación bastante extraña con la arquitectura, toda vez que éstas no son las que la determinan. Sin lugar a dudas, dado un conjunto de funcionalidades requeridas, existen un sin número de arquitecturas que podrían ser creadas para satisfacer dichas funcionalidades. Si las funcionalidades fueran lo único que importase, entonces no se tendría que dividir y estructurar un sistema en diversos componentes. La razón por la cual se divide un sistema en conjuntos de elementos estructurales cooperativos, es porque que quieren lograr otro tipo de propósitos; éstos se conocen comúnmente como los atributos de calidad.

Un atributo de calidad es una propiedad medible o comprobable de un sistema que se utiliza para indicar qué tan bien el sistema satisface las necesidades de las partes interesadas. Se puede pensar en un atributo de calidad como la medida de bondad de un producto a lo largo de alguna dimensión de interés. Éstos se logran satisfacer por las diversas estructuras diseñadas en la arquitectura, los comportamientos y las interacciones de los elementos que componen esas estructuras

Muchos sistemas de software son rediseñados frecuentemente; no porque sean funcionalmente deficientes, puesto que sus reemplazos en general son funcionalmente idénticos, sino porque son difíciles de mantener, portar, escalar, son demasiado lentos, o presentan problemas de seguridad. De ahí tenemos que la arquitectura de software es el primer lugar en la creación de un sistema en donde los atributos de calidad juegan un papel fundamental. Por lo tanto, el grado en el que un sistema posea la combinación deseada de atributos de calidad; tales como facilidad de uso, rendimiento, fiabilidad y seguridad, indica el éxito del diseño y la calidad en general del sistema.

Existen dos categorías de atributos de calidad que comúnmente generan mayor importancia; la primera corresponde a los atributos de calidad que describen propiedades del sistema en tiempo de ejecución, como por ejemplo la disponibilidad, el rendimiento o la usabilidad; la segunda corresponde a los que describen propiedades que determinan el desarrollo del sistema, como lo son la modificabilidad y la capacidad de prueba.

Los atributos de calidad generalmente no se reflejan en componentes específicos; por el contrario, éstos se logran mediante la interacción o la estructuración de todos los componentes del sistema. Es por esto, que importantes arquitectos como Paul Clemest y Leen Bass; autores del galardonado libro Software Architecture in Practice, han manifestado su inconformidad por el uso negligente de las expresiones “funcional” y “no funcional” para referirse a los requerimientos de un sistema. Ellos establecen que un sistema se describe mejor como un conjunto de responsabilidades que deben lograrse con un determinado nivel de calidad; puesto que, no todos los requerimientos funcionales de un sistema corresponden al propósito del éste, y análogamente, para lograr un atributo de calidad es necesario inducir funcionalidades adicionales, o mejor dicho, responsabilidades adicionales, en el sistema. [3]

Finalmente, es importante destacar que, dentro de los sistemas complejos, los atributos de calidad no se pueden lograr aisladamente, el logro de un atributo de calidad, generalmente, tendrá un efecto positivo o negativo en el logro de otros. Un ejemplo muy claro, es el impacto negativo que tiene el logro de casi todos los atributos de calidad, en el rendimiento del sistema. Determinar el diseño que satisfará todos los atributos de calidad requeridos depende de realizar las compensaciones adecuadas entre éstos. [3]

Atributos de Calidad Frecuentes:

Existe una clasificación para los atributos de calidad que se divide en cuatro áreas específicas vinculadas con el diseño, la ejecución, el sistema y las cualidades de los usuarios.

En la siguiente tabla se presenta una lista con los atributos de calidad más comunes en el área de la arquitectura de software.

Tabla 1. Atributos de Calidad. [5]

Categoría	Atributo de Calidad	Descripción
Cualidades de Diseño	*Integridad Conceptual*	Define la consistencia y coherencia del diseño en general. Esto incluye la forma en que los componentes o módulos están diseñados, y otros factores como el estilo de codificación y la denominación de las variables.
	Modificabilidad	Es la capacidad del sistema para experimentar cambios con un cierto grado de facilidad. Estos cambios podrían afectar componentes, servicios, funciones e interfaces al añadir o cambiar la funcionalidad, la corrección de errores, y la satisfacción de las nuevas necesidades de negocio.
	Reusabilidad	Define la capacidad de los componentes y subsistemas para ser adecuados para su uso en otras aplicaciones y en otros escenarios. Reutilización reduce al mínimo la duplicación de componentes y también el tiempo de ejecución.

Categoría	Atributo de Calidad	Descripción
Cualidades en Tiempo de Ejecución	*Disponibilidad*	Define la proporción de tiempo que el sistema es funcional y trabaja. Se puede medir como un porcentaje del tiempo de inactividad del sistema total durante un período predefinido. La Disponibilidad se verá afectado por los errores del sistema, problemas de infraestructura, ataques maliciosos, y la carga del sistema.
	Interoperabilidad	Es la capacidad de un sistema o sistemas diferentes para operar con éxito mediante la comunicación y el intercambio de información con otros sistemas externos escritos y dirigidos por agentes externos. Un sistema interoperable hace que sea más fácil para intercambiar y reutilizar la información, tanto interna como externamente.
	Manejabilidad	Capacidad de administración que define lo fácil que es administrar la aplicación, por lo general a través de los instrumentos suficientes y útiles que se exponen para su uso en sistemas de control, de depuración y el ajuste del rendimiento.
Cualidades en Tiempo de Ejecución	*Rendimiento*	Es una indicación de la capacidad de respuesta de un sistema para ejecutar cualquier acción dentro de un intervalo de tiempo dado. Se puede medir en términos de latencia o rendimiento. La latencia es el tiempo necesario para responder a cualquier evento. El rendimiento es el número de eventos que tienen lugar dentro de un período de tiempo dado.
	Confiabilidad	Es la capacidad de un sistema para seguir funcionando con el tiempo. Se mide como la probabilidad de que un sistema no dejará de realizar las funciones previstas en un intervalo de tiempo especificado.
	Escalabilidad	Es la capacidad de un sistema para manejar cualquiera aumento en la carga sin sufrir impacto en el rendimiento del sistema, o la capacidad de ser fácilmente ampliado.
	Seguridad	Es la capacidad de un sistema para evitar acciones maliciosas o accidentales fuera del su uso diseñado, y para evitar la divulgación o pérdida de información. Un sistema de seguridad tiene como objetivo proteger los activos y evitar la modificación no autorizada de información.

Categoría	Atributo de Calidad	Descripción
Cualidades del Sistema	*Soportabilidad*	Es la capacidad del sistema para proporcionar información útil para la identificación y resolución de problemas cuando no puede trabajar correctamente.
	Capacidad de prueba	Es una medida de la facilidad que se tiene para crear criterios de prueba para el sistema y sus componentes, y para ejecutar estas pruebas a fin de determinar si se cumplen los criterios. Buena capacidad de prueba hace que sea más probable que los fallos en un sistema se puedan aislar de una forma oportuna y eficaz.
Cualidades del Usuario	*Usabilidad*	Define lo bien que la aplicación cumple con los requerimientos del usuario y del consumidor por ser intuitiva, fácil de localizar y globalizar, ofreciendo un buen acceso para minusválidos, y que resulta en una experiencia de usuario satisfactoria

Especificación de los Atributos de Calidad:

La elaboración de los atributos de calidad no es una tarea sencilla; de hecho, muchas de las definiciones que se realizan generalmente, no se pueden probar de forma exacta y contienen ambigüedades. Además, en ocasiones, resulta complejo establecer a qué tipo de atributo de calidad corresponde una necesidad en particular. Sin embargo, se han desarrollado metodologías que permiten orientar esta labor, como lo es, la definición de atributos de calidad por escenarios; medio sistemático que permite caracterizarlos adecuadamente.

Dentro de las categorías de atributos de calidad más frecuentes, se han desarrollado vocabularios propios para la definición de sus escenarios. Por ejemplo, los atributos relacionados con el rendimiento, se describen mediante eventos que llegan al sistema; los atributos de seguridad,

mediante ataques que sufre el sistema; los atributos de disponibilidad, mediante fallas que presenta un sistema, y los atributos de usabilidad, mediante interacciones del usuario.

Un escenario que describe un atributo de calidad debe especificar, como mínimo los siguientes cuatro aspectos:

- **Estimulo:** Es el evento o condición que requiere una respuesta cuando llega el sistema. Así mismo éste término aplica para un acción que motive una cualidad de desarrollo, como por ejemplo una solicitud de modificación al sistema.
- **Fuente del Estimulo:** Corresponde al origen del evento que llega al sistema. Ésta puede ser determinante en la forma como el sistema trata el estímulo. Dentro de las entidades que podrían generar los estímulos se encuentran, por ejemplo, los usuarios humanos o los sistemas de cómputo, entre otros.
- **Respuesta:** Determina como el sistema debe responder al estímulo, es decir, que tipo de actividades debe llevar a cabo. Aquí es donde se establecen las responsabilidades; en tiempo de ejecución o tiempo de diseño, que el sistema debe tener para dar una respuesta adecuada
- **Medida de la Respuesta:** Se requiere tener algún tipo de indicador que permita determinar si la respuesta es o no satisfactoria cuando ésta ocurra, es decir, si se logra o no el atributo de calidad.

Así mismo, se sugiere especificar dos aspectos adicionales al escenario que brindan una mejor claridad al escenario, estos son:

- **Ambiente:** Describe las circunstancias sobre las cuales el escenario de lleva a cabo. Adicionalmente permite cualificar el estímulo.
- **Artefacto:** Representa la porción del sistema, si es posible diferenciarla, donde aplica el requerimiento.

Es importante resaltar, que existe una diferenciación entre los escenarios de atributos de calidad generales, los cuales pueden pertenecer potencialmente a cualquier sistema; de los escenarios de atributos de calidad concretos, que son específicos a un sistema en particular. [3]

La siguiente ilustración, presenta la relación de todos los aspectos, de la especificación formal, de seis partes, de un atributo de calidad:

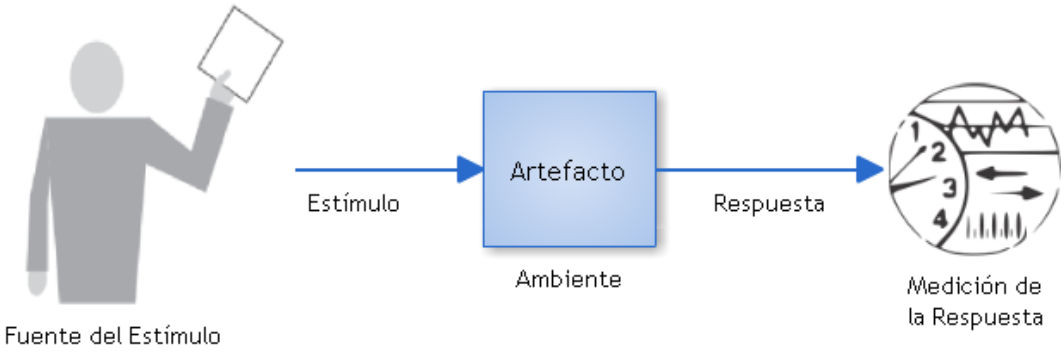

Ilustración 3. Partes de un Escenario de un Atributo de Calidad

3.5. Medios Arquitectónicos

Los medios arquitectónicos son un conjunto de conceptos y técnicas que conforman la caja de herramientas que apoyan el diseño de un arquitecto de software.

Los medios arquitectónicos se abstraen de dominios concretos, tecnologías, y enfoques tecnológicos. Estos medios se pueden jerarquizar basados en el nivel que influencian la arquitectura en construcción, donde la influencia aumenta desde los principios arquitectónicos hasta las arquitecturas de referencia. A mayor nivel de influencia tenga un medio arquitectónico, entonces representa una especificación más concreta.

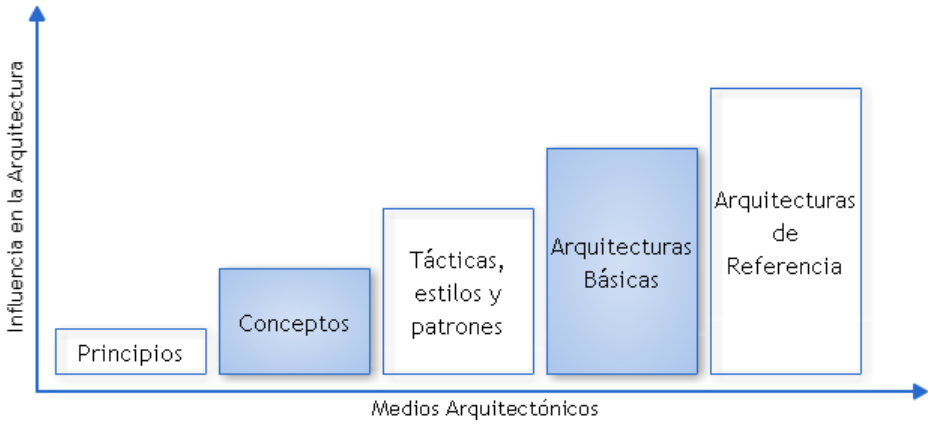

Ilustración 4. Influencia de los medios arquitectónicos [2, p. 117]

Los principios son directrices comprobadas que se deben aplicar cuando se desarrolle o se modifique una arquitectura. Sin embargo, no especifican nada acerca de cómo se deben aplicar estos principios en casos concretos. Los conceptos básicos hacen los principios más concretos y proporcionan una base para su implementación. Las tácticas hacen los principios generales aun una especificación más concreta, ofreciendo directrices basadas en atributos de calidad. Los patrones y estilos, que son conceptos muy similares, tienen aún más directrices concretas y detalladas. Finalmente los patrones de arquitectura y estilos arquitectónicos ofrecen enfoques detallados para apoyar las decisiones concretas de diseño. Por el contrario, las arquitecturas de referencia indican claramente cómo debería ser una arquitectura.

3.6. Tácticas Arquitectónicas

Se requiere una buena planeación para lograr que las respuestas de un sistema, de acuerdo con los atributos de calidad requeridos, permitan satisfacer los objetivos del negocio. Por lo tanto, son las Tácticas Arquitectónicas las técnicas que un arquitecto puede usar para lograr las respuestas del sistema adecuadas.

Básicamente, una táctica, es una decisión de diseño que determina o influencia el logro de un atributo de calidad. Cada táctica se basa en la respuesta del sistema respecto a un único atributo de calidad; por consiguiente, afecta directamente la respuesta a cada estímulo en particular.

Dentro de las tácticas, no se establece ninguna consideración respecto a los efectos que su elección puede tener sobre otros atributos de calidad; ésta labor tiene que ser controlada directamente por el arquitecto diseñador. Aquí se puede observar una gran diferencia respecto a los patrones arquitectónicos, dónde las compensaciones entre los efectos de los atributos de calidad, generalmente, son parte inherente.

Respecto a los patrones de diseño, que típicamente son agrupaciones de decisiones de diseño; en su mayoría, resultan bastante complejos para ser adoptados literalmente, puesto que se beben adaptar y modificar para que se ajusten a las necesidades particulares. Sin embargo, un arquitecto, que comprende claramente el rol de las tácticas, puede fácilmente, evaluar las opciones de utilizar un patrón arquitectónico existente para lograr un atributo de calidad, o diseñar un fragmento que le permita lograrlo a partir de principios generales. En síntesis, las tácticas que dan al arquitecto una visión de las propiedades que debe tener el diseño resultante.

3.7. Categorización De Las Decisiones De Diseño

La arquitectura, como se ha mencionado, puede ser vista como el resultado de aplicar un conjunto de decisiones de diseño; por eso, se recomienda realizar una categorización de las decisiones de diseño, que provea una división lógica de las necesidades y permitan al arquitecto enfocarse en las dimensiones de diseño que pueden resultar más problemáticas. Entre esas categorías se encuentran las siguientes:

- **Asignación De Responsabilidades:** Corresponde a identificar las responsabilidades más importantes del sistema, para determinar cómo se van a asignar a los elementos del sistema.
- **Modelo De Coordinación:** Define el mecanismo que establece como van a interactuar los elementos del sistema. Esta dimensión corresponde a tomar decisiones como por ejemplo, el mecanismo de comunicación
- **Modelo De Datos:** Define la forma como el sistema va a representar e interpretar la información. Aquí se deben decidir aspectos como: las abstracciones de datos más importantes, y la organización de los datos, es decir, si se utilizará una base de datos relacional o una colección de objetos, por ejemplo.
- **Administración De Recursos:** Esta dimensión define la forma como se van a administrar los recursos compartidos, como las unidades de procesamiento y memoria, por ejemplo. Se debe definir cuáles son los recursos que deben ser administrados, y sus limitantes, como también cual es el elemento que administrará cada recurso; entre otros
- **Mapeo Entre Los Elementos Arquitectónicos:** Cada elemento del diseño de la arquitectura se debe mapear a un elemento con menor nivel abstracción, es decir, componentes de Software corresponderán a unidades de ejecución, y éstos a su vez, corresponderán a elementos del entorno.
- **Establecer Decisiones De Tiempo:** Son las decisiones que introducen rangos permitidos de variación, respecto al ciclo de vida de las diferentes entidades de software. Por ejemplo, un sistema se puede diseñar para aceptar conexiones con nuevos dispositivos, con interfaces desconocidas, cuando se esté ejecutando, si se descargan los controladores correspondientes. Las decisiones en esta dimensión deben considerarse respecto al costo de implementarlas, y el costo

de realizar una modificación correspondiente a la decisión tomada, después de que ésta se encuentre implementada.

- **Selección De Tecnología:** Todas las decisiones definidas en el diseño de la arquitectura, generalmente, terminan siendo implementadas mediante un conjunto de tecnologías en particular. Si la selección de la tecnología no se ha realizado previamente; es decir, no constituye una restricción al diseño, el arquitecto deberá seleccionar entre las tecnologías que pueda disponer, las que le permiten realizar me mejorar maneras la decisiones tomadas en las dimensiones previas. Aspecto como el las herramientas que soportan una tecnología, la familiaridad que se tenga con la ella, el grado de soporte externo que ofrezca, y el grado de compatibilidad con el ambiente de tecnologías que se tienen; son criterios claves para tomar decisiones en este sentido. [3]

3.8. Requerimientos Arquitectónicamente Significativos

El objetivo principal de una arquitectura de software es permitir la construcción de un sistema que realmente satisfaga todos los requerimientos. Sin embargo, no todos los requerimientos tienen el mismo efecto en la definición de la arquitectura; es por esto, que se les denomina Requisitos Arquitectónicamente Significativos; ASR (*Architecturally Significant Requirement*) por sus siglas en inglés, a todos los requerimientos que tendrán un efecto profundo en el diseño de la arquitectura.

No es posible diseñar una buena arquitectura si realmente no se tienen claros este tipo de requerimientos; que generalmente, pero no siempre, se obtienen de los atributos de calidad requeridos por el sistema. En la medida que un atributo de calidad sea más importante y complejo es más probable que su efecto en el diseño sea mayor, y por lo tanto que sea un ASR.

Aunque normalmente se pensaría que el sitio ideal para la identificación de los ASR son los documentos de requerimientos o las historias de usuario, no siempre sucede así. Por eso, este tipo de requerimientos deben ser identificados por los arquitectos de software, mediante un trabajo de descubrimiento de ASR candidatos. Los arquitectos más experimentados, lo primero que suelen hacer es comunicarse con las partes interesadas más importantes para el sistema, muchas veces recolectando información que quizás ya ha sido identificada.

No se recomienda empezar el trabajo de arquitectura sólo cuando los requerimientos están totalmente listos; si no, cuando su definición todavía está en proceso. Esto se sustenta en el hecho, que gran parte de las especificaciones de requerimientos, se enfocan, en la definición de las funcionalidades o características que el sistema debe tener; que, de acuerdo a lo establecido anteriormente, puede tener muy poco impacto en el diseño de la arquitectura del sistema. Por otra parte, muchas de las típicas definiciones de requerimientos de la forma, "*el sistema debe ser modular*", "*el sistema debe exhibir alta usabilidad*" o "*el sistema debe cumplir con todas las expectativas de rendimiento de los usuarios*", no son realmente requerimientos; pero por lo menos, pueden orientar la búsqueda del arquitecto.

Los requerimientos arquitectónicamente significativos pueden ser recolectados de diversas formas. Aunque, como se observó, los documentos de requerimientos no siempre suelen ser tan efectivos, no dejan de ser una fuente primaria para dicha tarea. Otra de las formas más comunes para realizar esta actividad es mediante entrevistas a las partes involucradas; tienen mayor éxito los proyectos en los cuales los arquitectos apoyan la definición de los atributos de calidad requeridos; puesto que, usualmente, las partes interesadas no tienen un idea clara sobre cuales atributos de calidad desean en su sistema; y si se les insiste en establecer unos atributos

de calidad cuantitativos, muy probablemente se obtendrán números arbitrarios, es decir, requerimientos muy difíciles de satisfacer. Los arquitectos, usualmente, tienen amplio conocimiento respecto a los atributos de calidad que exhiben sistemas similares, y pueden sugerir algunos atributos de calidad que razonablemente el sistema debe proveer.

Los objetivos de negocio son la razón de ser para la construcción de un sistema; por consiguiente, los ASR también pueden ser obtenidos a partir de éstos. Los arquitectos de software se interesan en los objetivos de negocio porque generalmente son el origen directo de requerimientos que quizás no fueron recolectados en la especificación de requisitos, pero que su logro determina el éxito del diseño arquitectónico. Así pues, se evidencia que atributos de calidad como la seguridad y desempeño son derivados de propósitos más generales que buscan ofrecer un valor agregado a la solución. Es posible encontrar situaciones donde los objetivos del negocio pueden afectar directamente la arquitectura sin estar directamente relacionados a un atributo de calidad; pero también cabe resaltar que no todos los objetivos del negocio conllevan a una solución a nivel arquitectónico.

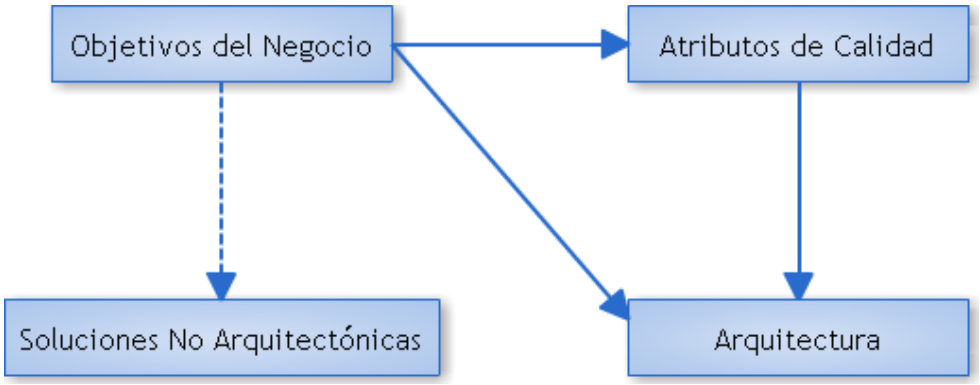

Ilustración 5. Relación entre los Objetivos del Negocio y La Arquitectura

Se recomienda que todos ASR recolectados sean almacenados de tal forma que puedan ser revisados, referenciados, usados para justificar las

decisiones de diseño y revisados a través del tiempo, en caso que el sistema presente mayores cambios. Adicionalmente, existen técnicas específicas, en este sentido, como la construcción del árbol de utilidad; cuya descripción queda por fuera del alcance del presente documento, y la cual consiste en una estructuración gráfica de los ASR, de tal forma que se puedan evaluar en contra los dos principios básicos de un ASR: el nivel de impacto que tienen en la arquitectura; que permite, establecer si su ausencia cambiaría realmente el resultado final de una arquitectura; y el significado o misión que cumple respecto a los objetivos del negocios del sistema. Este tipo de técnicas permiten minimizar el número de omisiones que se podrían incurrir respecto a las necesidades de cada una de las partes involucradas. [3]

3.9. Vistas Arquitectónicas

Visualizar todos los aspectos de un sistema de software de manera completa en una misma representación no es viable. Por tal razón, las vistas arquitectónicas son los medios que permiten hacer entendibles los sistemas complejos, puesto que separan los diferentes aspectos de un sistema.

Existe una gran variedad de definiciones para las vistas arquitectónicas; sin embargo, la definición que proporciona el estándar 1471 [IEEE 2007] define a una vista como una representación de un sistema completo desde la perspectiva de un conjunto determinado de asuntos de interés. Estas vistas generalmente son motivas por las partes interesadas puesto que sus intereses en el sistema determinan un punto de vista que representa cuestionamientos acerca de aspectos específicos

Un punto de vista generalmente determina los elementos básicos que definen una vista arquitectónica como su nombre, las partes interesadas y lo que se debe documentar. También especifican los medios, las plantillas,

los patrones, las mejores prácticas, las directrices y los métodos para crearlas

Teniendo en cuenta que la arquitectura de un sistema requiere de diferentes vistas para su representación, se han diseñado una serie de modelos de vista arquitectónica los cuales representan el conjunto de diferentes puntos de vista. Estos modelos pertenecen parcialmente a un dominio específico, y cada uno establece diferentes vistas de arquitectura útilmente relacionadas entre sí para su uso práctico. Un modelo debe cubrir todas las vistas arquitectónicas que permitan hacer que la visualización de una arquitectura sea tangible. [6]

El siguiente gráfico presenta el modelo de vista arquitectónico común que se ha abstraído siguiendo los modelos de [IEEE 2007] [Rozanaski and Woods 2005], y [Kruchten 2000].

Ilustración 6. Estructura abstraída entre los modelos comunes de vista arquitectónicos [2, p. 82]

4. METODOLOGÍA

A pesar de que existen diversas metodologías para la creación de un sistema de software, son pocas las veces que un proyecto, en este medio, sigue la ruta establecida inicialmente. Las metodologías sólo constituyen una guía, ya que el cambio es un elemento que surge frecuentemente. De acuerdo a lo anterior, el éxito del desarrollo de un sistema se determina de capacidad reaccionar adecuadamente ante el cambio, de tal forma que el proyecto no se desvíe de su rumbo hacia el objetivo general. La capacidad de reacción está estrechamente relacionada con la metodología seleccionada, y la capacidad de los miembros del equipo de adaptar las actividades a condiciones particulares; razón por la cual, las metodologías en cascada, actualmente, no resultan muy exitosas, por tanto que no permiten manejar el cambio adecuadamente.

De acuerdo a lo anterior, es de esperarse que ningún método exitoso para la definición de una arquitectura de software pueda asumir que el diseño estará totalmente completo antes de su implementación; si así fuera, no se podría adaptar fácilmente a los cambios que surjan en los requerimientos. Por otra parte, de ser así, la validación de la arquitectura terminaría realizándose muy tarde en el ciclo productivo de desarrollo, lo que conlleva que la corrección de errores acarree un esfuerzo substancial y mayores costos.

Los métodos iterativos incrementales como USDP (Unified Software Development Process); Proceso Unificado de Desarrollo de Software, refinaron significativamente las dificultades frente al cambio, propio de las metodologías tipo cascada, permitiendo que las todas las actividades se pudieran ejecutaran en ciclos, y que el desarrollo se realizara paso a paso, pieza por pieza, y donde en cada paso se avanzará cada vez más hacia el objetivo general. [7]

Actualmente, las metodologías de desarrollo ágil, como XP y SCRUM han tomado mucha fuera en el sector. Estas metodologías limitan significativamente la documentación, pues consideran que gran parte de ésta es innecesaria; cuestión que ha generado un gran debate respecto a la documentación propia de la arquitectura del sistema. Para mitigar esto, las

metodologías ágiles proponen que el grado de diseño arquitectónico y su documentación correspondiente, debe satisfacer el principio de "suficiente para el propósito"; de esta forma se puede determinar, en cada situación, la cantidad de documentación necesaria.

Antes de continuar, es importante resaltar, que las metodologías de desarrollo sólo son plantillas que perfectamente pueden ser combinadas y adaptadas a contextos y proyectos específicos; de acuerdo con las necesidades particulares que impliquen, como lo son el tamaño, por ejemplo.

Independientemente de la metodología de desarrollo de un sistema, contar con un método propio para la arquitectura del sistema, es un prerrequisito para la definición satisfactoria de la misma. Es de suma importancia contar un procedimiento sistemático que apoye la labor del arquitecto, de tal forma que éste pueda seleccionar los medios adecuados.

Según el trabajo realizado por Bass et al. 2003 [3] y Vogel et al. [2], las actividades generarles que desarrolla un arquitecto, independientemente de la metodología general del proyecto, son las que se presentan en la siguiente ilustración, donde se describe el proceso general del Método de Arquitectura de Software mediante el uso de un diagrama de actividades en notación del estándar UML

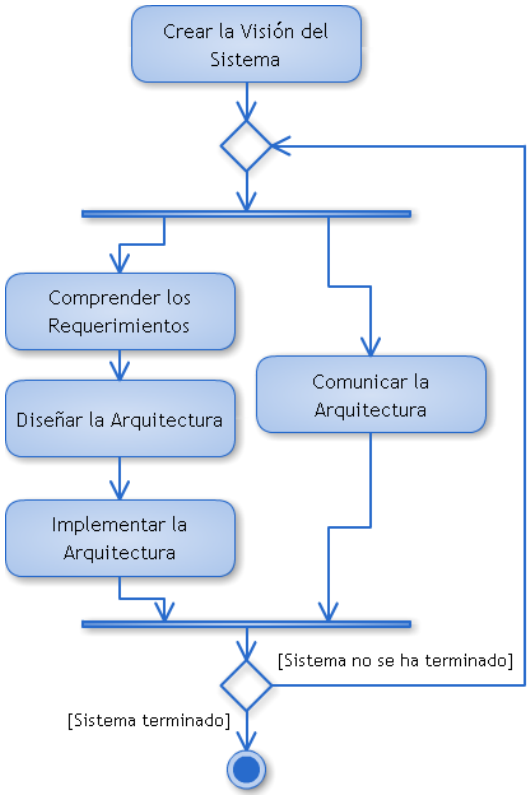

Ilustración 7. Descripción General del Método de Arquitectura de Software

La **creación de la visión del sistema** es un proceso que permite evaluar la utilidad de negocio de una iniciativa; como también, permite definir los requerimientos esenciales para el futuro sistema. Esta es la fase donde arquitecto tiene la responsabilidad de cuestionar los requerimientos para validar si son viables dentro del contexto global de la tecnología de información de la organización. Específicamente, en esta etapa se debe, describir las oportunidades de negocio, identificar las partes interesadas, identificar los requerimientos funcionales y no funcionales, crear la visión del sistema y finalmente evaluarla.

Comprender los requerimientos del sistema implica identificar los requerimientos arquitectónicamente significativos, refinar dichos

requerimientos y finalmente priorizarlos. En particular, es muy importante que el arquitecto examine los requerimientos no funcionales deliberadamente

Es **diseño de la arquitectura** es una de las fases centrales y con mayor número de actividades; es ésta se define el contexto del sistema, se identifican las abstracciones principales, se seleccionan los requerimientos arquitectónicamente significativos, se identifican los medios arquitectónicos a utilizar, se definen los componentes del sistema y sus responsabilidades, se definen las dependencias entre éstos, como también sus interacciones, posteriormente se definen sus interfaces, luego se despliegan los componentes del sistema, y finalmente, se evalúa el diseño.

Implementar la arquitectura comprende las actividades necesarias para implementar el diseño satisfactoriamente. En esta etapa se encuentran actividades como: la definición de la estructura de implementación, la implementación de las bases técnicas, la implementación del esqueleto del sistema, y finalmente la implementación de cada componente del sistema. Esta etapa debe realizarse acompañada de una verificación constante de la conformidad de la implementación con el diseño de la arquitectura.

Comunicar la arquitectura, es quizás la fase, que en muchos proyectos, no se le otorga la importancia que realmente merece. El arquitecto más que diseñar la arquitectura, debe comunicarla a todas los interesados, buscando la mejor forma posible de que todos puedan comprenderla, junto con las decisiones y argumentos que la sustentan. Así, documentar las directrices de la arquitectura, documentar la arquitectura, proveer el entrenamiento requerido, presentar la arquitectura, son las actividades que componen esta fase. [2]

5. VISIÓN GENERAL DEL SISTEMA

5.1. Contexto del Sistema

Para comprender claramente el problema sobre el cual gira este proyecto es importante conocer un poco más afondo el contexto empresarial en el cual se sitúa.

En la unidad de negocio de Tercerización de Aplicaciones, de Carvajal Tecnología y Servicios, se encuentra un área exclusivamente dedicada a los Negocios Electrónicos o e-Business, denominada la Capacidad e-Business; la cual, se enfoca primordialmente en el sector de consumo masivo e industria. Ésta área, anteriormente correspondía a la empresa Assenda S.A.S, del Grupo Carvajal S.A; una empresa que fue ampliamente destacada en el sector.

Gran parte de las soluciones desarrolladas por la capacidad de e-Business, giran en torno a la interacción entre los diferentes actores involucrados en la cadena de abastecimiento, es decir, los fabricantes de productos de consumo masivo y las grandes superficies de comercialización al detal. Las aplicaciones que allí se desarrollan, buscan construir sinergias entre dichos actores, de tal forma, que les permita actuar oportuna y eficientemente de acuerdo a las condiciones y exigencias del mercado.

En la capacidad e-Business de Carvajal T&S, el eje central de interacción de los actores mencionados anteriormente corresponde a la plataforma de servicios CEN (Centro Electrónico de Negocios). El Centro Electrónico de Negocios, cuenta con varias líneas de servicios, entre las cuales se encuentra la línea de servicios transaccionales donde se desarrolló, hace más de diez años, la e-Business Suite, y la cuál se ha ido ajustando a las necesidades del mercado.

Por otra parte, también es importante conocer un poco más sobre la dinámica y el uso los estándares de Intercambio Electrónico de Datos, para comprender, de una mejor manera, lo que estos documentos electrónicos significan, su utilidad y uso.

El estándar EANCOM, básicamente, es un subconjunto del estándar internacional UN/EDIFACT, que corresponde a las siglas, en inglés, de: United Nations/Electronic Data Interchange For Administration, Commerce and Transport; lo cual traduce, en español: Intercambio Electrónico de Datos para la Administración, Comercio y Transporte, de la Organización de las Naciones Unidas. Esté estándar, comprende un conjunto de normas y directrices acordadas internacionalmente, para el intercambio electrónico de datos estructurados; en particular, los relacionados con el comercio de bienes y servicios, entre sistemas de información independientes. [8]

El intercambio electrónico de datos bajo el estándar UN/EDIFACT, gracias a su amplia aceptación mundial, ha facilitado significativamente el comercio entre diversas entidades, independientemente de su localización geográfica. Los procesos más beneficiados han sido los relacionados con el flujo de información necesaria para el movimiento de las mercancías; gracias a la simplificación y normalización los documentos de comercio exterior que ofrece el estándar. [9]

Los esfuerzos de los comerciantes y fabricantes en el mundo por crear un sistema de estándares para interactuar de una mejor manera datan desde el año 1977. Ésta iniciativa surgió inicialmente en Europa y se consolidó con la creación de EAN International (European Article Numbering Association – EAN), que posteriormente, cuando se adhirieron países fuera del continente, cambió su nombre a International Article Numbering Association, pero conservando la abreviatura EAN como el distintivo del sistema de numeración y marcación. [10] Sin embargo, en el año 2005 dicha organización adquirió un nuevo nombre, GS1, el cual se mantiene hasta la fecha.

GS1 es quien ha tenido la responsabilidad de desarrollar y mantener los estándares de acuerdo a cada una de las necesidades particulares de las diferentes regiones geográficas; buscando siempre, proporcionar un sistema integrado, que permita la exacta identificación y comunicación efectiva de información sobre productos, bienes, servicios y ubicaciones. Por lo tanto; ha desarrollado un subconjunto del estándar UN/EDIFACT, el cual se mencionó inicialmente y es el que soporta la e-Business Suite, el EANCOM; que significa

"EAN + Comunication". Éste estándar establece la guía para la implementación del UN/EDIFACT acorde a las necesidades de cada localización geográfica. [11]

El flujo de información bajo el EANCOM, se realiza a través de diferentes tipos de mensajes. Generalmente, más de un tipo de mensaje es requerido para llevar a cabo una completa transacción de negocios; es decir, una transacción de tipo comercial, de transporte o financiera. Teniendo en cuenta la función que desempeña cada mensaje del estándar, se agrupan en las siguientes cuatro categorías: mensajes para la *alineación de Información maestra*, que permiten el intercambio de información sobre productos y servicios entre socios comerciales; m*ensajes para transacciones*, utilizados para requerir productos y realizar pagos por ejemplo; m*ensajes para planeación y reportes*, los cuales proveen a los socios comerciales con información sobre futuros requerimientos; y m*ensajes varios*, que permiten el intercambio de información general para el soporte de aplicaciones entre otros usos por ejemplo. La siguiente gráfica representa cómo interactúan los diversos socios comerciales que interactúan a través de EANCOM:

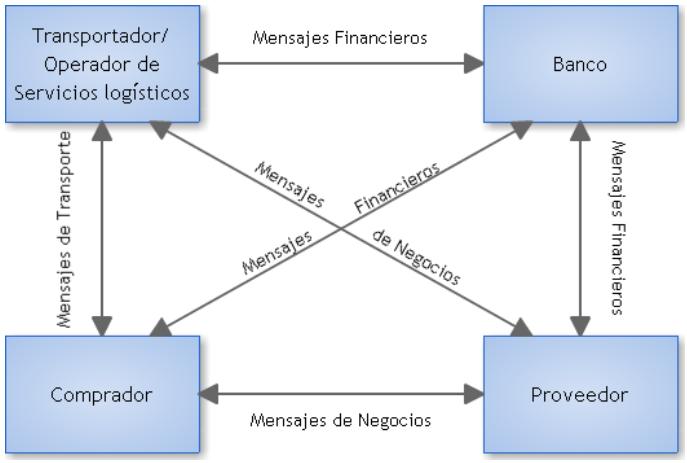

Ilustración 8. Flujo de Información EANCOM

Como se mencionó en la formulación del problema; dadas las necesidades del negocio, las funcionalidades que tienen una mayor prioridad para este proyecto, son las que giran en torno a la gestión y generación de los

documentos o como mejor se conocen dentro del estándar, mensajes de Aviso de Despacho y Catalogo de Precios. De acuerdo a esto, a continuación se expondrá más a fondo la función de dichos mensajes en particular.

El Aviso de Despacho, cuyo identificador dentro del estándar es DESADV; del inglés Despatch Advice Message, es un mensaje que puede ser usado para indicar, como su nombre lo dice, el despacho de bienes que están siendo despachados, o para indicar el despacho de bienes que están siendo devueltos. Este mensaje se debe enviar antes de despachar o regresar los bienes, de tal forma que la parte receptora pueda utilizar la información para prepararse de manera eficiente para la recepción de las mercancías. La intención de este mensaje es avisar detalladamente el contenido de un envío, desde un vendedor a un comprador, o entre sus agencias respectivas. Específicamente, mediante este mensaje se le permite al receptor conocer cuando fueron o serán despachados los bienes, detalles precisos del envío, dar los primeros pasos hacia el despacho de aduanas en el caso de envíos internacionales, y tener un control entre los bienes despachados y su correspondientes facturas. [10]

El mensaje de Catálogo de Precios, identificado como PRICAT; de acuerdo a su nombre en inglés Price/Sales Catalogue Message, permite la transmisión de información sobre los precios y los detalles del catálogo de bienes y servicios ofrecidos por un vendedor a un comprador. Este mensaje también se puede ser enviado desde un comprador a un vendedor para especificar requisitos especiales, tales como el tipo de etiquetado que utiliza el comprador o requisitos de embalaje, o para proporcionar una respuesta de aceptación o rechazo a un precio o catálogo de ventas previamente recibido.

5.2. Visión Inicial del Sistema

La visión general de sistema se desarrolló a partir de la información suministrada por las partes interesadas (Stakeholders), y de la e-Business Suite actual. La comunicación con cada una de las partes interesadas que pertenecían a la organización se realizó mediante reuniones; dándole mayor extensión y frecuencia a las reuniones con las partes interesadas que se encontraban más cercanas al negocio y por lo tanto eran los más adecuados para el establecimiento de los objetivos del negocio.

A continuación se presenta la información correspondiente a cada una de las partes interesadas que participaron en la definición de la visión inicial del sistema. El orden de la tabla corresponde al nivel de influencia que tuvo cada interesado dicha definición; siendo el primer interesado el de mayor influencia.

Tabla 2. Partes Interesadas al Interior de la Organización

ÁREA DENTRO DE LA ORGANIZACIÓN	NOMBRE	CARGO
Vertical Consumo E Industria	Alejandra Murillo Reina	Consultor Global de Procesos
Familia Servicios Comerciales	Dairo Antonio Cortes Gaviria	Jefe Servicios Comerciales
Familia Servicios Comerciales	Adrián Collazos Carvajal	Coordinador de Desarrollo
Familia Servicios Comerciales	Javier Llanos Caicedo	Ingeniero de Desarrollo
Familia Servicios Comerciales	Alejandro Muñoz Andrade	Arquitecto de Software
Familia Servicios Comerciales	Diego Fernando Tejada	Ingeniero de Desarrollo Senior
Soporte Técnico – Primer Nivel	Juan Camilo Maya Agudelo	Técnico Especializado de Aplicaciones E-Business
Implementación	Alberto Borja Pérez	Ingeniero de Implementación Industria
Administración De Contenido	Juan Pablo Payan	Ingeniero Aplicaciones Ebusiness

Para la comunicación con las partes interesadas externas a la organización; como los son los clientes y usuarios finales del sistema, se realizó una encuesta electrónica, vía web, a una muestra representativa de los clientes actuales de

e-Business Suite; especialmente a sus respectivos usuarios dentro de estas empresas. La encuesta nos permitió conocer las experiencias de los usuarios y sus necesidades particulares.

Adicionalmente se realizó una visita a una de las instalaciones de un cliente de e-Business Suite, ICOLTRANS, que aunque sólo utiliza las características del sistema relacionadas con el Aviso de Despacho, es un cliente estratégico por ser un operador logístico; lo que básicamente representa a un prestador de servicios de transporte de bienes. E-Business Suite juega un papel muy crítico en la operación y producción de éste cliente, cuyos servicios de transporte cubren todo el territorio nacional, por lo que cuenta con un amplio número de clientes; por ende, el volumen y flujo de información que gira en torno a e-Business suite es significativamente elevado. Dadas estas condiciones especiales, se determinó que era muy indicado conocer su estado actual, sus necesidades y la forma como opera.

Como resultado de las primeras etapas en el levantamiento de la información, se estableció, por parte de uno de los involucrados más relevantes del proyecto desde la perspectiva del negocio, la premisa, que el nuevo sistema debía ser desarrollado bajo el modelo de negocio de distribución de software SaaS; acrónimo del inglés que traduce, Software como Servicio (Software as a Service). Esta decisión tuvo un significativo impacto en la dirección del proyecto, que implicó un trabajo extenso para determinar la viabilidad de dicha necesidad.

El trabajo de investigación mencionado anteriormente, permitió conocer que las características y condiciones del entorno sobre el cual se implementa la herramienta en los clientes son significativamente heterogéneas. Sin embargo, después de un análisis de esta caracterización se logró categorizar el modo de empleo de la herramienta por parte de los clientes y así se pudieron tratar con mayor facilidad cada una de sus particularidades; concluyendo finalmente que si era factible transformar el modelo de negocio actual de e-Business Suite y desarrollar un sistema que permitirá ofrecer cada una de sus característica como un servicio.

La categorización general que se realizó de los clientes según el modo de operación fue la siguiente:

- Clientes sin sistema interno:

Estos clientes corresponden a las empresas fabricantes de menor tamaño; los cuales no cuentan con un sistema interno del tipo ERP en el cual se requiera involucrar la información de los documentos de Aviso de Despacho o Catálogo de Precios. Para estos clientes, se evidenció que la transformación del modelo de negocio tendría un impacto mínimo

- Clientes con sistema interno:

Este grupo de empresas fabricantes por tu tamaño y estructuración requieren de un sistema interno, generalmente del tipo ERP, en el cual se requiere tener toda la información correspondiente a su operación; por ende involucran directamente a la información que requiere e-Business Suite para generar los documentos de aviso de despacho y catálogo de precios. Las órdenes de compra, un insumo clave para la generación del Aviso de Despacho, son un claro ejemplo de la información que tanto e-Business Suite como los sistemas ERP requieren.

Esta categoría de clientes fue la que mayor preocupación generó durante el diseño de la arquitectura, pero finalmente se logró definir, con un mínimo grado de incertidumbre, un mecanismo de integración que les permitiera a todos los clientes conservar lo esencial del esquema de operación actual, generándoles un mínimo impacto

- Operadores logísticos:

La categoría de operadores logísticos fue también bastante compleja, puesto que, además de ser empresas de gran tamaño, de tener sistemas internos de tipo ERP y de prestar el servicio de despacho a más de una empresa fabricante, para casi todas las empresas a las cuales les prestan el servicio de despacho, cuentan con procesos de operación diferentes. Sin embargo, se observó que los procesos de operación se podían ajustar a un único esquema de operación;

donde al igual que en la categoría anterior, la efectividad del sistema y la minimización del impacto recaían en las características de integración que la nueva e-Business Suite pudiera ofrecer.

6. ARQUITECTURA DEL SISTEMA

6.1. Tácticas Empleadas En El Diseño de la Arquitectura

A continuación se presentan las tácticas seleccionadas, dentro de cada una de las siete categorías de atributos de calidad que son universalmente importantes, que permiten alcanzar los atributos de calidad que este proyecto requería en particular.

6.1.1. Disponibilidad

La disponibilidad, en términos generales, es la propiedad de un sistema de estar listo para ejecutar sus tareas cuando un usuario lo requiere. Un sistema de software que requiere de un alto nivel de disponibilidad debe estar en la capacidad de enmascarar o reparar las fallas de tal forma que el periodo de tiempo acumulativo por fuera de servicio no exceda el valor requerido en un determinado intervalo de tiempo.

La disponibilidad se encuentra estrechamente relacionada a la seguridad y el rendimiento. Un ataque de denegación de servicio podría dejar un sistema inasequible, o una petición compleja podría implicar una respuesta extremadamente lenta, sobre la cual no se podría fácilmente determinar si el sistema ha o no fallado.

Es importante destacar que los fracasos en la operación de un sistema, es decir, las desviaciones del sistema de su especificación, sólo se consideran a partir del momento que han sido visibles externamente por otro sistema u observador humano. Por lo tanto, el tiempo que transcurre en reparar la falla es el tiempo que ocurre hasta que la falla ya no es visible por la entidad que la detectó.

Cuando se piensa en disponibilidad, es clave orientar el análisis mediante cuestionamientos como los siguientes: ¿Cómo detectar las fallas?, ¿Con qué frecuencia podría ocurrir una falla?, ¿Qué pasaría cuando una falla ocurriese?, ¿Cuánto tiempo está permitido para que un sistema esté fuera de operación?

Para diseñar un sistema de alta disponibilidad es indispensable considerar que las fallas son inevitables; no obstante, considerando a su vez, que pueden ser prevenidas, toleradas, removidas o pronosticadas. De ahí que lo que hará que un sistema sea altamente disponible, es la planeación que se realice para atender dichas fallas.

Existen diferentes métodos para identificar cuales con las fallas a las cuales un sistema será más propenso y cuáles son sus consecuencias. El análisis de riesgos es una técnica que buscar catalogar los riegos que pueden ocurrir durante la operación de un sistema acuerdo a su severidad. Una categorización muy común es la que establece los niveles de catastrófico, peligroso, mayor, menor o sin efecto. Por otra parte, el análisis del árbol de fallas, es una técnica que especifica a estado del sistema que puede impactar negativamente su disponibilidad y luego se realiza un análisis para determinar todas las alternativas que pueden conllevar a dicho estado.

Las tácticas de disponibilidad seleccionadas para este proyecto obedecieron a los escenarios elaborados para la descripción de los atributos de calidad en este contexto. Los escenarios de disponibilidad contemplaron la relación de diversos valores para cada uno de sus componentes. Entre éstos se destacan los siguientes:

Tabla 3. Valores empleados en la descripción de los escenarios de disponibilidad

Componente Del Escenario	Valores Empleados
Fuente	Internas y externas: usuarios, hardware, software e infraestructura física
Estímulo	Fallas: Omisiones, caídas, errores de sincronización y respuestas incorrectas
Artefacto	Componentes de procesamiento, canales de comunicación, almacenamiento persistente
Ambiente	Operación normal
Respuesta	Detección de fallas: log de las fallas y notificación a las entidades apropiadas Recuperación de fallas: enmascarar las fallas, estar por fuera de servicio, operación en modo degradado.

Medición De La Respuesta	Tiempo en detectar la falla Tiempo en reparar la falla

Dentro de la especificación de los escenarios de disponibilidad fue de gran dificultad establecer la medición de la respuesta, puesto que, resultaba muy riesgoso determinar valores en particular sin tener la suficiente experiencia y conocimiento sobre la efectividad de las tácticas que se iban a seleccionar.

Las tácticas de disponibilidad seleccionadas que se presentaran a continuación, se encuentran categorizadas de acuerdo los aspectos de mayor importancia en nuestro contexto:

- **Detección de fallas**

Antes de que un sistema pueda tomar una acción respecto a una falla, la presencia de la falla debe ser detectada o anticipada. Las tácticas seleccionadas para lograr esto en nuestro sistema fueron:

  - **Monitor:** Se determinó adicionar un componente que permitiera monitorear constantemente el estado de los demás componentes del sistema.

  - **Marcas De Tiempo:** Con esta táctica se pretende detectar las secuencias incorrectas de eventos, especialmente en el intercambio de mensajes, toda vez que este es uno de los mecanismos de interoperabilidad que contará el sistema.

  - **Detección De Excepciones:** De esta forma, se detectarán todas las condiciones del sistema que alteren el flujo normal de su ejecución.

- **Reparación de fallas**

Estas tácticas nos permiten preparar el sistema para una falla, de tal forma que pueda raparla, o pueda realizar reintroducciones. De acuerdo a las necesidades del sistema se seleccionaron las siguientes tácticas:

- **Redundancia Activa:** Para la implementación de esta táctica se requiere contar con un grupo de nodos de procesamiento, donde más de uno nodo se encuentre activo, de tal forma que éstos puedan recibir y procesar peticiones en paralelo; de esta forma es posible que componente pueda hacerse cargo del servicio inmediatamente si algún otro presenta una falla. Dadas las restricciones particulares al proyecto, sólo será posible emplear la forma más simple de redundancia activa conocida como 1+1

- **Manejo De Excepciones:** Se espera que después de que una excepción ha sido detectada, el sistema recolecte la información necesaria relacionada, como el origen y la causa, de tal forma que el sistema pueda usar esta información para intentar enmascarar la falla, de ser posible o brinde información útil y efectiva a quien deba atender la falla.

- **Rollback:** Mediante esta táctica se podrá revertir el sistema a un estado previo correcto, después de la detección de una falla.

- **Reintentos**: Cuando la causa de la falla se considere como transitoria, mediante la adopción de esta táctica se permitirá reintentar la operación un número determinado de veces, antes de declararla como una falla permanente.

- **Prevención de fallas**

La prevención de fallas permite que el sistema en vez de detectar la falla y tratar de recuperarse de ella, pueda prevenir que ocurra. La táctica seleccionada para este aspecto fue:

- **Manejo de Transacciones:** Esta táctica permite que los intercambios de información asincrónica entre los componente distribuidos, sean atómicos, consistentes, aislados y durables, es decir, tengan las “propiedades ACID” por sus siglas en ingles. De esta forma se garantiza que todas las operaciones que hacen parte de la

transacción se realicen exitosamente, o que ninguna de estas persista; que si los datos son consistentes antes de que comience la operación, entonces lo serán también después de que finalice la transacción; que los efectos de una transacción que se está realizando se encuentran ocultos de todas las demás transacciones; y finalmente, que cuando una transacción finaliza todos los resultados son persistentes y que sobrevivirán a un fallo del sistema.

6.1.2. Interoperabilidad

La interoperabilidad trata el grado en el cual dos o más sistemas pueden intercambiar, útilmente, información significativa mediante sus interfaces en un contexto en particular. En este sentido, la interoperabilidad requiere que, además de un adecuado intercambio de datos a nivel sintáctico, se interprete correctamente la información intercambiada, es decir, a nivel semántico.

Las razones por las cuales se debió pensar seriamente en la interoperabilidad para este proyecto, se sustentan en el hecho de que el sistema en cuestión debe proveer unos servicios que podrán ser usados por diversos sistemas, y adicionalmente debe integrarse a la plataforma existente de intercambio de electrónico de datos de Carvajal Tecnología y Servicios; el Centro Electrónico de Negocios Transaccional.

Para determinar los escenarios de interoperabilidad que permitieran definir las tácticas a utilizar en este contexto, se emplearon los siguientes valores:

Tabla 4. Valores empleados en la descripción de los escenarios de Interoperabilidad

Componente Del Escenario	Valores Empleados
Fuente	Sistemas internos de la organización: CENT(Sistema transaccional de documentos), SIGO (Sistema para la gestión operativa y de contenido), SGC (Sistema de gestión de Costos) Sistemas externos: Sistemas ERP, dispositivos móviles, navegadores web.
Estímulo	Intercambio de información, intercambio de documentos.
Artefacto	e-Business Suite

Ambiente	Sistemas con los que se desea interactuar son descubiertos en tiempo de ejecución. Sistemas con los que se desea interactuar son conocidos previo a tiempo de ejecución.
Respuesta	Peticiones adecuadamente rechazadas y notificaciones a las entidades correspondientes Peticiones apropiadamente aceptadas e información intercambiada satisfactoriamente Peticiones registradas (log) por e-Business Suite.
Medición De La Respuesta	Porcentaje de intercambio de información correctamente procesadas Porcentaje de intercambio de información correctamente rechazada

La interoperabilidad fue uno de los factores claves para determinar el estilo arquitectural predominante que será presentado más adelante. Sin embargo, dentro de las dos categorías de tácticas de interoperabilidad; localización y manejo de interfaces, la táctica que se seleccionó en este sentido para el diseño arquitectural fue:

- **Manejo de Interfaces:**
    - **Orquestación:** Esta táctica propone el uso de un mecanismo de control para coordinar, administrar y secuenciar la invocación de los diversos servicios involucrados en la solución. Se considera necesaria la orquestación, toda vez que algunas de las tareas que se requieren realizar dependen de la interacción de varios componentes internos y externos al sistema.

6.1.3. Modificabilidad

La modificabilidad es el atributo de un sistema que representa la capacidad de éste hacia el cambio, por lo tanto, es factor determinante para la estimación del riesgo y costo de implementarlo.

¿Qué puede cambiar?, ¿Cuál es la probabilidad que ese cambio surja?, ¿Cuándo sería realizado?, ¿Quién lo realizaría?, y ¿Cuál es su costo? Fueron algunos cuestionamientos claves que permitieron orientar el análisis en este sentido.

Como se mencionó anteriormente, la poca modificabilidad de la e-Business Suite es precisamente lo que motivo a este proyecto, puesto que, el cambio es una constante en este medio cuyo manejo, actualmente, está generando muchas dificultades. Adicionalmente, con la nueva visión del sistema como una solución SaaS, se pretende que éste pueda soportar la adición de nuevas funcionalidades; como la generación de otros documentos, por ejemplo, la orden de compra, de forma rentable.

Las tácticas de modificabilidad se seleccionaron de acuerdo a los escenarios elaborados para los atributos de calidad en este contexto. Los escenarios contemplaron la relación de diversos valores para cada uno de sus componentes, entre los que se destacan de manera general:

Tabla 5. Valores empleados en la descripción de los escenarios de Modificabilidad

Componente Del Escenario	**Valores Empleados**
Fuente	Cliente (fabricante u operador logístico), usuarios finales, Consultores de procesos de vertical consumo masivo e industria de CT&S
Estímulo	Directriz de agregar, eliminar o modificar una funcionalidad, cambiar un atributo de calidad o tecnología
Artefacto	Código Fuente, datos, interfaces, componentes, recursos o configuraciones
Ambiente	Tiempo de ejecución, tiempo de desarrollo, tiempo de diseño
Respuesta	Realizar la modificación, probar la modificación o implementar la modificación.
Medición De La Respuesta	Costo en termino de: Número, tamaño o complejidad de los artefactos afectados Esfuerzo Tiempo Calendario Dinero Impacto a otros funcionalidades o atributos

El objetivo de las tácticas de modificabilidad es poder controlar la complejidad de realizar un cambio, que en otros términos se traduce como también el tiempo y el costo que tomaría.

El nivel de modificabilidad de un sistema se encuentra determinado, básicamente, por el nivel de cohesión y acoplamiento de sus componentes. El acoplamiento, puede ser entendido, como la medida que indica la probabilidad que existe de implementar un cambio en una responsabilidad del sistema y que éste tenga un impacto sobre más de un módulo; en otros términos, cuando un cambio sobre una responsabilidad afecta más de un componente, resulta generalmente, mucho más costoso que si éste solo impactara a uno solo.

De igual modo, la cohesión permite medir que tan relacionadas se encuentran las responsabilidades de un módulo, es decir, el nivel de unidad de propósito del módulo. En otros términos, la cohesión representa la probabilidad de que un cambio en una responsabilidad del sistema afecte otras responsabilidades, por lo tanto al aumentar la cohesión disminuye la probabilidad de que un cambio impacte múltiples responsabilidades.

Las tácticas de modificabilidad seleccionadas para este proyecto se pueden categorizar de la siguiente forma:

- **Reducción del Tamaño del Módulo**
    - **Dividir los Módulos:** Está táctica propone que los módulos no deberían tener un gran número de capacidades, puesto que realizar cualquier modificación sobre éstos será, muy probablemente, más costosa. Un módulo que se refina en otros más pequeños debe reducir el costo promedio de realizar un cambio en el futuro.
- **Aumento de Cohesión**
    - **Aumentar la Coherencia Semántica:** El objetivo de adoptar esta táctica es evitar que dos responsabilidades que no se relacionan en un mismo propósito sean asignadas a un mismo módulo del sistema. Esta relación se puede identificar entre dos responsabilidades, realizando el movimiento hipotético de una de éstas y analizando si la otra es o no afectada. Si se observa que no hay ningún impacto,

entonces las dos responsabilidades no debería estar en el mismo lugar.

- **Reducción de Acoplamiento**

  - **Encapsular:** Mediante la encapsulación se introducen interfaces explícitas a los módulos, de acuerdo a sus responsabilidades. De esta forma se reduce la probabilidad de que un cambio en un módulo se propague a otros. Al adoptar esta táctica, los puntos fuertes de acoplamiento ya no se generan entre los módulos, sino entre sus interfaces; además, éstos se reducen puesto que las interfaces limitan las formas como responsabilidades externas pueden interactuar con los módulos, es decir, que sólo se podrá interactuar con los módulos a través de las interfaces que exponen. Para lograr este efecto, se debe tener en cuenta que las interfaces deben ser abstractas con respecto a los detalles internos que tienen la mayor probabilidad de cambiar.

  - **Usar intermediarios:** El uso de intermediaros permite romper algunas dependencias; por ejemplo, los patrones de publicación y subscripción, o más precisamente, los mecanismos de mensajería, permiten, que quien produce la información no requiera conocer de quienes van a consumirla.

  - **Abstracción de Servicios Comunes:** Esta táctica establece que en los casos donde dos o más módulos provean servicios similares, resulta más rentable implementar un servicio en una forma abstracta más general; por ejemplo, haciendo uso de la parametrización, de tal forma que cuando se requiera realizar un cambio al servicio se deba realizar una sola vez.

- **Posposición de la Asociación de Datos:**

- Esta táctica, en consecuencia de las anteriores, propone que debe procurarse, si resulta rentable, establecer un mecanismo que permita

establecer los valores de los servicios que sean parametrizables lo más lejos posible dentro del ciclo de vida del software, de tal forma que no se requiera fijarles valores particulares en etapas tempranas.

- Como ejemplos de estas tácticas encontramos:
    - En tiempo de compilación: Parametrización en tiempo de compilación, aspectos.
    - En tiempo de Implementación: Asociación de valores en tiempo de configuración.
    - En tiempo de inicialización: Archivos de recursos
    - En tiempo de ejecución: Interpretación de parámetros, asociación de valores en tiempo de inicio, publicación y suscripción, repositorios conjuntos.

6.1.4. Rendimiento

El rendimiento corresponde a la habilidad de un sistema para cumplir con los requisitos respecto al tiempo, es decir, el tiempo que puede tardar el sistema para generar una respuesta determinada ante un evento. Las cuestiones respecto al rendimiento giran, básicamente, en torno a la administración de los recursos del sistema. El rendimiento puede ser medido en términos del procesamiento y la latencia, sin embargo, dependiendo del contexto del sistema uno de estos dos criterios puede ser de mayor importancia.

El rendimiento, ha sido uno de los factores clásicos que ha predominado en la definición de las arquitecturas de software, hasta el punto que su logro, frecuentemente, solía impedir el de otros atributos de calidad; sin embargo, gracias a la caída de la relación precio sobre rendimiento del hardware, ha permitido que surjan otros atributos como competidores importantes del rendimiento.

Por otra parte, cabe resaltar que el rendimiento tiene una estrecha relación con la escalabilidad, la cual determina la capacidad que puede ser

incrementada en un sistema de tal forma que éste continué operando en las condiciones de tiempo requeridas.

Las tácticas de rendimiento se seleccionaron de acuerdo a los escenarios elaborados para los atributos de calidad en este contexto. Los escenarios contemplaron la relación de diversos valores para cada uno de sus componentes, entre los que se destacan de manera general los siguientes:

Tabla 6. Valores empleados en la descripción de los escenarios de Rendimiento

Componente Del Escenario	**Valores Empleados**
Fuente	Interna o externa al sistema
Estímulo	Llegada de eventos periódicos o esporádicos
Artefacto	El sistema, uno o más de sus componentes.
Ambiente	Modo de operación normal
Respuesta	Procesar los eventos, cambiar el nivel de servicio
Medición De La Respuesta	Latencia o rendimiento

El tiempo que un sistema toma en responder a un evento es la suma de la contribución entre el tiempo de procesamiento y el tiempo que éste se pueda estar bloqueado. Durante el tiempo de procesamiento, los componentes que atienden el evento consumen diversos tipos de recursos como CPU, almacenamiento, ancho de banda y memoria, o también otras entidades propias del sistema en diseño. El comportamiento de estos recursos está sujeto a su capacidad y su nivel de demanda en un momento dado; por lo tanto, su rendimiento puede verse significativamente degradado cuando se encuentran altamente cargados. Por otra parte, cuando alguno de estos recursos no se encuentra disponible, o el procesamiento requiere el resultado de alguna otra computación que no se ha realizado, el sistema entra, entonces, en tiempo de bloqueo.

Las tácticas de rendimiento que se seleccionaron para este proyecto, de tal forma que se pudiera controlar la demanda y administrar los recursos son:

- **Controlar la Demanda de Recursos**

    - **Priorizar Eventos:** Está táctica propone categorizar los eventos que llegan al sistema de acuerdo al nivel de importancia que representan, de tal forma que se pueda definir un modelo de atención donde los eventos más importantes tengan mayor prioridad.

      La priorización de eventos es un aspecto clave en el contexto de este proyecto, puesto que, Carvajal Tecnología y Servicios establece diversos tipos de acuerdos de nivel de servicio con sus clientes de acuerdo a diversos criterios propios del negocio, por ejemplo, para las empresas cuyos los documentos corresponden a la comercialización de alimentos que pueden perecer rápidamente.

- **Administrar los Recursos**

    - **Aumentar Recursos:** Por su nombre se puede intuir que lo que esta táctica propone es aumentar los recursos, es decir, adquirir procesadores más rápidos o adicionales, más memoria y redes más rápidas cuando se considere necesario; en vista de que recursos como el hardware son comúnmente la forma más económica de adquirir mejoras de rendimiento inmediatas.

    - **Introducir Concurrencia:** La concurrencia permite que varias peticiones puedan ser atendidas en paralelo y por consiguiente disminuir significativamente los tiempos de bloqueo. Es importante establecer un esquema de planificación justo de tal forma que se pueda existir una equidad entre las solicitudes que se reciben al mismo tiempo

6.1.5. Seguridad

La seguridad es la medida de la habilidad de un sistema para proteger la información del acceso no autorizado, a su vez proporcionando el acceso a las entidades que si se encuentran autorizadas. Cualquier acción que sea realizada en contra de un sistema con la intensión de hacerle daño es denominada como un ataque; no obstante estos ataques pueden ser de diversos tipos; por ejemplo, un intento por acceder a datos o servicios no autorizados, o por modificar datos, o por denegar el acceso a los servicios a usuarios legítimos.

Existen tres aspectos que permiten caracterizar generalmente la seguridad de un sistema, estos son: la confidencialidad, la integridad y la disponibilidad. La confidencialidad es la propiedad que representa el nivel de protección de los datos o servicios de accesos no autorizados. De igual modo, la integridad es la propiedad que representa que los datos o servicios no sean sujetos de manipulación no autorizada. Finalmente, la disponibilidad, un atributo que como se mencionó anteriormente, tiene mucha relación con la seguridad, representa que el sistema esté disponible para usuarios legítimos.

Análogamente a la metodología del análisis del árbol de fallas, presentada anteriormente, en el área de seguridad se recomienda realizar un árbol que permita determinar posibles amenazas. Como raíz del árbol de parte de un ataque que ha sido exitoso y en sus nodos se establecen las causas posibles de dicho ataque y así sucesivamente.

Las tácticas de seguridad se seleccionaron de acuerdo a los escenarios elaborados para los atributos de calidad en este contexto. Los escenarios contemplaron la relación de diversos valores para cada uno de sus componentes, entre los que se destacan de manera general los siguientes:

Tabla 7. Valores empleados en la descripción de los escenarios de Seguridad

Componente Del Escenario	Valores Empleados
Fuente	Humano por fuera de la organización, Sistema externo

Estímulo	Acceso no autorizado para capturar información, modificarla o eliminarla. Acceso no autorizado a los servicios del sistema para modificar su comportamiento o reducir su disponibilidad
Artefacto	Servicios del sistema Datos del Sistema
Ambiente	En línea
Respuesta	Datos y servicios protegidos de accesos no autorizados. Datos y servicios no manipulados sin autorización. Identificación de las partes que interactúan con el sistema. Partes del sistema no pueden repudiar sus implicaciones Datos y servicios disponibles a usuarios legítimos Grabación del acceso o intentos de acceso a recursos, datos y servicios del sistema. Notificación a las entidades correspondientes cuando un aparente ataque ocurra.
Medición De La Respuesta	Porcentaje de sistema comprometido cuando un componente o dato en particular queda comprometido. Tiempo que transcurre antes de que un ataque sea detectado. Número de ataques resistidos. Tiempo de recuperación después de un ataque exitoso. Cantidad de datos vulnerables a un ataque en particular.

Las tácticas de seguridad suelen ser agrupadas en las siguientes cuatro categorías:

- **Detectar Ataques**
    - **Verificar la Integridad de los Mensajes:** Esta táctica propone el uso de técnicas como el uso de sumas de verificación (Checksums) o valores Hash para verificar la integridad de los mensajes, archivos de recursos y archivos de configuración, entre otros.
    - **Detectar el retraso de los Mensajes:** Mediante esta táctica es posible detectar ataques donde alguna entidad maliciosa esté interceptando y posiblemente modificando los mensajes. El objetivo es analizar el tiempo que se toma en entregar los mensajes para determinar si existe algún comportamiento sospechoso.
- **Resistir Ataques**

- **Identificar Actores:** Esta táctica establece que se debe identificar la fuente de cualquier entrada externa al sistema. Para esto se pueden utilizar identificaciones de usuario, códigos de acceso, direcciones IP, protocoles, puertos, entre otros.

- **Autenticar Actores:** El objeto de esta táctica es asegurarse de que un actor es realmente quien o lo que pretende ser. Contraseñas, contraseñas de un solo uso, certificados digitales, identificación biométrica son algunos medios que permiten autenticar los actores.

- **Autorizar Actores:** Mediante la adopción de esta táctica se asegura que un actor autenticado tiene los derechos de acceder y modificar los datos o servicios. En otros términos, la autorización de actores representa el mecanismo de control de accedo dentro del sistema.

- **Encriptar Datos:** Está es la forma permite proteger la información de accesos no autorizados. Se logra, normalmente, mediante la aplicación de alguna forma de encriptación a los datos o a su comunicación. La encriptación de datos puede ser concebida como una capa extra de protección a los datos más alla de la autorización.

- **Reaccionar a los ataques**

  - **Revocar Acceso:** Esta táctica está destinada a responder a los ataques. El sistema o alguno de sus administradores puede limitar el acceso a un actor, que inclusive pude ser legítimo, si se cree que está ejecutando un ataque.

  - **Informar Actores:** Es indispensable que cuando el sistema logre detectar que se está ejecutando un ataque, se le notifique a las entidades correspondientes para que éstos puedan ejecutar las acciones necesarias en defensa del sistema.

- **Recuperarse de los Ataques**

- **Mantener Registro de Auditoría:** Este registro es un conjunto de información sobre las acciones ejecutadas por los actores en el sistema y sus efectos, que permite monitorear e identificar un atacante, como también, mejorar las barreras de defensa.

6.1.6. Capacidad de Prueba

La capacidad de prueba hace referencia a la facilidad con la cual se pueden demostrar los fallos en un sistema a través de la ejecución de pruebas. En otros términos, representa la probabilidad; asumiendo que el sistema tiene por lo menos una falla, que esta será descubierta en la siguiente ejecución de las pruebas. Hasta que la industria no indico que entre el 30 y 50 por ciento del costo de desarrollar sistemas con alto nivel de calidad, es consumido por pruebas, este atributo de calidad no tuvo la misma atención que otros como la modificabilidad, el rendimiento o la disponibilidad. Por lo tanto, cualquier aporte que el arquitecto brinde para reducir el alto costo de realizar las pruebas puede producir un alto beneficio.

La ejecución de las pruebas no solo implica la verificación de las salidas del sistema de acuerdo a su especificación, también abarca aspectos adicionales; derivados de otros atributos de calidad, como el tiempo que toma en producir dicha salida. En este sentido, para que un sistema pueda ser debidamente verificable, debe ser posible controlar las entradas de cada componente, observar sus salidas, pero adicionalmente, manipular y observar su estado interno, durante y después de la ejecución de las pruebas. En síntesis, el control y la observación son dos características que determinan la capacidad de prueba de un sistema.

Las tácticas de capacidad de prueba se seleccionaron de acuerdo a los escenarios elaborados para los atributos de calidad en este contexto. Los escenarios contemplaron la relación de diversos valores para cada uno de sus componentes, entre los que se destacan de manera general los siguientes:

Tabla 8. Valores empleados en la descripción de los escenarios de Capacidad de Prueba

Componente Del Escenario	**Valores Empleados**
Fuente	Pruebas unitarias, pruebas de integración, pruebas de aceptación, ejecutados manualmente o mediante herramientas de automatización de pruebas.
Estímulo	Conjunto de pruebas ejecutadas
Artefacto	La porción del sistema que está siendo verificada.
Ambiente	Tiempo de Desarrollo. Tiempo de Integración. Tiempo de Implementación.
Respuesta	Capturar los resultados. Capturar la actividad que resultó en una falla. Controlar y monitorear el estado del sistema.
Medición De La Respuesta	Esfuerzo para encontrar una falla. Esfuerzo para alcanzar un determinado porcentaje de cobertura Probabilidad de descubrir una falla en la siguiente ejecución de una prueba. Tiempo para ejecutar las pruebas. Longitud de la cadena de dependencias más larga in las pruebas. Tiempo para preparar el ambiente de pruebas.

Básicamente existen dos categorías para las tácticas de la capacidad de prueba, estas son:

- **Controlar y Observar el Estado del Sistema:**
    - **Interfaces especializadas:** Ésta táctica permite controlar o capturar el valor de las variables de un componente a través de una herramienta de prueba o por medio de la ejecución normal. Ejemplos de interfaces especializadas son los siguiente: métodos *set* y *get* para variables, modos o atributos importantes; métodos de *reporte* (*report*) que retorne todo el estado del objeto; método de *restablecer* (*reset*) que permita establecer el estado interno en uno en específico; método para activar una salida detallada de información log de eventos y monitoreo de recursos.

- **Limitar Complejidad:**

    - **Limitar Complejidad Estructural:** Teniendo en cuenta que el software complejo es más difícil de probar; puesto que el espacio que ocupan sus estados es muy amplio, lo que dificulta mucho poder recrearlos; y que el objetivo de probar un sistema no sólo es encontrar sus fallas, si no, encontrar el error que causó la falla para que pueda ser removido; hacen que la adopción de esta táctica sea muy conveniente. Básicamente, esta táctica promueve: evitar o resolver las dependencias cíclicas entre los componentes, asilar o encapsular dependencias sobre el medio ambiente externo, y reducir dependencias entre los componentes en general.

6.1.7. Usabilidad

La usabilidad representa el grado de facilidad con el cual un usuario puede completar una tarea deseada y el tipo de soporte que el sistema le provee. La usabilidad comprende los siguientes aspectos: la facilidad con la que un usuario puede aprender una característica del sistema, el nivel de eficiencia que obtiene operando el sistema, la minimización de errores en los que puede incurrir, el nivel de adaptación a sus necesidades, la capacidad de incrementar el nivel de confianza y satisfacción del usuario.

Como se había mencionado anteriormente, la usabilidad es uno de los atributos de calidad que mayor interés y prioridad tiene para las partes interesadas al interior de la organización; porque la experiencia ha demostrado que el aprendizaje de la e-Business Suite actual ha generado grandes dificultades a la comunidad de usuarios, lo que ha implicado realizar capacitaciones contantemente y atender un amplio número de llamadas al servicio de soporte técnico, en este sentido.

Las tácticas de usabilidad se seleccionaron de acuerdo a los escenarios elaborados para los atributos de calidad en este contexto. Los escenarios contemplaron la relación de diversos valores para cada uno de sus componentes, entre los que se destacan de manera general los siguientes:

Tabla 9. Valores empleados en la descripción de los escenarios de Usabilidad

Componente Del Escenario	**Valores Empleados**
Fuente	Usuario Final
Estímulo	Un intento de usar el sistema eficientemente. Aprendizaje del uso del sistema. Minimización del impacto de errores. Adaptación del sistema. Configuración del sistema.
Artefacto	Portal web de acceso a servicios de e-Business Suite.
Ambiente	Tiempo de ejecución
Respuesta	Proveer al usuario con la característica que necesita. Anticiparse a las necesidades del usuario.
Medición De La Respuesta	Tiempo que toma realizar la tarea. Número de errores. Número de tareas completadas. Grado de satisfacción del usuario.

Las investigaciones respecto a la interacción humano-computador suelen usar los términos *iniciativa del sistema, iniciativa del usuario,* e *iniciativa mixta* para describir quien dentro de la pareja humano-computador toma la iniciativa en realizar alguna acción y como continua la interacción.

La usabilidad tiene una estrecha relación con la modificabilidad del sistema respecto a interfaz de usuario; ya que es de suma importancia que el diseño facilite la experimentación con la interfaz; de tal forma que se puedan incluir cambios rápidamente a partir de la retroalimentación. Para lograr esto, la interfaz tiene que ser lo más independientemente posible de los demás componentes del sistema. Esta relación evidencia que a diferencia de los demás atributos de calidad que suelen estar en conflicto, la usabilidad y la modificabilidad se complementan, porque una de las mejores formas para hacer un sistema más usable es hacerlo modificable.

Existen muchas técnicas que permiten aumentar la usabilidad en un sistema sin embargo, dos tácticas generales que fueron adoptadas en este proyecto son las siguientes:

- **Apoyar la Iniciativa del Usuario:**

La usabilidad puede ser incrementada si se le brinda al usuario una retroalimentación respecto a lo que el sistema realiza cuando está en ejecución, de tal forma que éste pueda realizar acciones adecuadas. Por ejemplo, incluir características como cancelar, deshacer, pausar y reanudar apoyan al usuario a que la corrección de errores o ser más eficiente.

- **Apoyar la Iniciativa del Sistema:**

Esta táctica propone la inclusión de un modelo de tareas, modelo de usuarios o modelo del sistema, de tal forma que le permitan predecir su comportamiento o la intensión del usuario y así tomar iniciativas que le permitan asistir al usuario.

6.2. Patrones Arquitecturales

Los patrones y las tácticas arquitecturales son conjuntos de decisiones que representan la forma como se consolidan las buenas estructuras de diseño, que han sido probadas en la práctica, y que pueden ser reutilizadas. Sin embargo, es imposible listar un conjunto finito de patrones, dado que éstos surgen espontáneamente de acuerdo a la relación que tengan con las condiciones del ambiente del sistema a diseñar; de este modo, cada que existe un cambio en dichas condiciones surgirá un nuevo patrón.

Normalmente un diseño arquitectural no surge a partir principios. Los arquitectos más experimentados suelen ejecutar este proceso como una selección, ajuste y combinación patrones. La clave se encuentra en la forma como se instancian los patrones en contextos específicos y de acuerdo a las restricciones del problema. De esta forma se evidencia que los patrones, a diferencia de las tácticas, están destinados a solventar más de un aspecto o necesidad que debe soportar el sistema. No obstante, cabe aclarar, que casi todos los patrones son el resultado de la combinación de más de una táctica.

Los patrones arquitecturales pueden ser entendidos de mejor forma si se expresan como una relación entre un contexto, un problema y una solución. El contexto representa una situación común que generalmente conduce a un problema en particular; el problema, de forma general, representa los atributos de calidad que se buscan alcanzar; y la solución, describe un diseño arquitectural abstracto que ha sido exitoso resolviendo el problema. La solución debe describir las responsabilidades, las relaciones, el comportamiento o la interacción entre los elementos que componen el diseño. Adicionalmente un patrón debe ser muy claro en la forma como la configuración de los elementos permite lograr los atributos de calidad enmarcados en el problema.

De acuerdo a las tácticas identificadas previamente, para lograr los atributos de calidad requeridos en este proyecto, se seleccionaron y ajustaron varios patrones de diseño, como se realiza comúnmente en el diseño de sistemas

complejos como éste. A continuación se presentará una breve descripción de cada uno en la cual se evidencian las razones de su adopción.

6.2.1. Capas (Layer Pattern)

El patrón de capas, uno de los más comunes en la ingeniería de software, se adopta cuando se requiere desarrollar, mantener y evolucionar las partes del sistema independientemente. Este patrón, que otorga portabilidad, modificabilidad y reusabilidad a cambio de algo de rendimiento, establece que para lograr este propósito debe existir una separación muy clara entre cada uno de los aspectos del sistema, de tal forma que las modificaciones a un módulo tengan poca interacción con los demás.

El sistema, entonces, debe dividirse en unidades, las cuales se denominan capas, y se caracterizan por ofrecer un conjunto cohesivo de servicios mediante una interfaz de uso público. Adicionalmente, la implementación de este patrón impone una restricción en la relación de uso entre las capas, ya que ésta debe ser unidireccional. En otros términos, las capas son construidas para interactuar de acuerdo a una relación de orden estricta.

Si determinamos tres capas adyacentes A, B y C, donde A es la capa superior, ésta podrá usar todos los servicios provistos por la capa B; de igual forma B podrá utilizar todos los servicios de B. En ciertas circunstancias A, requerirá interactuar directamente con C, cuestión que aunque no es recomendada es permitida en éste patrón. A dicho salto se conoce como capa de puenteo. Si ésta situación se presenta frecuentemente en un diseño, se estará perjudicando la portabilidad y modificabilidad del sistema. Sin embargo, este patrón nunca permitirá usos hacia arriba, es decir, que C no podrá usar servicios de A o B, por ejemplo.

6.2.2. Intermediario (Broker Pattern)

El patrón Intermediario, más conocido por su nombre en inglés, Broker, se emplea cuando se diseñan sistemas que ofrecerán una colección de servicios en

múltiples servidores, porque, permite establecer el mecanismo como se operará con dichos servicios, y la forma como se garantizará su disponibilidad.

Este patrón propone una forma de estructurar sistemas distribuidos, la cual consiste en la asignación de un componente intermediario entre los usuarios y los servicios del sistema. El intermediario tiene la responsabilidad de localizar el proveedor del servicio adecuado para cumplir las peticiones de los usuarios, redirigirlas hacia el servidor determinado y retornar los resultados. Los usuarios, por lo tanto, no conocerán las características, localización e identidad de los proveedores del servicio; permitiendo que fácilmente se puedan cambiar, dinámicamente, los enlaces entre éstos y los proveedores del servicio.

Los beneficios de la adopción de este patrón se ven reflejados en la modificabilidad, la disponibilidad y el rendimiento. La modificabilidad, puesto que, como se expresaba en las tácticas, al introducir un intermediario se rompen algunas dependencias entre los componentes del sistema, lo que resulta muy favorable para disminuir el acoplamiento. La disponibilidad, ya que si un alguno de los servidores llega a quedar por fuera de servicio, un remplazo puede ser asignado dinámicamente por el Broker de tal forma que el cliente no se vea afectado, como lo proponen las tácticas de redundancia. Y el rendimiento, puesto que el Broker permite la asignación de trabajo a los servidores que estén menos congestionados.

No obstante, este patrón también acarrea algunas desventajas como lo son: el nivel adicional de complejidad que se le adicional al sistema, y el nivel de indirección que se le adiciona entre el cliente y el servidor, lo cual puede introducir latencia en la comunicación. Adicionalmente, se corre el riesgo que un Broker que no sea bien diseñado pueda ser el punto único de falla de un amplio sistema, o que se convierta en un cuello de botella para la comunicación.

6.2.3. Modelo Vista Controlador (IMVC Pattern)

Las interfaces de usuario son generalmente las partes de los sistemas de software que se modifican con mayor frecuencia en las aplicaciones interactivas; por lo tanto, es de suma importancia que éstas se encuentren separadas del resto del sistema. El patrón Modelo Vista Controlador, MVC, es una alternativa que ha tenido una gran acogida, la cual establece una separación de la funcionalidad de la aplicación en tres tipos de componentes: el Modelo, la Vista y el Controlador.

El modelo es el componente que contiene la información de la aplicación. La vista es quien muestra los datos del modelo que se requieran, y quien interactúa con el usuario. El controlador, por su parte, es quien media entre el modelo y la vista, administrando las interacciones entre éstos mediante notificaciones correspondientes a los cambios de estado. Estos tres componentes se conectan unos a otros mediante algún tipo de notificación, como eventos o *callbacks*. Un cambio en el modelo necesita ser comunicado a la vista de tal forma que ésta pueda ser actualizada. Un evento externo, como una interacción de un usuario, debe ser comunicado al controlador, el cual, debería en su turno, actualizar la vista o el modelo

Dado que los componentes del MVC terminan siendo débilmente acoplados, si se ha implementado adecuadamente el patrón, es fácil desarrollarlos y probarlos en paralelo, puesto que los cambios que se realicen en un componente tendrán un mínimo impacto en los demás.

En sistemas cuyas interfaces de usuario no sean complejas; lo cual no sería el caso de este proyecto, puede que la adopción de este patrón no sea lo más recomendable o no tenga sentido, dado el costo de su diseño e implementación.

6.2.4. Cliente Servidor (Client Server Pattern)

El patrón cliente-servidor es uno de los más clásicos dentro de los empleados en este proyecto. Este patrón establece, como su nombre lo indica, la

estructuración del diseño en componentes denominados clientes y servidores, cuyas funciones, básicamente, pueden ser intuidas a partir de sus nombres. Es el patrón ideal cuando existen recursos compartidos y servicios que un gran número de clientes distribuidos quieren acceder, y cuando, adicionalmente, se debe controlar el acceso a éstos y la calidad de los servicios.

La interacción entre los clientes y servidores se realiza mediante peticiones de los servicios ofrecidos por los éstos, que son iniciadas por los clientes, y respuestas de los servidores a estas peticiones, que regresadas a los clientes con sus resultados correspondientes. Los servicios pueden estar centralizados en un gran servidor o en muchos servidores distribuidos, y la conexión entre estos puede ser implementada mediante cualquier tipo de conector de petición/respuesta (*request*/*reply*), como lo es el protocolo HTTP, por ejemplo.

El flujo computacional de los sistemas cliente-servidor es asimétrico, los clientes inician la interacción invocando servicios a los servidores, por lo tanto, el cliente debe conocer la identidad de los servicios para poder invocarlos. En contraste, los servidores no conocen la identidad de los clientes en momentos previos a la petición del servicio y solo responden a las peticiones iniciadas por los clientes. Por otra parte, es importante resaltar que los servidores, a su vez, pueden ser clientes de otros servicios.

El patrón cliente-servidor es el común a todas las aplicaciones web donde los clientes corresponden a los navegadores web. Como también, el de los sistemas de información implementados sobre redes, donde el servidor corresponde a un sistema de administración de base de datos (DBMS).

Son múltiples las ventajas que se obtiene de adoptar ese patrón que separa las aplicaciones cliente de los servicios que usa. Por ejemplo, de esta forma, se pueden simplificar los sistemas mediante la refactorización de los servicios comunes, lo que permite que, cuando se requieran modificar, sólo deba realizarse en una sola localización o en un pequeño número de éstas. Adicionalmente, permite que estos servicios comunes sean reutilizables. Por

otra parte, dado que los servidores pueden ser accedidos por cualquier número de clientes, resulta relativamente fácil agregar nuevos clientes al sistema. Así mismo, los servidores pueden ser replicados para soportar escalabilidad y disponibilidad.

No obstante, el patrón cliente servidor conlleva algunas desventajas. El servidor, por ejemplo, puede ser un cuello de botella respecto al rendimiento y puede ser el punto único punto de falla de un gran sistema. Por otra parte, movilizar funcionalidades del cliente hacia el servidor o viceversa puede ser bastante complejo y costoso después de que el sistema se ha construido.

6.2.5. Arquitectura Orientada a Servicios (SOA Pattern)

El patrón de Arquitectura Orientada a Servicios es adoptado cuando se cuenta con servicios, que son ofrecidos y descritos por proveedores, y consumidos por clientes que necesitan poder entender y usar esos servicios sin tener ningún conocimiento detallado de su implementación. De esta forma, SOA ofrece la posibilidad de generar la interoperabilidad requerida sobre un conjunto componentes distribuidos, que quizás, pueden estar presentes en diferentes plataformas, o escritos en diferentes lenguajes de implementación, o ser provistos por diferentes organizaciones, y ser distribuidos a través del internet; logrando un rendimiento, seguridad y disponibilidad razonable.

Son varios los componentes que pueden pertenecer a un diseño bajo el patrón SOA. Sin embargo, los proveedores de servicio y los consumidores de éstos son los componentes ineludibles. Los proveedores de servicio, proveen uno o más servicios a través de sus interfaces públicas; aspectos importantes como su rendimiento, limitantes y demás propiedades, en ocasiones suelen especificarse en los acuerdos de nivel de servicio (SLA). Por otra parte los consumidores de servicios, invocan los servicios directamente o a través de un intermediario.

Adicional a estos dos componentes mencionados anteriormente, en las aplicaciones SOA, se emplean componentes especializados que actúan como intermediarios y que proveen servicios de infraestructura; los cuales, permiten

elaborar, los flujos de trabajo requeridos. Dentro de estos componentes se encuentran: los buses de servicios de empresa (ESB), que pueden direccionar y transformar mensajes entre los proveedores y consumidores de servicios; los registros de servicios, que pueden ser usados por los proveedores de servicios para registrar sus servicios y por los consumidores para descubrir estos servicios en tiempo de ejecución; los servidores de orquestación, que coordinan la interacción entre los consumidores y proveedores, basados en lenguajes de procesos de negocios o flujos de trabajo.

Todos estos componentes mencionados se conectan mediante diversos tipos de conectores. Algunos de los más destacados en este contexto son los siguientes: SOAP, el cual se basa en el protocolo SOAP para la comunicación asincrónica entre los servicios web, típicamente funcionando sobre HTTP; REST, el cual se basa en las operaciones básicas (POST, GET, PUT, DELETE) de petición y respuesta del protocolo HTTP, para informar al proveedor del servicio que debe crear, actualizar o eliminar un recurso; Mensajería Asincrónica, la cual usa un sistema de mensajería para el intercambio asíncrono de mensajes, mediante comunicación punto a punto o de publicación y suscripción.

Se puede observar que el patrón SOA comparte muchos de los conceptos de diseño y objetivos del patrón Broker; sin embargo, el mayor beneficio de SOA es la interoperabilidad. No obstante, los sistemas SOA son mucho más complejos de construir, y cuentan con la desventaja de no poder tener un control sobre la evolución de los servicios independientes. Adicionalmente, SOA impone una sobrecarga de rendimiento asociada al middleware. Así mismo, no siempre es posible contar con el nivel de servicio que se requiere para cada uno de los servicios a integrar

6.2.6. Multinivel (Multi tier Pattern)

El patrón Multinivel es empleado cuando existe la necesidad de distribuir la infraestructura de un sistema en distintos subconjuntos, como resultado de razones operacionales o de negocio; de tal forma que cada subconjunto represente una estructura de ejecución computacionalmente independiente,

que se conecta las demás por un medio de comunicación. Este patrón, entonces, permite organizar las estructuras de ejecución como un conjunto de agrupaciones lógicas de componentes; donde cada agrupación es denominada nivel, y agrupan generalmente componentes de funcionalidades similares.

La agrupación de componentes en niveles pude obedecer a una variedad de criterios, como por ejemplo, el tipo de componente, el hecho de compartir el mismo ambiente de ejecución, o el hecho de tener el mismo propósito de ejecución. El uso de niveles puede ser aplicado a cualquier colección de componentes en ejecución. Puesto que los niveles no son componentes como tal, más bien, agrupaciones de componentes.

Este patrón además de imponer la restricción de no permitir que un componente pertenezca a más de una capa, contiene una gran debilidad que se refleja en su complejidad y el costo significativo de su implementación.

Frecuentemente se suele confundir el patrón multinivel con el patrón de capas lo cual no es correcto, puesto que capas son un patrón modular, es decir de unidades de implementación, mientras que los niveles aplican solo para entidades en ejecución.

7. CONCLUSIONES

- El diseño de una arquitectura de software deliberada, mediante el seguimiento de una adecuada metodología, es clave para el éxito de un sistema; puesto que, un buen diseño permite reflejar todas las decisiones y estrategias que han sido tomadas de acuerdo a las necesidades del negocio y basándose en los principios o soluciones previas probadas, que garantizan el cumplimiento de los objetivos de negocio y el propósito real del sistema.

- El Arquitecto de Software es un rol de mucha responsabilidad, que puede ser determinante en el éxito de una solución; por lo tanto, debe ser asumido por alguien, quien además de conocer claramente el proceso para la elaboración del diseño, tenga un amplio conocimiento del contexto del negocio de la solución, y una extensa experiencia a nivel técnico y tecnológico; de tal forma, que pueda sustentar adecuadamente cada una de las decisiones que implican el diseño de una arquitectura de software.

- Una de las etapas más importantes dentro del diseño de una arquitectura es la definición de los requerimientos arquitectónicamente significativos; puesto que constituyen el sustento de todas las decisiones que se tienen que tomar posteriormente en la elaboración del diseño, y donde cada uno influencia de forma particular la arquitectura. De este mismo modo, se evidencia también, que los factores determinantes del diseño de una arquitectura no se sustentan en especialmente las características funcionales, si no en los atributos de calidad requeridos.

- Las tácticas de diseño, son las herramientas fundamentales con las que cuenta un arquitecto de software, para lograr el cumplimiento de todos los requerimientos del sistema en circunstancias donde no existan otros medios arquitectónicos, como patrones de diseño o arquitecturas de referencia, que se puedan ajustarse fácilmente a las necesidades requeridas.

- El mundo del intercambio electrónico de datos presenta modificaciones constantes, puesto que los procesos dentro de la cadena de abastecimiento son optimizados con frecuencia; por lo tanto, las aplicaciones de software que se diseñan para este contexto tienen que ser concebidas para el cambio
- El modelo de distribución de software como servicio SaaS, está tomando una gran fuerza en el mercado por ser altamente atractivo, gracias a la versatilidad que ofrece en la elaboración de modelos de negocios basados paquetes de servicios.

8. TRABAJO FUTURO

De acuerdo a la metodología planteada en la sección cuatro del presente documento; para dar cumplimiento total al ciclo de desarrollo que se propone, se requiere que la arquitectura sea implementada de tal forma que se pueda validad y ajustar; teniendo en cuenta que el diseño inicial contiene un grado de incertidumbre y que hay una probabilidad muy alta que algunos requerimientos que no se encuentran estables puedan cambiar.

Se recomienda enfáticamente realizar el desarrollo del sistema mediante el uso de una metodología iterativa que permita validar progresivamente los requerimientos que sean implementados. Para la labor de la validación se requiere contar con la colaboración de un conjunto representativo de los clientes de tal forma que la retroalimentación sea más efectiva, uno de los factores más claves del éxito. Esta observación se realiza desde la percepción personal sobre la inestabilidad de los requerimientos dado la amplitud de funcionalidades, su complejidad y las características heterogéneas de la comunidad de clientes.

9. REFERENCIAS

[1] Carvajal Tecnología y Servicios S.A.S, *Presentación Comercial e-Business Suite,* 2012.

[2] O. Vogel, I. Arnold, A. Chughtai y T. Kehrer, Software architecture: A Comprehensive Framework and Guide for Practitioners, New York: Springer, 2011.

[3] L. Bass, P. Clements y R. Kazman, Software Architecture in Practice, 3rd ed., Upper Saddle River, NJ: Addison-Wesley, 2013.

[4] The Carnegie Mellon Software Engineering Institute (SEI), «ATAM: Method for Architecture Evaluation,» Carnegie Mellon University, Hanscom AFB, 2000.

[5] Microsoft Patterns and Practices Team, Microsoft Application Architecture Guide, 2nd ed., Redmond, Wash.: Microsoft, 2009.

[6] P. Clements, Documenting Software Architectures: Views and Beyond, Upper Saddle River, NJ: Addison-Wesley, 2011.

[7] P. Kroll y P. Kruchten, The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP, Boston: Addison-Wesley, 2003.

[8] UNECE, United Nations Economic Commission For Europe, *Introducing UN/EDIFACT,* 2013.

[9] UNECE, United Nations Economic Commission For Europe, *UN/EDIFACT - Introduction and Rules - Part 1: Introduction,* 2013.

[10] EAN International, EDI Standards Manual - EANCOM® 1997 (UN/EDIFACT D.96A), 1997.

[11] GS1 International, EANCOM® 2002 Standard, Edition 2012 - Syntax 4, Part 1 ed., GS1, 2012.

[12] S. Chopra y P. Meindl, Supply Chain Management: Strategy, Planning, and Operation, 5th ed., Boston: Pearson, 2012.

[13] I. Sommerville, Software Engineering, 9th ed., Boston: Pearson, 2011.

[14] R. S. Pressman, Software Engineering: A Practitioner's Approach, 7th ed., New York: McGraw-Hill, 2010.

[15] J. A. O'Brien, Management Information Systems, 10th ed., New York: McGraw-Hill/Irwin, 2011.

[16] E. Hull, Requirements Engineering, 3rd ed., London ; New York: Springer, 2011.

[17] United Nations Economic Commission for Europe, *Despatch Advice DESADV UN/EDIFACT Standard,* 2000.

[18] United Nations Economic Commission for Europe, *Price Catalog PRICAT UN/EDIFACT Standard,* 2000.

[19] GS1 International, *Serial Shipping Container Code (SSCC) Technical Definition,* 2013.

[20] GS1 International, *GS1 EANCOM Overview,* 2013.

10. ANEXOS

Como se estableció inicialmente en el documento de anteproyecto, los resultados de este trabajo serán almacenados exclusivamente en los repositorios indicados por Carvajal Tecnología y Servicios, puesto que estos contienen información confidencial y son de uso exclusivo de la empresa.

Los artefactos involucrados en el desarrollo de este proyecto fueron:

1. **FORA701K – Acta de Constitución del Proyecto**

Nombre de archivos:

- [GL-UENEBZ-EBZSUI-Consumo-2013-01] FORA701K - Acta Constitución Proyecto.doc
- [GL-UENEBZ-EBZSUI-Consumo-2013-01] Anexo 1 - Acta de Constitución.doc

2. **FORA707D – Documento de Especificación de Requisitos**

Nombre de archivos:

- [GL-UENEBZ-EBZSUI-Consumo-2013-01] FORA707D.doc

3. **FORA725D – Documento de Arquitectura**

Nombre de archivos:

- FORA725D-EBZ_Documento_de_Arquitectura_de_Software-v1.1.doc

4. **Glosario (De carácter público)**

Nombre de archivos:

- ANEXO 4 - GLOSARIO DOCUMENTO FINAL TRABAJO DE GRADO.pdf