

REPOSITORIO ACADÉMICO UPC

Formalización del proceso de elaboración de una arquitectura de software

Item Type	info:eu-repo/semantics/bachelorThesis
Authors	Gonzales Yapapasca, César Armando; Torres Cárdenas, José Emilio
Citation	[1] C. A. Gonzales Yapapasca and J. E. Torres Cárdenas, "Formalización del proceso de elaboración de una arquitectura de software," Universidad Peruana de Ciencias Aplicadas (UPC), Lima, Perú, 2018.
Publisher	Universidad Peruana de Ciencias Aplicadas (UPC)
Rights	info:eu-repo/semantics/openAccess
Download date	04/09/2024 13:16:31
Item License	http://creativecommons.org/licenses/by-nc-sa/3.0/us/
Link to Item	http://hdl.handle.net/10757/624131



UNIVERSIDAD PERUANA DE CIENCIAS APLICADAS

Facultad de Ingeniería

Carrera de Ingeniería de Software

**FORMALIZACIÓN DEL PROCESO DE
ELABORACIÓN DE UNA ARQUITECTURA DE
SOFTWARE**

TESIS

Para optar por el título de INGENIERO DE SOFTWARE

AUTORES

Gonzales Yapapasca, César Armando (0000-0002-0999-0894)

Torres Cárdenas, José Emilio (0000-0002-2322-4475)

ASESOR DE TESIS

García Paucar, Luis Hernán

Lima, Julio de 2018

Esta trabajo de investigación está dedicado a nuestras familias, cuyo apoyo se mantuvo incondicional durante nuestra vida universitaria y crecimiento profesional. Gracias por todo.

Resumen Ejecutivo

El presente proyecto tiene como propósito formalizar el proceso de elaboración de una arquitectura de software y presentar los métodos empleados en cada parte del proceso, así como guías, plantillas y ejemplos. Para ello, se estudió la información disponible sobre arquitectura de software del Software Engineering Institute (SEI) de la Carnegie Mellon University (CMU) por ser una entidad dedicada a la investigación de dicho tema.

Producto del desarrollo del proyecto, se formalizó el proceso de elaboración de una arquitectura de software y se generaron guías, plantillas y ejemplos para los métodos inmersos en dicho proceso, siendo estos:

- El taller de atributos de calidad (QAW)
- El diseño basado en atributos (ADD)

Finalmente, en conclusiones se presenta los trabajos futuros necesarios para complementar el proceso.

Palabras clave : Arquitectura de Software, Documentación de una Arquitectura de Software, Taller de atributos de calidad, QAW, Diseño basado en atributos, ADD.

Abstract

This project has as a purpose the definition and characterization of the required process for elaborating a software architecture. Furthermore the project presents some of the methods involved in the stages of this process. For each one it was elaborated guidelines, templates and examples on how they should be applied. The project based its research in the knowledge generated by the Software Engineering Institute (SEI) from the Carnegie Mellon University (CMU) because of its known experience and researches about the topic.

The project generated a detailed process of elaborating a software architecture and also generated guidelines, templates and examples of two methods involved in this process which are:

- The Quality Attribute Workshop (QAW)
- The Attribute Driven Design (ADD)

As part of the conclusions obtained it is defined the required work in the topic to complete and evolve this process.

Keywords : Software Architecture, Documenting Software Architectures, Quality Attribute Workshop, QAW, Attribute Driven Design, ADD.

Tabla de Contenido

INTRODUCCION	09
CAPITULO 1. DEFINICION DEL PROYECTO	10
Objeto de Estudio.....	10
Dominio del problema	10
Planteamiento de la solución	11
Objetivos del proyecto	11
Objetivo general	11
Objetivos específicos	11
Indicadores de éxito	12
Planificación del proyecto.....	13
Alcance	13
Plan de Gestión del Tiempo.....	14
Plan de Gestión de Recursos Humanos	15
Plan de Comunicaciones	16
Plan de Gestión de Riesgos.....	17
CAPITULO 2: MARCO TEORICO.....	18
Arquitectura de software.....	18
Software Engineering Institute (SEI) en el campo de la arquitectura de software	19
CAPITULO 3. ESTADO DEL ARTE	20
Revisión de la literatura	20
Enfoque del software Engineering Institute (SEI)	21
Enfoque de IBM.....	23
CAPITULO 4: DEFINICION DE LOS ATRIBUTOS DE CALIDAD	27
Taller de atributos de calidad (QAW).....	27
CAPITULO 5: DISEÑO BASADO EN ATRIBUTOS (ADD)	31
Diseño basado en atributos (ADD)	31
CAPITULO 6: IMPLEMENTACION.....	34
Presentación del caso	34
Funcionalidades	37
Restricciones	38
QAW	39

Atributos de Calidad	39
Diseño basado en atributos (ADD)	40
CAPITULO 7: GESTION DEL PROYECTO.....	42
Producto final.....	42
Gestión del tiempo	43
Gestión de los recursos humanos.....	44
Gestión de las comunicaciones	45
Gestión de los riesgos	46
Lecciones aprendidas	46
CONCLUSIONES	47
RECOMENDACIONES.....	49
GLOSARIO	50
ANEXOS	51
REFERENCIAS BIBLIOGRÁFICAS.....	203

Índice de Figuras

Figura 1. Cronograma del proyecto	15
Figura 2. Diagrama de Contexto de uReader.....	35
Figura 3. Pantalla de Agregar nueva Fuente de Información	35
Figura 4. Pantalla de contenido de un elemento (noticia, evento, etc.) de una fuente de información	36
Figura 5. Pantalla de listado de fuentes de información	36
Figura 6. Pantalla de ingreso de comando de voz.....	37
Figura 7. Cronograma del proyecto	44

ÍNDICE DE TABLAS

Tabla 1. Tabla de declaración del problema y sus causas	10
Tabla 2. Tabla de organización del proyecto	16
Tabla 3. Tabla de comunicación	16
Tabla 4. Tabla de riesgos	17
Tabla 5. Listado de los comandos de voz y acción esperada.....	38
Tabla 6. Tabla de organización del proyecto	41
Tabla 7. Tabla de organización del proyecto.....	45
Tabla 8. Tabla de riesgos	46

Introducción

Con el propósito de una mejor formación de los estudiantes en Ingeniería de Software, la escuela de Ingeniería de la Universidad Peruana de Ciencias Aplicadas ideó un conjunto de empresas virtuales donde los alumnos de los últimos ciclos pueden vivir la experiencia del trabajo en empresas reales, allí son evaluados exigentemente para asegurar su correcta formación. Sin embargo, aún existen deficiencias específicamente en lo que es la documentación del diseño de arquitecturas de software tanto en los alumnos de la carrera como en las empresas virtuales.

Este proyecto se enfoca en resolver este problema y así aportar a las empresas virtuales y a los estudiantes de la carrera con los conocimientos necesarios que le permitan realizar una documentación correcta de una arquitectura de software. Y así, contribuir no sólo con la misión de la carrera de Ingeniería de Software sino también con la misión de la Universidad Peruana de Ciencias Aplicadas.

De esta manera, se llevará a cabo una investigación con una duración total de dos ciclos académicos, de la cual se obtendrá la documentación y conocimiento necesario de los procesos que se deben seguir para poder realizar la documentación de una arquitectura de software según las buenas prácticas del Software Engineering Institute (SEI). Además, se busca obtener los artefactos necesarios que participen dentro de dichos procesos. Finalmente, se realizará un piloto de lo generado sobre un proyecto provisto por el cliente y de esta forma, se verificará la calidad de la documentación alcanzada.

Capítulo 1. Definición del Proyecto

Este capítulo presenta la problemática principal bajo la cual nace el proyecto, así como también la solución planteada. Además, muestra los objetivos planteados por el equipo y los identificadores de éxito definidos para medir el logro. Finalmente, se presenta la planificación realizada para la ejecución del proyecto.

Objeto de Estudio

Procesos de Documentación de Arquitectura de Software según el Instituto de Ingeniería de Software (SEI - por sus siglas en inglés) de la Universidad Carnegie Mellon (CMU) para su aplicación en las empresas virtuales Software Factory y Quality Assurance de la escuela de Ingeniería de Sistemas y Computación de la Universidad Peruana de Ciencias Aplicadas.

Dominio del problema

Tabla 1. Tabla de declaración del problema y sus causas

Problema	Causa
Durante el desarrollo de proyectos en las empresas virtuales, los miembros del proyecto no tienen en claro el análisis, elaboración e implementación requeridos para la arquitectura de software del proyecto.	Los proyectos presentados en las empresas virtuales de la Universidad Peruana de Ciencias Aplicadas no poseen una documentación de arquitectura de software correcta.
Los alumnos egresados de la carrera de Ingeniería de Software no alcanzan el conocimiento necesario en arquitecturas de software, por lo cual no se cumple del todo con las metas propuestas por la carrera.	Los proyectos presentados en las empresas virtuales de la Universidad Peruana de Ciencias Aplicadas no siguen una evaluación correcta de arquitectura de software.

Elaboración propia

Planteamiento de la solución

Formalizar el proceso para la documentación de una arquitectura de software y generar los artefactos y guías necesarios para su aplicación.

Objetivos del proyecto

Objetivo general

Definir el proceso para la elaboración de una Arquitectura de Software, así como generar los artefactos que dicho proceso implique.

Objetivos específicos

A continuación, se listan los objetivos específicos del proyecto:

- OE1: Analizar los requerimientos para la captura de escenarios de atributos de calidad (QAW).
 - Generar la guía en español de cómo realizar la captura de atributos de calidad (QAW).
 - Generar la plantilla de apoyo para el taller de captura de atributos de calidad (QAW).
 - Generar la plantilla y ejemplos de apoyo para la captura de escenarios.
 - Generar la plantilla y ejemplos de apoyo para el refinamiento de escenarios.
- OE2: Diseñar la arquitectura de software basado en atributos (ADD).
 - Generar la guía en español de cómo realizar un diseño de arquitectura de software basado en atributos (ADD).
 - Generar la guía en español de cómo integrar los métodos de captura de atributos de calidad (QAW) y el diseño de una arquitectura de software basado en atributos (ADD).
 - Generar la plantilla de la matriz de patrones y conductores arquitecturales.
 - Generar el checklist del diseño de una arquitectura basado en atributos (ADD).
- OE3: Construir los artefactos producidos durante la documentación de una arquitectura de software.
 - Generar la plantilla completa de un documento de arquitectura de software (SAD).
 - Generar la plantilla light de un documento de arquitectura de software (SAD).

- Generar la plantilla del anexo para el análisis del enfoque arquitectural.
- Generar la plantilla del anexo para la documentación de decisiones arquitecturales.
- Generar el ejemplo 1 en español de un documento de arquitectura de software (SAD).
- Generar el ejemplo 2 en español de un documento de arquitectura de software (SAD).
- OE4: Presentar resultados de la investigación a través de formatos actualizados.
 - Formalizar y caracterizar el proceso de la captura de atributos de calidad (QAW).
 - Formalizar y caracterizar el proceso de diseño de una arquitectura de software basado en atributos (ADD).

Indicadores de éxito

A continuación, se listan los indicadores de éxito del proyecto:

- IE1: Conformidad con los documentos del proceso de captura de atributos de calidad (QAW) generados:
 - Guía en español de cómo realizar la captura de atributos de calidad (QAW).
 - Proceso definido y caracterizado de la captura de atributos de calidad (QAW).
 - Plantilla de apoyo para el taller de captura de atributos de calidad (QAW).
 - Plantilla de apoyo y ejemplos para la captura de escenarios.
 - Plantilla de apoyo y ejemplos para el refinamiento de escenarios.
- IE2: Conformidad con los documentos del proceso de diseño de una arquitectura de software basado en atributos (ADD) generados:
 - Guía en español de cómo realizar un diseño de arquitectura de software basado en atributos (ADD).
 - Proceso definido y caracterizado del diseño de una arquitectura de software basado en atributos (ADD).
 - Guía en español de cómo integrar los métodos de captura de atributos de calidad (QAW) y el diseño de una arquitectura de software basado en atributos (ADD).
 - Plantilla de la matriz de patrones y conductores arquitecturales.
 - Checklist del diseño de una arquitectura basado en atributos (ADD).

- IE3: Conformidad con los artefactos relacionados al proceso de elaboración de una arquitectura de software generados:
 - Plantilla completa de un documento de arquitectura de software (SAD).
 - Plantilla light de un documento de arquitectura de software (SAD).
 - Plantilla del anexo para el análisis del enfoque arquitectural.
 - Plantilla del anexo para la documentación de decisiones arquitecturales.
 - Ejemplo 1 en español de un documento de arquitectura de software (SAD).
 - Ejemplo 2 en español de un documento de arquitectura de software (SAD).
 - Ejemplo 2 en inglés de un documento de arquitectura de software (SAD).
- IE4: Conformidad del cliente con el material elaborado para la capacitación y la capacitación de las empresas virtuales quality assurance y software factory.
 - Acta de conformidad con el material elaborado.
 - Acta de conformidad con la capacitación realizada en la empresa virtual quality assurance.
 - Acta de conformidad con la capacitación realizada en la empresa virtual software factory.

Planificación del proyecto

A fin de controlar satisfactoriamente el desarrollo del proyecto, se realizó la planificación detallada a continuación.

Alcance

A continuación, se detalla el trabajo que se incluirá en el proyecto, y el trabajo que no está incluido, ambos basados en los requisitos recogidos con el cliente y aprobados por el mismo.

Inclusiones

- Primer semestre
 - Definición del proceso de la documentación de una arquitectura de software.
 - Elaboración de artefactos de los métodos involucrados en el proceso de documentación de la arquitectura de software.

- Definición de los siguientes atributos de calidad
 - Rendimiento (Performance)
 - Capacidad de Modificación (Modifiability)
 - Disponibilidad (Availability)
- Segundo semestre
 - Realización de ejemplos de documentación de arquitectura de software.
 - Realización de un proyecto piloto siguiendo los procesos obtenidos y empleando los artefactos presentados.
 - Capacitación sobre los procesos obtenidos y usos correctos para los artefactos generados.

Exclusiones

- La realización de un producto software que automatice los procesos de documentación de una arquitectura de software.
- Adopción de los procesos generados en las empresas virtuales de la Escuela de Sistemas y Computación de la UPC.

Plan de Gestión del Tiempo

A continuación, se muestra un cronograma actualizado del proyecto mostrando las actividades e hitos, con las fechas consideradas.

- Formalización de procesos de documentación y evaluación de Arquitectura de Software	1.422 hrs	169 días?	mar 09/04/13	jue 28/11/13		
CG - Publicación de rol de exposiciones de charters y cronograma TP1	0 hrs	0 días	mar 09/04/13	mar 09/04/13		
+ Planificación	48 hrs	3 días	mar 09/04/13	jue 11/04/13		
- Investigación	322 hrs	24 días	jue 11/04/13	mar 14/05/13	3	
- Documentación de una Arquitectura de Software	322 hrs	24 días	jue 11/04/13	mar 14/05/13		
CG - Exposición de charters y cronogramas TP1	0 hrs	0 días	jue 11/04/13	jue 11/04/13		
Recopilación de información	2 hrs	4 días	vie 12/04/13	mié 17/04/13	3	Cesar Gonzales[3%];Jose Torres[3%]
Estudio de la información	80 hrs	5 días	jue 18/04/13	mié 24/04/13	9	Cesar Gonzales;Jose Torres
- Proceso de documentación	240 hrs	15 días	jue 25/04/13	mar 14/05/13	10	
Definición del proceso	48 hrs	3 días	jue 25/04/13	lun 29/04/13		Cesar Gonzales;Jose Torres
Representación del proceso	32 hrs	2 días	mar 30/04/13	mié 01/05/13	12	Cesar Gonzales;Jose Torres
+ Revisión QA	96 hrs	6 días	mié 01/05/13	mié 08/05/13	13	
Generación de artefactos involucrados en el proceso	64 hrs	4 días	jue 09/05/13	mar 14/05/13	14	Cesar Gonzales;Jose Torres
Inicio de actividades ciclo académico 2013-2	0 hrs	0 días	lun 12/08/13	lun 12/08/13		
+ Modificación de los Deployment Packages de la ISO/IEC29110	176 hrs	14 días	jue 05/09/13	mar 24/09/13		
+ Realización de piloto	404 hrs	78,5 días	lun 12/08/13	jue 28/11/13	26	
- Proceso de documentación de la arquitectura	404 hrs	78,5 días	lun 12/08/13	jue 28/11/13		
Primera Entrega de ejemplo SAD	208 hrs	26 días	lun 12/08/13	lun 16/09/13		Cesar Gonzales[50%];Jose Torres[50%]
Realización de Taller de Atributos de Calidad	32 hrs	4 días	vie 13/09/13	mié 18/09/13		Cesar Gonzales[50%];Jose Torres[50%]
Selección del proyecto piloto	24 hrs	3 días	mar 24/09/13	jue 26/09/13		Cesar Gonzales[50%];Jose Torres[50%]
Revisión del proyecto	24 hrs	3 días	vie 27/09/13	mar 01/10/13	39	Cesar Gonzales[50%];Jose Torres[50%]
Entrega de guías y documentación	0 hrs	0 días	vie 08/11/13	vie 08/11/13		
Instrucción de uso de las guías	32 hrs	4 días	vie 08/11/13	mié 13/11/13	41	Cesar Gonzales[50%];Jose Torres[50%]
Supervisión y control durante el uso de las guías	60 hrs	7,5 días	jue 14/11/13	lun 25/11/13	42	Cesar Gonzales[50%];Jose Torres[50%]
Reporte de resultados del piloto	24 hrs	3 días	lun 25/11/13	jue 28/11/13	43	Cesar Gonzales[50%];Jose Torres[50%]
+ Presentación de Proyecto	128 hrs	12 días	mar 01/10/13	mié 16/10/13		
Elaboración de Memoria - 3ra Entrega	64 hrs	8 días	mar 01/10/13	jue 10/10/13		Cesar Gonzales[50%];Jose Torres[50%]
Revisión BankMin: Memoria - 3ra Entrega	0 hrs	0 días	jue 10/10/13	jue 10/10/13	46	
Entrega Memoria - 3ra Entrega	0 hrs	0 días	jue 10/10/13	jue 10/10/13	47	
Elaboración de Presentación	64 hrs	4 días	vie 11/10/13	mié 16/10/13	46	Cesar Gonzales;Jose Torres
Exposición del Proyecto	0 hrs	0 días	mié 16/10/13	mié 16/10/13	49	
+ Cierre de Proyecto	344 hrs	31 días	jue 17/10/13	jue 28/11/13	45	
Elaboración Perfil de Proyecto	192 hrs	12 días	jue 17/10/13	vie 01/11/13		Cesar Gonzales;Jose Torres
Entrega de Perfil de Proyecto	0 hrs	0 días	vie 01/11/13	vie 01/11/13	52	
Elaboración Reporte de Cierre de Proyecto	56 hrs	7 días	lun 04/11/13	mar 12/11/13	53	Cesar Gonzales[50%];Jose Torres[50%]
Elaboración de Memoria - 4ta Entrega	72 hrs	9 días	mié 13/11/13	lun 25/11/13	54	Cesar Gonzales[50%];Jose Torres[50%]
Revisión BankMin: Memoria - 4ta Entrega	0 hrs	0 días	lun 25/11/13	lun 25/11/13	55	
Entrega Memoria - 4ta Entrega	0 hrs	0 días	lun 25/11/13	lun 25/11/13	56	
Elaboración de Presentación	24 hrs	3 días	mar 26/11/13	jue 28/11/13	55	Cesar Gonzales[50%];Jose Torres[50%]
Exposición del Proyecto	0 hrs	0 días	jue 28/11/13	jue 28/11/13	58	
+ Reuniones programadas del proyecto	0 hrs	133 días?	mar 16/04/13	mié 16/10/13		

Figura 1. Cronograma del proyecto

Elaboración propia

Plan de Gestión de Recursos Humanos

A continuación, se identifican los roles dentro del proyecto, las personas asignadas a dichos roles, así como las responsabilidades que cumplen dentro del proyecto.

Tabla 2. Tabla de organización del proyecto

Responsable	Roles	Responsabilidades
César Gonzales Yapapasca	Jefe de Proyecto	Llevar a cabo la investigación y gestionar las actividades de los recursos involucrados.
José Torres Cárdenas	Jefe de Proyecto	Llevar a cabo la investigación y gestionar las actividades de los recursos involucrados.
Comité de evaluación de Proyectos de la UPC	Comité de proyectos	Evaluación del proyecto enfocándose en la presentación de la documentación.
Asesor	Prof. Luis García Paucar	Brindar asesoría durante el proyecto de investigación.
Cliente	Prof. Luis García Paucar	Brindar las necesidades y alcance del proyecto de investigación.

Elaboración propia

Plan de Comunicaciones

A continuación, se detalla la necesidad de información entre los integrantes e interesados del proyecto, así como los canales y flujos de comunicación.

Tabla 3. Tabla de comunicación

Tipo de comunicación	Descripción	Frecuencia	Formato	Participantes	Entregables	Propietario
Reportes de monitoreo	Reporte del estado del proyecto según las actividades planificadas	Semanal	Documento virtual	Jefes del Proyecto, Empresa virtual Bankmin	Reporte de monitoreo	Jefe de Proyecto
Informes de seguimiento	Informes de seguimiento del asesor y cliente al proyecto.	Semanal	Documento virtual	Jefes del Proyecto, Empresa virtual Bankmin, Asesor	Informe de seguimiento	Jefe de Proyecto
Actas de reunión	Actas de reunión entre los jefes del proyecto y el cliente	Semanal/Quincenal	Documento Virtual	Jefes del Proyecto, Representante del cliente	Actas de reunión	Jefe de Proyecto

Elaboración propia

Plan de Gestión de Riesgos

A continuación, se enlistan los posibles riesgos que pueden hacer peligrar la culminación del proyecto. Cada riesgo va acompañado de una estimación de probabilidad de que ocurra, el impacto y las estrategias de mitigación tomada por el equipo.

Tabla 4. Tabla de riesgos

#	Riesgo	Probabilidad	Impacto	Estrategia de mitigación
1	Incumplimiento con las reuniones con el experto.	Media	Alto	Reprogramación de la reunión.
2	Dificultades para la elección del proyecto piloto.	Media	Alto	Uso de un proyecto externo a las empresas virtuales.
3	Falta de apoyo con un recurso asignado al proyecto.	Bajo	Alto	Distribución del trabajo del recurso entre los jefes de proyecto.

Elaboración propia

Capítulo 2: Marco Teórico

Este capítulo define los conceptos básicos que permiten comprender el proyecto. Además, se brinda una breve introducción a la temática de la arquitectura de software dentro del campo de la ingeniería de software y el porqué de su importancia. Finalmente se describe brevemente cuál es el proceso de elaboración de una arquitectura de software desde el análisis, diseño, evaluación y documentación; también se explica cómo realmente la documentación resulta transversal a las demás etapas.

Arquitectura de software

Existen muchas definiciones de arquitectura de software, las cuales pueden ser fácilmente encontradas con tal solo tipearlo en la web. Algunas de ellas la definen como las decisiones más tempranas y/o de mayor importancia dentro de un sistema. Sin embargo, si bien es cierto que son decisiones importantes, éstas no necesariamente son tempranas en todos los casos. Por ejemplo, en desarrollo de proyectos ágiles o espirales. La definición que establece el Software Engineering Institute (SEI) es:

“La arquitectura de software de un sistema es el conjunto de estructuras necesarias para razonar acerca del sistema, el cual comprende elementos de software, relaciones entre estos y propiedades de los mismos”.

Una estructura es un conjunto de elementos cohesionados por sus relaciones. Los sistemas de software son compuestos de muchas estructuras que le dan soporte al sistema en sí. Existen tres tipos de estructuras que juegan un rol importante en el diseño, documentación y análisis de arquitecturas. La primera categoría de estructura parte el sistema en unidades de implementación, los cuales son llamados módulos. La segunda categoría son dinámicas, es decir, se enfocan en la forma en que los elementos interactúan entre sí en tiempo de ejecución para llevar a cabo las funcionalidades del sistema. Finalmente, la tercera categoría describe el mapa entre las estructuras de software y los ambientes organizacionales, de desarrollo, de instalación y ejecución del sistema. Cabe resaltar que, una estructura es arquitectural si da soporte al razonamiento del sistema y las propiedades del sistema.

Por otro lado, debido a que la arquitectura consiste de estructuras y las estructuras consisten de elementos y sus relaciones, entonces quiere decir que una arquitectura comprende elementos de software y cómo sus elementos interactúan entre sí. Por lo tanto, una arquitectura omite cierta información acerca de los elementos que no es relevante para el razonamiento del sistema, particularmente, omite información que no tiene ramificaciones fuera de un solo elemento. Por ende, una arquitectura es una abstracción del sistema que selecciona ciertos detalles y suprime otros.

Además, en el caso más trivial de un sistema, éste es en sí mismo una arquitectura de un solo ejemplo. Tal vez, no sea una arquitectura importante o útil, pero sigue siendo una arquitectura del sistema.

Finalmente, todo sistema tiene una arquitectura independientemente de su descripción o especificación, ya que ésta es inherente al sistema.

Software Engineering Institute (SEI) en el campo de la arquitectura de software

Este enfoque de definición de una arquitectura de software está dado por el Instituto de Ingeniería de Software (SEI por sus siglas en inglés) de la Universidad Carnegie Mellon de Estados Unidos. Este instituto es un centro de investigación y desarrollo respaldado por el departamento de Defensa de los Estados Unidos de América, el cual ayuda en los principios y prácticas de ingeniería de software que contemplan los siguientes tópicos: ingeniería de software, seguridad de computadoras y mejora de procesos. El SEI trabaja muy de cerca con las organizaciones de defensa y gobierno de los Estados Unidos, la industria y las organizaciones académicas para mejorar continuamente sistemas de software. En este sentido, el SEI posee un capítulo destinado a la investigación y planteamiento de lineamientos de la Arquitectura de Software. Y, es quien en la actualidad propone un proceso bien documentado sobre las mejores prácticas en términos de software, y específicamente las mejores prácticas en la elaboración y documentación de una arquitectura de software.

Capítulo 3. Estado del arte

Este capítulo presenta el estado del arte de dos diferentes enfoques de elaboración de una arquitectura de Software. El primer enfoque es el propuesto por IBM con su Rational Unified Process (RUP) y el segundo es la propuesta de elaborada por el Software Engineering Institute. Ambos enfoques son expuestos de acuerdo a sus lineamientos.

Revisión de la literatura

La presente investigación expone dos de los procesos más conocidos para el diseño de una arquitectura de software. Para luego, generar la formalización del mejor proceso para poder ser empleado en la empresa cliente. A continuación, se describen brevemente ambos métodos.

El Diseño Basado en Atributos, que fue desarrollado en el Software Engineering Institute (SEI), en un enfoque de elaborar el diseño de una arquitectura de software basando el proceso de diseño en los requerimientos de atributos de calidad.

En el proceso del Diseño Basado en Atributos se basa en una descomposición recursiva, donde en cada iteración se toma un elemento del sistema a descomponer y se escogen las tácticas arquitecturales y los patrones de diseño necesarios para satisfacer el conjunto de atributos de calidad relacionados. Lo generado por el Diseño Basado en Atributos representa un conjunto de decisiones arquitecturales de alto nivel documentados. Luego, las vistas a emplear dependen de la naturaleza del proyecto, contando al menos con una de cada categoría: vistas modulares, vistas componente-conector y vistas de asignación [Bachmann 2006].

Por otro lado, uno de los enfoques más conocidos es el presentado por el Rational Unified Process (RUP); el cual es un proceso de desarrollo creado por IBM. Para el Rational Unified Process (RUP) la arquitectura de software de un sistema comprende esencialmente las decisiones arquitecturales sobre la organización del sistema como por ejemplo [Rational 1998]:

- La elección de los elementos y sus interfaces por las cuales un sistema es compuesto.
- El comportamiento especificado entre los elementos de colaboración.

El Rational Unified Process (RUP) define un método de diseño arquitectural usando el concepto desarrollado por Kruchten en 1995, RUP 4+1. El cual consiste en emplear cuatro vistas para describir el diseño: una vista lógica, una vista de procesos, una vista de implementación y una vista de despliegue; y adicionalmente una vista de casos de uso o de escenarios.

En el proceso del Rational Unified Process el diseño de la arquitectura se realiza a lo largo de iteraciones, comenzando en la fase de elaboración, donde se van populando las 4 vistas principales. Este diseño arquitectural es conducido por los casos de usos arquitecturalmente significativos, los requerimientos no funcionales y los riesgos.

Enfoque del software Engineering Institute (SEI)

Producto de más de dos décadas en investigaciones, el Software Engineering Institute (SEI) ha generado herramientas, métodos y cursos sobre el campo de la arquitectura de software. Además, el SEI ha trabajado de cerca con docenas de organizaciones en la industria del software, el gobierno y organizaciones académicas, lo que han permitido madurar y evolucionar las prácticas de ingeniería relacionadas a la arquitectura del software. De esta manera, el SEI ha publicado cinco libros, cientos de artículos científicos, papers, reportes técnicos y ha realizado numerosas presentaciones y videoconferencias a través del internet.

Para que un sistema pueda alcanzar los atributos de calidad requeridos por el cliente se debe contar con una buena arquitectura de software que lo soporte. Dicha arquitectura, desde su diseño debe haber considerado lo necesario para poder satisfacer los requerimientos no funcionales. Para poder lograr ello es importante realizar una correcta definición de los atributos de calidad deseados para el sistema y hacerlo antes de iniciar con el diseño de la arquitectura y con la participación de todos los interesados (stakeholders) involucrados. Es así que, el Software Engineering Institute (SEI) propone el método Quality Attribute Workshop (QAW) cuyo objetivo principal es este, establecer los atributos de calidad del sistema con la participación de todos los involucrados.

Para poder lograr con sus objetivos, el método del Quality Attribute Workshop (QAW) debe contar con la presencia de todos los involucrados en la elaboración del sistema o en su defecto la mayor cantidad posible. Además, se cuenta con un moderador o facilitador quien está a cargo de dirigir el taller y un encargado de la redacción quien tomará las notas de las decisiones importantes. Durante el taller de Quality Attribute Workshop (QAW) se realiza la presentación

de la misión y negocio de la empresa cliente; esto es realizado por los representantes del negocio que estén participando del taller. También se dan a conocer los conductores del negocio (Business and/or mission drivers) para que quede expreso y claro a todo el equipo. También se realiza una presentación de lo que se tenga documentado sobre el sistema a construir; esto también es realizado por representantes del negocio del cliente. En este punto se suelen presentar papeles como estrategias, planes, requerimientos técnicos clave o restricciones del sistema. Luego se identifican los conductores arquitecturales de todo lo escuchado durante el taller. Estos conductores suelen incluir requerimientos de alto nivel, preocupaciones en cuanto a la misión y negocio, objetivos, metas y atributos de calidad. También se realiza una lluvia de ideas de escenarios basados en la lista de los conductores arquitecturales obtenidos. Uno de las metas del taller es generar por lo menos un escenario para cada conductor arquitectural listado por los interesados. Una vez con la lista y el consenso se priorizan los escenarios que se crearon. Mientras el tiempo lo permita, se escogen los cuatro o cinco de los primeros escenarios y se procede a detallarlos.

Por otra parte, para poder realizar el diseño de una arquitectura de software para un sistema, el Software Engineering Institute (SEI) propone un método conocido como el diseño basado en atributos (Attribute Driven Design - ADD). El método comprende un proceso recursivo que descompone el sistema o elemento del sistema aplicando las tácticas y patrones arquitecturales que buscan satisfacer los requerimientos de atributos de calidad. De esta manera y a través de un proceso iterativo se pretende alcanzar con un diseño sólido de la arquitectura de software del sistema.

Para poder iniciar con el método se requiere contar con los requerimientos funcionales, los cuales nos indican qué funciones busca proveer el sistema; restricciones de diseño, los cuales son decisiones de diseño que deben ser parte del diseño final; y requerimientos de atributos de calidad, los cuales nos indican los grados a los cuales un sistema deberá exponer varias propiedades, todos estos definidos y priorizados por los interesados del negocio.

Como resultado de aplicar el método se espera obtener un diseño de sistema que comprende los roles, responsabilidades, propiedades y relaciones entre los elementos de software que lo conforman. Además, contar con el diseño documentado usando varios tipos de vistas arquitecturales (Vistas Modulares, Vistas Componente-Conector, Vistas de Asignación).

Durante el método se verificar que exista la información necesaria de los requerimientos funcionales, restricciones de diseño y requerimientos de atributos de calidad, todos estos revisados, priorizados y aceptados por los interesados del sistema. Luego, se escoge el elemento que se va a descomponer. En caso de estar pasando por primera vez, el único elemento disponible para descomponer es el sistema en sí. Luego se identifican los candidatos a conductores arquitecturales, para lo cual se deben priorizar por segunda vez los requerimientos que afecten al elemento seleccionado. De la lista priorizada, se escogen entre cinco a seis requerimientos con alta prioridad los cuales se convierten en potenciales conductores arquitecturales. Finalmente, y como objetivo principal del método, se eligen los tipos de elementos, relaciones y sus interacciones. Se debe tomar cada preocupación de diseño relacionada a cada candidato a conductor arquitectural y sugerir patrones de diseño que ayuden a solventar el problema o preocupación. Finalmente se definen los servicios y propiedades requeridas por los elementos para satisfacer nuestro diseño. Estos detalles luego deben ser registrados en el documento de interfaz del elemento respectivo. Es importante verificar que la descomposición del elemento realizada cumpla con los requerimientos funcionales, requerimientos de atributos de calidad y las restricciones de diseño y luego traducir las responsabilidades que fueron asignadas a los elementos hijos en requerimientos funcionales para los elementos individuales.

Enfoque de IBM

El Rational Unified Process (RUP) es un Proceso de Ingeniería de Software. Este provee un enfoque disciplinado para asignar tareas y responsabilidades dentro de la organización del desarrollo. Su meta es asegurar la producción de software de alta calidad que satisfaga las necesidades de sus usuarios finales, dentro un cronograma y presupuesto previsibles [Rational 1998].

Además, el Rational Unified Process es una guía para cómo usar efectivamente usar el Lenguaje de Modelamiento Unificado (UML por sus siglas en inglés). El UML es un lenguaje estandarizado en la industria que nos permite comunicar claramente requerimientos, arquitecturas y diseños [Rational 1998].

RUP posee 4 fases dentro del ciclo de vida del software: Incepción, Elaboración, Construcción y Transición. Cada fase es concluida con un bien definido hito - un punto en el tiempo en el

cual ciertas decisiones críticas deben ser hechas y además los objetivos principales deben ser alcanzados.

Para efectos de este análisis solo es necesario enfocarnos en la fase Elaboración, ya que es en ella en donde encontraremos la definición y elaboración de la arquitectura.

El propósito de la fase de Elaboración es analizar el dominio del problema, establecer una buena base arquitectónica, desarrollar el plan de trabajo y eliminar los elementos de alto riesgo del proyecto. Para alcanzar estos objetivos, se debe tener una visión total del sistema a desarrollar. Las decisiones arquitecturales tienen que ser hechas dentro de un entendimiento general del sistema: su alcance, las funcionalidades principales y los requerimientos no funcionales como requerimientos de performance.

Es fácil argumentar que la fase de elaboración es la más crítica de las 4 fases. Al final de esta fase, la ingeniería más dura es considerada completa y el proyecto experimenta su día más importante de cálculo: la decisión de avanzar o no hacia las fases de construcción y transición. Para muchos proyectos, esto también corresponde a la transición de un móvil, ligero y ágil operación de bajo riesgo hacia una operación de alto costo y alto riesgo con una inercia sustancial. Mientras el proceso siempre tiene que acomodarse a los cambios, las actividades de la fase de elaboración aseguran que la arquitectura, requerimientos y planes son suficientemente estables y los riesgos son suficientemente mitigados, por ende, se puede determinar predeciblemente el costo y el cronograma para la finalización del desarrollo. Conceptualmente, este nivel de fidelidad correspondería a un nivel necesario para una organización para realizar una fase de construcción con un costo fijo.

Para iniciar el proceso de elaboración de arquitectura de software es necesario identificar los casos de uso de negocio principales y entender los requerimientos no funcionales, restricciones y conductores del negocio.

Para documentación y elaboración de una arquitectura de software dentro del proceso de RUP se establecen los siguientes pasos para la definición de una arquitectura candidata:

Desarrollar una vista general del sistema El propósito es facilitar la visión del sistema, explorando y evaluando opciones arquitecturales de alto nivel. Además, tiene el propósito de comunicar al sponsor, equipo de desarrollo, y otros stakeholders un entendimiento temprano de las estructuras de alto nivel del sistema.

Encuesta de activos disponibles El propósito es identificar los activos que pueden ser relevantes para este proyecto, decidir si basar las áreas del sistema en activos y localizar y listar los activos que son potencialmente reusables en el proyecto.

Definir en alto nivel la organización de los subsistemas En este se crea una estructura inicial del Modelo de Diseño. El modelo de diseño es normalmente organizado en capas. El número de capas no es fijo, pero varía de una situación a otra. Durante el análisis arquitectural se enfoca normalmente en 2 capas de alto nivel, es decir, las capas de aplicación y de negocio.

Identificar abstracciones principales Se identifican las abstracciones principales (representación de conceptos identificados durante el modelamiento de negocio y las actividades de requerimiento) que el sistema debe manejar.

Desarrollar un modelo de despliegue a alto nivel Se provee la base para evaluar la viabilidad de implementar el sistema y ganar un entendimiento de la distribución geográfica y complejidad operacional del sistema.

El modelo de despliegue debe ser capaz de dar soporte a los usuarios en ubicaciones ejecutando los casos de uso que acceden a componentes que acceden a datos, mientras se satisfacen los requerimientos no funcionales y restricciones.

Crear realizaciones de casos de uso El propósito es crear los artefactos del Modelo de Diseño usados para expresar el comportamiento de los casos de uso.

Identificar interacciones estereotipadas El propósito de este paso es identificar estas interacciones entre las abstracciones principales en el sistema, los cuales caracterizan tipos significativos de actividad en el sistema. Estas interacciones son capturadas como realizaciones de caso de uso.

Revisar los resultados Se deben revisar los mecanismos arquitecturales, los subsistemas, paquetes y clases que han sido identificados para asegurar que estén completos y consistentes. Los escenarios de los casos de uso principales deben ser usados para validar las elecciones arquitecturales hechas en diferentes niveles.

Las vistas que se usan en RUP son las desarrolladas por Phillippe Krutchen. Estas son [Kruchten 1995]:

Vista de Escenarios / Casos de Uso: Se toman los casos de uso más significativos representando la interacción entre los objetos y procesos. Se puede usar el diagrama de Caso de Uso.

Vista Lógica: Define el modelo de diseño del sistema. Se puede usar diferentes diagramas de UML tales como Capas, Clases, Secuencia y diagrama de Actividades.

Vista de Desarrollo: Define el sistema desde el punto de vista del desarrollador. Este se enfoca en la organización modular del software tales como subsistemas, librerías, frameworks, etc. Se puede usar diferentes diagramas UML como diagrama de Componentes que describen los componentes del sistema y diagrama de Paquetes para representar la vista de desarrollo.

Vista de Proceso: Captura los aspectos de concurrencia y de sincronización del diseño describiendo los procesos de negocio y los componentes que dan soporte a estos procesos. Se puede usar el diagrama de Actividades. Vista de Despliegue: Describe la topología de los componentes de Software en la infraestructura física y los conectores entre estos componentes. Se puede usar el diagrama de Despliegue.

Capítulo 4: Definición de los atributos de calidad

Este capítulo presenta una herramienta para la especificación de los atributos de calidad dentro de la etapa de análisis de la arquitectura de software. Se presenta paso a paso el método a seguir para definir claramente los atributos de calidad. También se describe en qué momento hacer uso de las plantillas generadas y además de cómo hacerlo.

Taller de atributos de calidad (QAW)

Una funcionalidad describe la capacidad de un sistema, pero un sistema comprende más que características enteramente funcionales, también comprende características no funcionales que son expresadas en atributos de calidad. En la elaboración de un sistema, nuestra preocupación usualmente va más allá que tener respuestas funcionalmente correctas, un sistema que se esté realizando para atender un propósito claro debe tomar en cuenta tanto los requerimientos funcionales como los atributos de calidad. Por ejemplo, si un sistema no cumple con un criterio de rendimiento, a pesar de que la respuesta que produzca sea correcta, esta puede no llegar a tiempo. Las cualidades de un sistema son tan importantes como las funcionalidades.

El logro de atributos de calidad es crítico para el éxito del sistema. A pesar de las técnicas cualitativas y cuantitativas usadas para el análisis específico de los atributos de calidad, por si solas, no son lo suficientemente consistentes ni para el diseño ni para la evaluación de una arquitectura de software.

Es muy importante poder determinar con la mayor exactitud posible cuales son los atributos de calidad con mayor importancia en el desarrollo de un sistema. Por ello, los requerimientos de calidad deben ser descritos concretamente antes de comenzar con la elaboración de una arquitectura. En caso de ser obviados, la arquitectura obtenida por el arquitecto carecerá de estructura. Frente a esto, existe el taller de captura de atributos de calidad (QAW), el cual es un método que permite involucrar a los interesados (Stakeholders), los cuales serán afectados directa o indirectamente con el sistema en elaboración.

Es por eso, que el QAW se enfoca y centra en ellos, y es importante ser realizado antes de que comience la creación de la arquitectura. Este taller ofrece una oportunidad para que se puedan juntar todos los interesados en el sistema y poder, además, capturar sus necesidades y expectativas con respecto al proyecto, ahí comienzan a resaltar los distintos atributos de calidad que son de particular interés. Toda esta información es capturada y sometida al método que busca poder organizar, priorizar y aclarar dichos atributos.

La importancia de que este método se desarrolle en el inicio del ciclo de vida del proyecto, se debe a que los componentes, así como los mecanismos de comunicación y los planos para la arquitectura deben ser diseñados o seleccionados de manera que puedan satisfacer y atender las necesidades no funcionales del sistema a producir.

Durante la elaboración del taller se identifican escenarios en los que participan los involucrados y se describen desde su punto de vista. Estos escenarios pueden ser luego usados por los ingenieros para analizar, proponer la mejor arquitectura y posiblemente también estrategias de mitigación en algunos casos. Un escenario específico del sistema es usado para describir de manera más clara los atributos de calidad que son importantes en el sistema y cuáles deberían ser sus respuestas a dichos atributos de calidad. Los escenarios son historias cortas que describen una interacción con el sistema que ejercita un atributo de calidad particular. Por ejemplo, para un requerimiento de capacidad de modificación:

“Un componente de generación de eventos mejorado está disponible para el sistema; el sistema debe permitir a los ingenieros remover el viejo generados de eventos e incorporar el nuevo en menos de dos semanas-hombre.”

Un escenario debe tener, una respuesta y un estímulo claros. El estímulo describe un agente o factor que inicia la reacción en el sistema de alguna manera. La respuesta es la reacción del sistema al estímulo. Se debe también indicar y poner en claro cómo se mide la respuesta. También se debe incluir el entorno en donde el escenario toma lugar.

El taller de QAW depende de la participación de los involucrados en el sistema, tales como usuarios finales, administradores, redes, centros de atención, entrenadores, arquitectos, adquisidores, ingenieros etc. En ese sentido, debe tener al menos cinco integrantes y no exceder los treinta en un solo taller.

La colaboración de los involucrados es esencial durante un taller de QAW. Los participantes son libres de comentar y realizar preguntas durante cualquier momento del taller. Cuando las discusiones se excedan del tiempo determinado o cuando se salgan del foco deseado, los moderadores o facilitadores pueden darlas por terminado. Es importante que los involucrados permanezcan concentrados, puntuales, y con un límite de discusiones por día.

El primer paso del QAW es la presentación e introducción, donde los facilitadores describen la motivación e importancia de desarrollar un taller de este tipo. En este paso también se presentan todos los asistentes, comentan sobre su experiencia, roles de trabajo y su relación con el sistema que será construido.

Como segundo paso, se realiza una presentación de la misión y el negocio por parte de los involucrados que representen al negocio. Es importante que se explique cómo se involucra el sistema en el negocio. Usualmente un representante ejecutivo o director general se toma una hora presentando la misión y contexto del sistema con el negocio de la empresa, los requerimientos funcionales, restricciones y requerimientos de atributos de calidad. Por eso, es importante que los facilitadores estén escuchando atentamente para poder capturar cualquier información relevante que pueda dar luz sobre los atributos de calidad más importantes, sobre los que se basara el diseño y construcción de la arquitectura.

El tercer paso es la presentación del plan de la arquitectura. Generalmente, se han creado artefactos que son descripciones de alto nivel del sistema, gráficos u otros que ayuden a entender el sistema, estos se presentan y exponen mientras que los facilitadores continúan capturando aspectos claves en la presentación para futura referencia.

El cuarto paso es la identificación de conductores de la arquitectura. Luego del segundo y tercer paso, los facilitadores capturaron información que corresponde a conductores arquitecturales. Estos son presentados en forma de lista y en caso se requiera deben pedir aclaraciones. La idea es obtener un consenso sobre la lista de conductores arquitecturales que se muestran. Esta lista final luego ayuda a los involucrados a enfocarse durante los siguientes pasos para asegurar que se estén cubriendo todas las necesidades.

El quinto paso es realizar una lluvia de ideas de escenario, esto se logra a través del facilitador. Cada involucrado en el sistema generará escenarios teniendo en cuenta las partes del mismo (estímulo, entorno y respuesta) y se deben asegurar que cada escenario este correctamente formado. Los involucrados deben expresar sus necesidades en cada escenario. Se pueden

producir dos rondas para que al menos cada involucrado contribuya con dos escenarios. Es importante que no se termine este paso, sin al menos haber creado un escenario por conductor de arquitectura.

Luego de la lluvia de ideas, el siguiente paso consiste en la consolidación de escenarios. En este punto se cuenta con una serie de escenarios y este paso se considera para poder refinarlos con el fin de consolidarlos.

El séptimo paso consiste en priorizar los escenarios. A cada involucrado se le brinda un número de votos total igual al 30% del número total de escenarios generados. Los votos se realizan en turnos y en 2 rondas. Luego se contabilizan los votos y se realiza la priorización debidamente.

Para terminar, se realiza un refinamiento y aclaración de los escenarios. Mientras el tiempo lo permita, se escogen escenarios al azar y se busca detallarlos a un mayor nivel. Se deben enfocar en detallar el estímulo, la respuesta, la fuente del estímulo, el entorno, el artefacto estimulado y la medida de la respuesta. También es posible presentar preguntas y observaciones sobre los escenarios, las preguntas deben enfocarse en los atributos de calidad y cualquier duda que se pueda tener sobre cómo lograr la respuesta descrita en el escenario.

Capítulo 5: Diseño basado en atributos (ADD)

Este capítulo presenta cada uno de los pasos para poder diseñar la arquitectura basándose en candidatos arquitecturales. Para ello, el sistema es dividido en elementos que son analizados uno a uno en todo el proceso de diseño.

Diseño basado en atributos (ADD)

El Diseño Basado en Atributos (Attribute-Drive Design - ADD) fue elaborado por el Instituto de Ingeniería de Software de la Universidad Carnegie Mellon. El método ADD es un enfoque para definir una arquitectura de software, en el cual el proceso está basado en los requerimientos de atributos de calidad de software. ADD comprende un proceso recursivo que descompone un sistema o elemento de un sistema aplicando tácticas y patrones arquitecturales que satisfagan sus requerimientos de atributos de calidad.

ADD como todo proceso posee entradas y salidas. Las entradas de ADD son los requerimientos funcionales, restricciones de diseño y requerimientos de atributo de calidad del sistema. Los requerimientos funcionales especifican que funciones un sistema debe proveer para alcanzar las necesidades explícitas o implícitas de los interesados cuando un software es usado bajo condiciones específicas [4]. Además, las restricciones de diseño son decisiones acerca del diseño de un sistema que deben ser incorporadas en el diseño final del sistema. Por ejemplo; la interacción con sistemas ya desarrollados, el uso de tecnologías específicas para el desarrollo, el uso de tecnologías específicas para la Base de Datos, el uso de un sistema operativo específico. Finalmente, los requerimientos de atributo de calidad son requerimientos que indican los grados a los cuales un sistema deberá exponer ciertas propiedades. Por ejemplo: el sistema deberá recuperarse de una caída en no más de un segundo (disponibilidad), el sistema deberá procesar la entrada del sensor en un segundo (Desempeño/Rendimiento), el sistema deberá denegar el acceso a usuarios no autorizados el 100% del tiempo (Seguridad), entre otros. Cabe resaltar que los requerimientos pueden llegar implícitos y se deben expresar como elementos únicos. Con respecto a las salidas de ADD se conforman por un diseño de sistema en términos de roles, responsabilidades, propiedades y relaciones entre los elementos de software que lo conforman. Además, este diseño es documentado usando los diferentes tipos

de vistas arquitecturales como las Vistas Modulares, Vistas Componente-Conector y Vistas de Asignación.

En los siguientes párrafos se describirán brevemente todos los pasos que involucra ADD. Paso 1: Confirmar Requerimientos – En este primer paso, se confirma si existe suficiente información acerca de los requerimientos para proceder con ADD. En esencia, se asegura que los interesados del sistema hayan priorizados los requerimientos de acuerdo a las metas de la misión y del negocio. También, se deberá confirmar si existe suficiente información acerca de los requerimientos de atributos de calidad para proceder con ADD.

Paso 2: Elegir Elemento a Descomponer - usted elige qué elemento del sistema será el objetivo de diseño en los pasos subsiguientes. Si es la primera vez que se llega este pasó, el elemento a elegir es el propio sistema. De lo contrario, el sistema ha sido divido en dos o más elementos y los requerimientos han sido asignados a estos elementos y se debe elegir uno de estos elementos basándose en estas 4 áreas: conocimiento actual de la arquitectura, riesgo y dificultad, criterio del negocio y criterio organizacional.

Paso 3: Identificar los Candidatos a Conductores Arquitecturales - Durante este paso, se prioriza los requerimientos por segunda vez basados en su relativo impacto en la arquitectura. Esta segunda priorización puede ser tan simple como asignar “alto impacto”, “mediano impacto” y “bajo impacto” a cada requerimiento. Cada requerimiento podría plantearse como un par de letras, cuyas opciones serían H (alto), M (medio), L (bajo). La primera letra correspondería a la priorización del interesado mientras que la segunda a la priorización según el impacto en la arquitectura. Con esto se vuelve a ordenar tomando en cuenta que una combinación H, H es la puntuación más alta. Entonces, de estos pares se debe elegir 5 o 6 requerimientos con las mayores prioridades, los cuales serán llamados “candidatos a conductores arquitecturales” para el elemento actual que se está descomponiendo.

Paso 4: Elegir un Concepto de Diseño - se debe elegir los tipos de elementos más importantes que aparecerán en la arquitectura y los tipos de relaciones entre los mismos. Las restricciones de diseño y los requerimientos de atributo de calidad (los cuales son candidatos a conductores arquitecturales) son usados para determinar los tipos de elementos, relaciones y sus interacciones. En este paso se deben listar todos los patrones que podrían solucionar los requerimientos para luego realizar un cuadro de doble entrada donde se seleccionarán los patrones que mejor satisfacen a los candidatos a conductores arquitecturales.

Paso 5: Generar Instancias de los Elementos Arquitecturales - De cada tipo de elemento de software se crean instancias y se les asigna responsabilidades de acuerdo al tipo de elemento. Estas responsabilidades también son derivadas de los requerimientos funcionales asociados a los candidatos a conductores arquitecturales. Al final de este paso, cada requerimiento funcional asociado con el elemento padre debe ser representado por una secuencia de responsabilidades dentro de los elementos hijos. Cabe resaltar que, se debe analizar y documentar las decisiones de diseño mediante los diferentes tipos de vistas arquitecturales como las Vistas Modulares, Vistas Componente-Conector y Vistas de Asignación.

Paso 6: Definir Interfaces – Se define los servicios y las propiedades requeridas y proporcionadas por los elementos de software en nuestro diseño. En ADD, estos servicios y propiedades son llamados como “Interfaces de los Elementos”. Nótese que una interfaz no es simplemente una lista de firmas de operaciones. Las interfaces describen los supuestos que PROPORCIONA y REQUIERE que los elementos de software hacen uno u otro.

Paso 7: Verificar y Refinar Requerimientos - En este paso, se verifica que la descomposición del elemento hasta ahora cumple con los requerimientos funcionales, requerimientos de atributo de calidad y las restricciones de diseño. Además, se debe preparar los elementos hijos para las descomposiciones posteriores.

Paso 8: Verificar Fin de Descomposición – En este paso se verifica que el elemento a descomponer sea el último elemento analizado. Si lo es, el proceso termina, de lo contrario se retorna al Paso 2 para seguir con la descomposición.

Con estos 8 pasos se puede realizar una documentación de una arquitectura según los lineamientos del SEI. Estos pasos han sido resumidos y para más detalles se debe remitir a la documentación de ADD.

Capítulo 6: Implementación

Este capítulo explica cómo se aplicó el conocimiento generado con la ayuda de las guías, plantillas y checklist en un escenario real. Además, describir cómo el conocimiento generado permitió al equipo a proponer una actualización en dos paquetes de despliegue de la ISO/IEC 29110, para lo cual se realizó la propuesta directamente con los profesores Claude y Laporte, ambos PhD. de la Escuela de Tecnología Superior (ETS). También de la propuesta de solución realizada a un problema del Profesor Félix Bachmann PhD. del SEI. Finalmente, se explicará se debe realizar una capacitación sobre el conocimiento generado.

Presentación del caso

uReader es desarrollado para Ummitech, empresa especializada en el desarrollo de soluciones para dispositivos móviles. uReader es una aplicación que permite a los usuarios escuchar la lectura de sus fuentes de información. Además, le permite al usuario ejecutar acciones sobre la aplicación con comandos de voz. Esto es posible dado que uReader incorpora componentes de reconocimiento de voz y lectura de texto. Ummitech considera como características de alta importancia:

- El rendimiento de la aplicación y capacidad de respuesta lo más altas posibles: El cliente considera que lograr un alto rendimiento de la aplicación es crucial para poder competir en el mercado actual que cuenta con aplicaciones similares.

Es importante entender el contexto bajo el cual nace la idea de uReader. En atención a requerimientos planteados por la organización cliente Ummitech, la cual desarrolla productos que ayudan a personas con discapacidad visual, nace la necesidad de realizar una aplicación enfocada en poder ayudar a las personas que padecen alguna discapacidad visual grave y atender a sus diversas necesidades identificadas como:

- Mantenerse actualizados de las novedades de fuentes de información (noticias, blogs, etc.) a tiempo real desde un smartphone y de manera autónoma.
- Tener la habilidad de pausar, detener, avanzar y retroceder fácilmente entre las novedades de las fuentes de información.

- Poder administrar (CRUD) las fuentes de información en la aplicación (No necesariamente debe realizarse de manera autónoma).

La aplicación móvil en su primera versión, estará disponible en smartphones con sistema operativo Windows Phone 8.

Una representación del contexto del sistema uReader se muestra en la ilustración 2.

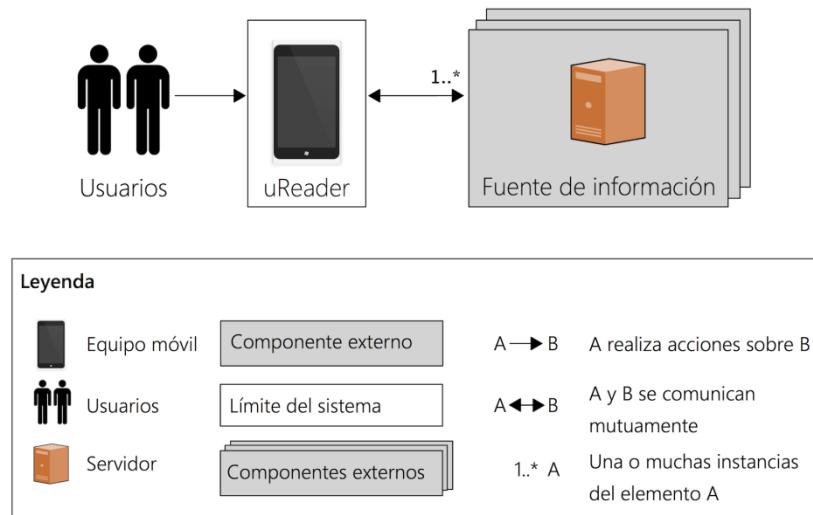


Figura 2. Diagrama de Contexto de uReader

Elaboración propia

Las interfaces de usuario (UI) del sistema uReader se muestran a continuación.



Figura 3. Pantalla de Agregar nueva Fuente de Información

Elaboración propia

En la ilustración 4 se aprecia la interfaz de usuario responsable de la adición de una nueva fuente de información a uReader.

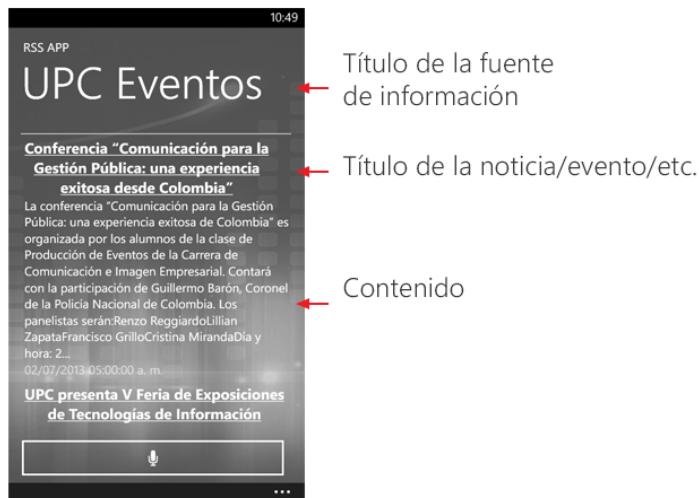


Figura 4. Pantalla de contenido de un elemento (noticia, evento, etc.) de una fuente de información

Elaboración propia

En la ilustración 5 se aprecia la interfaz de usuario responsable de mostrar el contenido de un elemento recibido de una fuente de información específica.



Figura 5. Pantalla de listado de fuentes de información

Elaboración propia

En la ilustración 6 se aprecia la interfaz de usuario responsable de la adición de una nueva fuente de información a uReader.

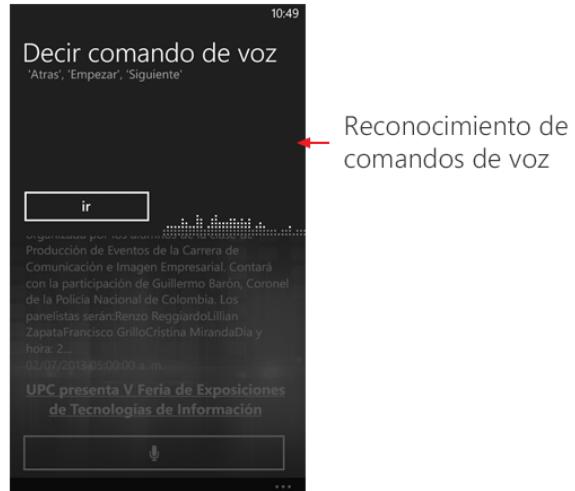


Figura 6. Pantalla de ingreso de comando de voz

Elaboración propia

En la ilustración 6 se aprecia la interfaz de usuario responsable del reconocimiento de voz.

Reconocimiento de voz: Elemento gráfico responsable de mostrar el estado del proceso de reconocimiento en el componente SpeechRecognizer

Funcionalidades

uReader permite la administración de diferentes fuentes RSS. Es decir, se pueden agregar o eliminar diferentes fuentes de RSS a la aplicación. En este sentido, el usuario podrá elegir de la lista de fuentes de RSS y se mostrará el listado de eventos de la fuente elegida.

Por otro lado, uReader también permite reconocer comandos de voz del usuario que el sistema interpreta y realiza acciones basado en la configuración de los comandos. Los comandos considerados para la versión 1.0 se muestran en la tabla 5:

Tabla 5. Listado de los comandos de voz y acción esperada.

Comando de Voz	Descripción de acción esperada
Leer	<p>La aplicación comienza a leer la publicación actualmente seleccionada.</p> <p>En caso de no contar aún con una publicación seleccionada, como cuando el usuario apenas ingresa a la aplicación, la aplicación por defecto selecciona la primera publicación disponible de la primera fuente de noticias disponible.</p> <p>En caso de no contar con ninguna publicación disponible, el sistema le comunicará al usuario que debe agregar primero una fuente de noticias.</p>
Pausar	La aplicación pausa la lectura del artículo actual, con capacidad de reanudar la lectura.
Parar	La aplicación detiene la lectura del artículo actual sin capacidad de reanudar la lectura.
Siguiente	La aplicación se desplaza a la siguiente publicación y la comienza a leer.
Atrás	La aplicación se desplaza a la publicación anterior y la comienza a leer.

Elaboración propia

Restricciones

Las restricciones definen las limitaciones o límites en los que la aplicación uReader es creado.

Las siguientes restricciones fueron identificadas:

Restricciones relacionadas a la tecnología:

- La aplicación debe ser desarrollada para Windows Phone y debe tener soporte de la versión 8 en adelante.
- La aplicación debe ser desarrollada empleando C#.
- El componente a emplear para la funcionalidad de voice-recognition debe ser SpeechRecognizer.
- El componente a emplear para la funcionalidad de text-to-speech debe ser SpeechSynthesizer.

Restricciones relacionadas a los sistemas externos:

- Todas las fuentes de información deben publicar la data empleando RSS 2.0 y siguiendo correctamente con su formato.

QAW

Como parte de la implementación de los lineamientos para la elaboración de una arquitectura se realizó un taller con todos los interesados para realizar la captura de los atributos de calidad, filtrarlos y plasmarlos en escenarios de atributos de calidad (EAC).

Atributos de Calidad

Usabilidad

- EAC 1: La aplicación deberá alertar con un mensaje auditivo “RSS cargado” en un tiempo no mayor a 3 segundos cuando la información del RSS seleccionado por el usuario ya ha sido cargada del servidor.
- EAC 2: Cuando un usuario esté descargando la data de una Fuente de información en la aplicación de uReader, se deberá mostrar el porcentaje actual de la descarga en la barra de estado del dispositivo móvil.
- Rendimiento
- EAC 3: La aplicación deberá procesar el XML proveniente del RSS Server en condiciones normales (RSS versión 2.0 y XML no mayor a 900KB) en un tiempo no mayor a 2 segundos.

Capacidad de modificación

- EAC 4: Se debe ser capaz de reemplazar, agregar o remover los componentes de reconocimiento de voz y de lectura de texto en un esfuerzo no mayor a una semana hombre.

Capacidad de verificación

- EAC 5: La aplicación deberá capturar con gran facilidad por lo menos el 90% de fallas generadas en el administrador de eventos de lectura y reconocimiento de voz en el ambiente de producción.

Diseño basado en atributos (ADD)

El segundo paso dentro de la elaboración de la arquitectura de software es el diseño de la misma. Al ya contar con los atributos de calidad procedemos a priorizarlos para poder enfocarnos principalmente en los escenarios de mayor complejidad desde el punto de vista arquitectural y de mayor prioridad para los interesados.

Considerar los siguientes posibles valores para el primer elemento del par de prioridad:

- **Alto:** Si este escenario no es satisfecho, el sistema será considerado como no exitoso.
- **Medio:** Este escenario es altamente deseado para el sistema; pero si existe una razón muy buena que justifique por qué no puede ser empleado, el sistema será considerado como exitoso.
- **Bajo:** Este escenario no es imperativo, pero sería muy bueno tenerlo en el sistema.

Considera los siguientes posibles valores para el segundo elemento del par de prioridad:

- **Alto:** El equipo de arquitectura no sabe cómo satisfacer el escenario.
- **Medio:** El equipo de arquitectura sabe cómo satisfacer el escenario, pero es muy complicado de realizar.
- **Bajo:** El equipo de arquitectura sabe cómo satisfacer el escenario y lo considera fácil de realizar.

Tabla 6. Tabla de organización del proyecto

Atributos de Calidad	Caracterización del atributo	Escenarios	Prioridad
Usabilidad	Tiempo de la tarea (task time)	La aplicación deberá alertar con un mensaje auditivo “RSS cargado” en un tiempo no mayor a 3 segundos cuando la información del RSS seleccionado por el usuario ya ha sido cargada del servidor.	(M,M)
Usabilidad	Dar a conocer al usuario (gain user knowledge)	Cuando un usuario esté descargando la data de una Fuente de información en la aplicación de uReader, se deberá mostrar el porcentaje actual de la descarga en la barra de estado del dispositivo móvil.	(M,B)
Performance	Latencia (latency)	La aplicación deberá procesar el XML proveniente del RSS Server en condiciones normales (RSS versión 2.0 y XML no mayor a 900KB) en un tiempo no mayor a 2 segundos.	(A,M)
Capacidad de modificación	Esfuerzo (effort)	Se debe ser capaz de reemplazar, agregar o remover componentes en un esfuerzo no mayor a una semana hombre.	(B,B)
Testability	Aserciones ejecutables (executable assertions)	La aplicación deberá capturar con gran facilidad por lo menos el 90% de fallas generadas en el administrador de eventos de lectura y reconocimiento de voz en el ambiente de producción.	(M,B)

Elaboración propia

Bajo las priorizaciones de los escenarios de atributos de calidad se deben realizar las vistas necesarias para poder dar soporte a estos conductores arquitecturales. El Instituto de Ingeniería de Software señala tres grandes grupos de vistas arquitecturales: Modulares, Componentes y Conectores y de Asignación.

Capítulo 7: Gestión del proyecto

Este capítulo explica cómo se realizó la gestión del proyecto. Es decir, incluye toda la información de Gestión de Recursos Humanos, Planeamiento, Gestión de Comunicación, Gestión de Riesgos y las lecciones aprendidas del proyecto.

Producto final

A continuación, se detalla el producto final obtenido según los identificadores de éxito planteados para el proyecto:

- Para el proceso de captura de atributos de calidad (QAW) se generó:
 - Guía en español de cómo realizar la captura de atributos de calidad (QAW).
 - Proceso definido y caracterizado de la captura de atributos de calidad (QAW).
 - Plantilla de apoyo para el taller de captura de atributos de calidad (QAW).
 - Plantilla de apoyo y ejemplos para la captura de escenarios.
 - Plantilla de apoyo y ejemplos para el refinamiento de escenarios. Estado:
- Para el proceso de diseño de una arquitectura de software basado en atributos (ADD) se generó:
 - Guía en español de cómo realizar un diseño de arquitectura de software basado en atributos (ADD).
 - Proceso definido y caracterizado del diseño de una arquitectura de software basado en atributos (ADD).
 - Guía en español de cómo integrar los métodos de captura de atributos de calidad (QAW) y el diseño de una arquitectura de software basado en atributos (ADD).
 - Plantilla de la matriz de patrones y conductores arquitecturales.
 - Checklist del diseño de una arquitectura basado en atributos (ADD).
- Para el proceso de elaboración de una arquitectura de software se generó:
 - Plantilla completa de un documento de arquitectura de software (SAD).

- Plantilla light de un documento de arquitectura de software (SAD).
 - Plantilla del anexo para el análisis del enfoque arquitectural.
 - Plantilla del anexo para la documentación de decisiones arquitecturales.
 - Ejemplo 1 en español de un documento de arquitectura de software (SAD).
 - Ejemplo 2 en español de un documento de arquitectura de software (SAD).
- Sobre el proyecto piloto realizado se generó.
 - Documento de Arquitectura de Software Light
 - Anexo de Análisis de un enfoque arquitectural
 - Sobre la capacitación de las empresas virtuales quality assurance y software factory.
 - Presentación del taller
 - Acta de reunión

Gestión del tiempo

A continuación, se muestra el cronograma de trabajo efectivamente realizado:

- Formalización de procesos de documentación y evaluación de Arquitectura de Software	1.422 hrs 169 días?	mar 09/04/13	jue 28/11/13		
CG - Publicación de rol de exposiciones de charters y conograma TP1	0 hrs 0 días	mar 09/04/13	mar 09/04/13		
+ Planificación	48 hrs 3 días	mar 09/04/13	jue 11/04/13		
- Investigación	322 hrs 24 días	jue 11/04/13	mar 14/05/13	3	
- Documentación de una Arquitectura de Software	322 hrs 24 días	jue 11/04/13	mar 14/05/13		
CG - Exposición de charters y cronogramas TP1	0 hrs 0 días	jue 11/04/13	jue 11/04/13		
Recopilación de información	2 hrs 4 días	vie 12/04/13	mié 17/04/13	3	Cesar Gonzales[3%];Jose Torres[3%]
Estudio de la información	80 hrs 5 días	jue 18/04/13	mié 24/04/13	9	Cesar Gonzales;Jose Torres
- Proceso de documentación	240 hrs 15 días	jue 25/04/13	mar 14/05/13	10	
Definición del proceso	48 hrs 3 días	jue 25/04/13	lun 29/04/13		Cesar Gonzales;Jose Torres
Representación del proceso	32 hrs 2 días	mar 30/04/13	mié 01/05/13	12	Cesar Gonzales;Jose Torres
+ Revisión QA	96 hrs 6 días	mié 01/05/13	mié 08/05/13	13	
Generación de artefactos involucrados en el proceso	64 hrs 4 días	jue 09/05/13	mar 14/05/13	14	Cesar Gonzales;Jose Torres
Inicio de actividades ciclo académico 2013-2	0 hrs 0 días	lun 12/08/13	lun 12/08/13		
+ Modificación de los Deployment Packages de la ISO/IEC29110	176 hrs 14 días	jue 05/09/13	mar 24/09/13		
- Realización de piloto	404 hrs 78,5 días	lun 12/08/13	jue 28/11/13	26	
- Proceso de documentación de la arquitectura	404 hrs 78,5 días	lun 12/08/13	jue 28/11/13		
Primera Entrega de ejemplo SAD	208 hrs 26 días	lun 12/08/13	lun 16/09/13		Cesar Gonzales[50%];Jose Torres[50%]
Realización de Taller de Atributos de Calidad	32 hrs 4 días	vie 13/09/13	mié 18/09/13		Cesar Gonzales[50%];Jose Torres[50%]
Selección del proyecto piloto	24 hrs 3 días	mar 24/09/13	jue 26/09/13		Cesar Gonzales[50%];Jose Torres[50%]
Revisión del proyecto	24 hrs 3 días	vie 27/09/13	mar 01/10/13	39	Cesar Gonzales[50%];Jose Torres[50%]
Entrega de guías y documentación	0 hrs 0 días	vie 08/11/13	vie 08/11/13		
Instrucción de uso de las guías	32 hrs 4 días	vie 08/11/13	mié 13/11/13	41	Cesar Gonzales[50%];Jose Torres[50%]
Supervisión y control durante el uso de las guías	60 hrs 7,5 días	jue 14/11/13	lun 25/11/13	42	Cesar Gonzales[50%];Jose Torres[50%]
Reporte de resultados del piloto	24 hrs 3 días	lun 25/11/13	jue 28/11/13	43	Cesar Gonzales[50%];Jose Torres[50%]
- Presentación de Proyecto	128 hrs 12 días	mar 01/10/13	mié 16/10/13		
Elaboración de Memoria - 3ra Entrega	64 hrs 8 días	mar 01/10/13	jue 10/10/13		Cesar Gonzales[50%];Jose Torres[50%]
Revisión BankMin: Memoria - 3ra Entrega	0 hrs 0 días	jue 10/10/13	jue 10/10/13	46	
Entrega Memoria - 3ra Entrega	0 hrs 0 días	jue 10/10/13	jue 10/10/13	47	
Elaboración de Presentación	64 hrs 4 días	vie 11/10/13	mié 16/10/13	46	Cesar Gonzales;Jose Torres
Exposición del Proyecto	0 hrs 0 días	mié 16/10/13	mié 16/10/13	49	
- Cierre de Proyecto	344 hrs 31 días	jue 17/10/13	jue 28/11/13	45	
Elaboración Perfil de Proyecto	192 hrs 12 días	jue 17/10/13	vie 01/11/13		Cesar Gonzales;Jose Torres
Entrega de Perfil de Proyecto	0 hrs 0 días	vie 01/11/13	vie 01/11/13	52	
Elaboración Reporte de Cierre de Proyecto	56 hrs 7 días	lun 04/11/13	mar 12/11/13	53	Cesar Gonzales[50%];Jose Torres[50%]
Elaboración de Memoria - 4ta Entrega	72 hrs 9 días	mié 13/11/13	lun 25/11/13	54	Cesar Gonzales[50%];Jose Torres[50%]
Revisión BankMin: Memoria - 4ta Entrega	0 hrs 0 días	lun 25/11/13	lun 25/11/13	55	
Entrega Memoria - 4ta Entrega	0 hrs 0 días	lun 25/11/13	lun 25/11/13	56	
Elaboración de Presentación	24 hrs 3 días	mar 26/11/13	jue 28/11/13	55	Cesar Gonzales[50%];Jose Torres[50%]
Exposición del Proyecto	0 hrs 0 días	jue 28/11/13	jue 28/11/13	58	
+ Reuniones programadas del proyecto	0 hrs 133 días?	mar 16/04/13	mié 16/10/13		

Figura 7. Cronograma del proyecto

Elaboración propia

Gestión de los recursos humanos

A continuación, se detallan los roles desempeñados dentro del proyecto, así como las personas que los cubrieron.

Tabla 7. Tabla de organización del proyecto

Responsable	Roles	Responsabilidades
César Gonzales Yapapasca	Jefe de Proyecto	Llevar a cabo la investigación y gestionar las actividades de los recursos involucrados.
José Torres Cárdenas	Jefe de Proyecto	Llevar a cabo la investigación y gestionar las actividades de los recursos involucrados.
Comité de evaluación de Proyectos de la UPC	Comité de proyectos	Evaluación del proyecto enfocándose en la presentación de la documentación.
Asesor	Prof. Luis García Paucar	Brindar asesoría durante el proyecto de investigación.
Cliente	Prof. Luis García Paucar	Brindar las necesidades y alcance del proyecto de investigación.

Elaboración propia

Gestión de las comunicaciones

A continuación, se describe cómo efectivamente se gestionaron las comunicaciones dentro del proyecto, y los convenientes que surgieron:

- Reportes de monitoreo
 - Se emplearon reportes de monitoreo para demostrar el avance a la empresa virtual a la que pertenece el proyecto Bankmin y además mostrar el avance al representante del cliente.
- Informes de seguimiento
 - Se emplearon informes de seguimiento con el asesor y el gerente de la empresa virtual a la que pertenece el proyecto Bankmin a manera de dar a conocer al asesor el estado, acuerdos, cambios del proyecto.
- Actas de reunión
 - Se emplearon actas de reunión para dejar constancia de los acuerdos de actividades, entregas, cambios en el proyecto a lo largo del desarrollo del mismo.

Gestión de los riesgos

A continuación, se muestran los riesgos enfrentados, y las actividades de mitigación empleadas en el proyecto:

Tabla 8. Tabla de riesgos

#	Riesgo	Probabilidad	Impacto	Estrategia de mitigación
1	Incumplimiento con las reuniones con el experto.	Media	Alto	Reprogramación de la reunión.
2	Dificultades para la elección del proyecto piloto.	Media	Alto	Uso de un proyecto externo a las empresas virtuales.
3	Falta de apoyo con un recurso asignado al proyecto.	Bajo	Alto	Distribución del trabajo del recurso entre los jefes de proyecto.

Elaboración propia

Lecciones aprendidas

A continuación, se señalan los puntos que deberían mejorarse y tenerse en cuenta para futuros proyectos:

- Mejorar los canales de comunicación con el asesor y con el cliente del proyecto.
- Asegurarse de tener un alcance cuantificable.
- Manejar adecuadamente la gestión de cambios del proyecto.
- Registrar en actas todos los acuerdos pactados con el cliente.

Conclusiones

Luego de la investigación realizada sobre el proceso de elaboración de una arquitectura de software basado en el conocimiento generado por el Software Engineering Institute (SEI) de la universidad Carnegie Mellon (CMU) y basado en las sugerencias recibidas de los profesores Félix Bachmann y James McHale del Software Engineering Institute (SEI) luego de la exposición de la presente investigación que se les hizo, se concluye que:

- El proceso de elaboración de una arquitectura de software en teoría suele definirse como una etapa de análisis, una de diseño, una de evaluación y finalmente una de documentación. Esto en la práctica es poco cierto, dado que durante las etapas previas a la documentación ya se genera el conocimiento y artefactos necesarios para partes de la documentación. Entonces es recomendable, exigir un poco más de esfuerzo en la elaboración del conocimiento obtenido durante las tres etapas iniciales, de manera que la última etapa es eliminada. De esta manera, la documentación se convierte en una tarea transversal a lo largo de la elaboración de una arquitectura de software y no una etapa más.
- Es muy importante darle énfasis al resumen del documento dentro de la sección de Introducción del documento de arquitectura de software. Dado que esta parte esta propuesta para servir a manera de una capa sobre las demás secciones. En dicho sentido es a partir de esta sección donde un interesado del sistema puede tomar sólo las partes específicas del documento y obviar partes que no le van a proveer de información útil.
- Para la diagramación a realizar en la documentación de las vistas, es recomendable emplear una notación UML (Unified Modeling Language). Esto debido a que al ser una notación estandarizada permite que un interesado pueda comprender rápidamente los diagramas de la documentación sin tener que revisar antes la leyenda del diagrama.
- La investigación se apoya en dos métodos desarrollados por el SEI: el taller de atributos de calidad (QAW) y el diseño basado en atributos (ADD). Sin embargo, se concluye que es necesaria la incorporación de dos métodos adicionales al proceso: revisiones activas del diseño (ARID) y el método de análisis de compensación de la arquitectura (ATAM). Ambos métodos complementan en la parte de evaluación de una arquitectura de software, de manera parcial y total respectivamente.

- A pesar de que inicialmente se proponen plantillas para la documentación de una arquitectura de software. Basados en la evidencia encontrada y sugerencias obtenidas, el manejo de la documentación de una arquitectura de software debe realizarse con el apoyo de una herramienta de software especialmente diseñada para esta labor. El hacerlo reduce los problemas que usualmente están relacionados a la documentación de una arquitectura de software como el mantenimiento y la evolución de la documentación.

Recomendaciones

Como parte de futuros trabajos que ayuden a contribuir al conocimiento base generado se propone:

- La elaboración de guías, plantillas y ejemplo para los métodos de:
 - Revisiones activas del diseño (ARID)
 - Análisis de compensación de la arquitectura (ATAM)
- La publicación de todo el conocimiento generado en una página Wikipedia de la universidad que permita a los alumnos siempre contar con la última documentación relacionada a la elaboración de una arquitectura de software.
- La implementación de un producto software que sirva de asistente y contribuya a la documentación de una arquitectura de software siguiendo el conocimiento generado.

Glosario

- Ensemble Interactions: Definición teórica que define una serie de patrones para la implementación de aplicaciones multiplataforma.
- Cross Device Interactions: Concepto derivado de Ensemble Interactions orientado a la comunicación y transferencia de información entre dispositivos.
- Prueba de Concepto: Aplicativo software que busca demostrar una viabilidad de funcionalidades que pueden comprender un producto final.
- Aplicación: Programa informático que puede controlar y comprender hardware y software para permitir a un usuario realizar una o más tareas.
- Hardware: Elementos físicos y tangibles que presenta un dispositivo informático.
- Software: Conjunto de programas informáticos que pueden realizar una serie de acciones lógicas utilizando el Hardware de un dispositivo.
- Sistema Operativo: Software capaz de controlar el Hardware de un dispositivo.
- Framework: Marco de trabajo y referencia de buenas prácticas para el desarrollo de un aplicativo informático.

Anexos

Anexo 1: Acta de Reunión 01 – 14/05/2013

April 4, 2013

Acta de Reunión N°1

Sección 1. Información General

Nombre del Proyecto	Fecha de Reunión	Hora de Inicio	Lugar
Formalización de los procesos de documentación y evaluación de arquitecturas de software	19/03/2013	18:00	Oficina del profesor cliente – Universidad Peruana de Ciencias Aplicadas
Asistentes	Cargo	Hora de llegada	
Cesar Gonzales Yapapasca	Jefe de proyecto	18:00	
José Torres Cárdenas	Jefe de proyecto	18:00	
Prof. Luis García Paucar	Cliente	18:00	

Sección 2. Temas tratados (agenda)

Tema:	Descripción:
Presentación del proyecto.	Se realizó una descripción del proyecto por parte del cliente para dejar expresa las necesidades y objetivos que se busca lograr.

Sección 3. Acuerdos

Nº	Acuerdo:	Descripción:	Responsable	Fecha Límite
1	Elaborar un Project Charter con las necesidades expresadas.	Se acordó la elaboración de un documento formal del proyecto donde se expresen las necesidades claras y los objetivos a alcanzar.	Jefes de proyecto	09/04/2013

César Gonzales Yapapasca
Jefe de Proyecto

José Torres Cárdenas
Jefe de Proyecto

Prof. Luis García Paucar
Cliente

Anexo 2: Acta de Reunión 02

April 4, 2013

Acta de Reunión N°2

Sección 1. Información General

Nombre del Proyecto	Fecha de Reunión	Hora de Inicio	Lugar
Formalización de los procesos de documentación y evaluación de arquitecturas de software	02/04/2013	17:00	Salón de BANKMIN C35 – Universidad Peruana de Ciencias Aplicadas
Asistentes	Cargo	Hora de llegada	
Cesar Gonzales Yapapasca	Jefe de proyecto	17:00	
José Torres Cárdenas	Jefe de proyecto	17:00	
Prof. Luis García Paucar	Cliente	17:00	

Sección 2. Temas tratados (agenda)

Tema:	Descripción:
Revisión del avance del Project Charter y de la investigación	Se realizó una revisión del estado actual del avance del Project Charter y sobre la investigación.
Presentación del equipo de trabajo a Software Factory y Quality Assurance.	Se presentó del equipo de trabajo en las empresas virtuales Software Factory y Quality Assurance para la coordinación y apoyo correspondiente al proyecto.

Sección 3. Acuerdos

Nº	Acuerdo:	Descripción:	Responsable	Fecha Límite
1	Entregar el Project Charter durante la presente semana.	Se acordó el envío del Project Charter luego de recibidas las observaciones del profesor gerente de la empresa BANKMIN	Jefes de proyecto	05/04/2013

César Gonzales Yapapasca
Jefe de Proyecto

José Torres Cárdenas
Jefe de Proyecto

Prof. Luis García Paucar
Cliente

Anexo 3: Acta de Reunión 03 – 14/05/2013

Acta de Reunión N°3

Sección 1. Información General

Nombre del Proyecto	Fecha de Reunión	Hora de Inicio	Lugar
Formalización de los procesos de documentación y evaluación de arquitecturas de software	14/05/2013	18:00	Oficina del cliente – Universidad Peruana de Ciencias Aplicadas
Asistentes	Cargo	Hora de llegada	
Cesar Gonzales Yapapasca	Jefe de proyecto	18:00	
José Torres Cárdenas	Jefe de proyecto	18:00	
Prof. Luis García Paucar	Cliente	18:00	

Sección 2. Temas tratados (agenda)

Tema:	Descripción:
Presentación de artefactos del proceso de Documentación de Arquitectura de Software	Se realizó una revisión sobre los artefactos generados hasta el momento propios de la Documentación de Arquitectura de Software.

Sección 3. Acuerdos

Nº	Acuerdo:	Descripción:	Responsable	Fecha Límite
1	Elaborar escenarios de atributos de calidad ejemplos propios del método de documentación de una arquitectura de software.	Se acordó la elaboración de conjunto de escenarios de atributos de calidad para complementar los artefactos generados hasta el momento, con el fin de dejar mayor evidencia y material de apoyo sobre el proceso.	Jefes de proyecto	23/05/2013

César Gonzales Yapapasca
Jefe de Proyecto

José Torres Cárdenas
Jefe de Proyecto

Prof. Luis García Paucar
Cliente

Anexo 4: Acta de Reunión 04 – 21/05/2013

Acta de Reunión N°4

Sección 1. Información General

Nombre del Proyecto	Fecha de Reunión	Hora de Inicio	Lugar
Formalización de los procesos de documentación y evaluación de arquitecturas de software	21/05/2013	18:00	Oficina del cliente – Universidad Peruana de Ciencias Aplicadas
Asistentes	Cargo	Hora de llegada	
Cesar Gonzales Yapapasca	Jefe de proyecto	18:00	
José Torres Cárdenas	Jefe de proyecto	18:00	
Prof. Luis García Paucar	Cliente	18:00	

Sección 2. Temas tratados (agenda)

Tema:	Descripción:
Revisión de artefactos del proceso de Documentación de Arquitectura de Software	Se realizó una descripción del proyecto por parte del cliente para dejar expresa las necesidades y objetivos que se busca lograr.

Sección 3. Acuerdos

Nº	Acuerdo:	Descripción:	Responsable	Fecha Límite
1	Elaborar un resumen descriptivo para los atributos de calidad de • Rendimiento (Performance) • Disponibilidad (Availability) Capacidad de modificación (Modifiability)	Se acordó la elaboración de documentos donde se realice un resumen de cada atributo de calidad señalado y las tácticas relacionadas a estas.	Jefes de proyecto	31/05/2013

César Gonzales Yapapasca
Jefe de Proyecto

José Torres Cárdenas
Jefe de Proyecto

Prof. Luis García Paucar
Cliente

Anexo 5: Acta de Reunión 05 – 04/06/2013

Acta de Reunión N°5

Sección 1. Información General

Nombre del Proyecto	Fecha de Reunión	Hora de Inicio	Lugar
Formalización de los procesos de documentación y evaluación de arquitecturas de software	04/06/2013	18:00	Oficina del cliente – Universidad Peruana de Ciencias Aplicadas
Asistentes	Cargo	Hora de llegada	
Cesar Gonzales Yapapasca	Jefe de proyecto	18:00	
José Torres Cárdenas	Jefe de proyecto	18:00	
Prof. Luis García Paucar	Cliente	18:00	

Sección 2. Temas tratados (agenda)

Tema:	Descripción:
Prueba de métodos para el proceso de Documentación de Arquitectura de Software documentados con estudiantes de la carrera.	Se conversó la posibilidad de realizar una prueba de campo de los
Ajuste del alcance del proyecto para el ciclo académico 2013-1	Se conversó de la necesidad de realizar un ajuste al alcance del proyecto para el ciclo académico 2013-1 dada al cambio de prioridades para las tareas del product-backlog del proyecto.

Sección 3. Acuerdos

Nº	Acuerdo:	Descripción:	Responsable	Fecha Límite
1	Elaborar un taller de captura de atributos de calidad (QAW) en el curso de Arquitectura de Software.	Se acordó coordinar con el profesor a cargo del curso de Arquitectura de Software, Ilver Anache, la fecha del taller QAW y generar el material a usar.	Jefes de proyecto	07/06/2013
2	Actualizar las prioridades de las tareas del product-backlog del proyecto.	Se acordó realizar una actualización a las prioridades de las tareas del product-backlog del proyecto.	Jefes de proyecto	13/06/2013

César Gonzales Yapapasca
Jefe de Proyecto

José Torres Cárdenas
Jefe de Proyecto

Prof. Luis García Paucar
Cliente

Anexo 6: Acta de Reunión 06 – 13/08/2013

Acta de Reunión N°6

Sección 1. Información General

Nombre del Proyecto	Fecha de Reunión	Hora de Inicio	Lugar
Formalización de los procesos de documentación y evaluación de arquitecturas de software	13/08/2013	18:00	Oficina del cliente – Universidad Peruana de Ciencias Aplicadas
Asistentes	Cargo	Hora de llegada	
Cesar Gonzales Yapapasca	Jefe de proyecto	18:00	
José Torres Cárdenas	Jefe de proyecto	18:00	
Prof. Luis García Paucar	Cliente	18:00	

Sección 2. Temas tratados (agenda)

Tema:	Descripción:
Visita de la escuela al Software Engineering Institute en Estados Unidos.	Se conversó sobre la experiencia de la visita, los contactos obtenidos y como esto representa un importante avance para la universidad.
Aplicar los conocimientos adquiridos para proponer comentarios de corrección y mejora sobre los paquetes de despliegue de la norma técnica peruana ISO/IEC 29110 en lo que nuestra investigación comprende.	Se conversó de la importancia de realizar estas revisiones para identificar de qué manera se puede contribuir con este esfuerzo realizado por la universidad de mejorar las versiones de paquetes existentes y proponer mejoras. Esto mientras se establece lazos importantes entre la UPC y la ETS.

Sección 3. Acuerdos

Nº	Acuerdo:	Descripción:	Responsable	Fecha Límite
1	Revisar el paquete de despliegue de Arquitectura de Software y Diseño Detallado y proponer mejoras.	Se acordó revisar el paquete de despliegue de Arquitectura de Software y Diseño Detallado y proponer las mejoras según el conocimiento generado en el proyecto de investigación.	Jefes de proyecto	27/08/2013
2	Revisar el repositorio del proyecto open source ArchE y revisar la factibilidad de revisarlo y proponer mejoras.	Se acordó revisar el repositorio del proyecto open source ArchE y revisar la factibilidad de poder desplegar el proyecto, hacer pruebas e identificar las oportunidades de mejora correspondientes.	Jefes de proyecto	03/09/2013

César Gonzales Yapapasca
Jefe de Proyecto

José Torres Cárdenas
Jefe de Proyecto

Prof. Luis García Paucar
Cliente

Anexo 7: Acta de Reunión 07 – 27/08/2013

Acta de Reunión N°7

Sección 1. Información General

Nombre del Proyecto	Fecha de Reunión	Hora de Inicio	Lugar
Formalización de los procesos de documentación y evaluación de arquitecturas de software	27/08/2013	18:00	Oficina del cliente – Universidad Peruana de Ciencias Aplicadas
Asistentes	Cargo	Hora de llegada	
Cesar Gonzales Yapapasca	Jefe de proyecto	18:00	
José Torres Cárdenas	Jefe de proyecto	18:00	
Prof. Luis García Paucar	Cliente	18:00	

Sección 2. Temas tratados (agenda)

Tema:	Descripción:
Revisión de los avances producidos sobre las tareas pendientes con el cliente.	Se revisó lo avanzado como parte de las recomendaciones hacia el paquete de despliegue de Arquitectura de Software y Diseño Detallado y se detectó la necesidad de realizar también un análisis del paquete de despliegue de Análisis de Requerimientos de Software.

Sección 3. Acuerdos

Nº	Acuerdo:	Descripción:	Responsable	Fecha Límite
1	Realizar las correcciones pertinentes a lo avanzado de la revisión del paquete de despliegue de Arquitectura de Software y Diseño Detallado	Se hicieron algunas observaciones sobre lo generado como oportunidades de mejora del paquete de despliegue de Arquitectura de Software y Diseño Detallado y se acordó levantar las observaciones.	Jefes de proyecto	03/09/2013
2	Revisar el paquete de despliegue de Análisis de Requerimientos de Software.	Se acordó revisar también el paquete de despliegue de Análisis de Requerimientos de Software y brindar también una serie de oportunidades de mejora.	Jefes de proyecto	10/09/2013

César Gonzales Yapapasca
Jefe de Proyecto

José Torres Cárdenas
Jefe de Proyecto

Prof. Luis García Paucar
Cliente

Anexo 8: Acta de Reunión 08 – 10/09/2013

Acta de Reunión N°8

Sección 1. Información General

Nombre del Proyecto	Fecha de Reunión	Hora de Inicio	Lugar
Formalización de los procesos de documentación y evaluación de arquitecturas de software	10/09/2013	18:00	Oficina del cliente – Universidad Peruana de Ciencias Aplicadas
Asistentes	Cargo	Hora de llegada	
Cesar Gonzales Yapapasca	Jefe de proyecto	18:00	
José Torres Cárdenas	Jefe de proyecto	18:00	
Prof. Luis García Paucar	Cliente	18:00	

Sección 2. Temas tratados (agenda)

Tema:	Descripción:
Revisión de los avances con respecto al análisis del proyecto open source ArchE	Se conversó sobre lo avanzado en cuanto a la revisión del repositorio del proyecto open source ArchE.

Sección 3. Acuerdos

Nº	Acuerdo:	Descripción:	Responsable	Fecha Límite
1	Cambio de prioridad al análisis del proyecto open source ArchE	Luego de revisar los inconvenientes presentados por el equipo que impidieron realizar la tarea planteada, se determinó concentrar los esfuerzos en otras tareas más relevantes y cambiar la prioridad de esta tarea a una más baja.	Jefes de proyecto	-

César Gonzales Yapapasca
Jefe de Proyecto

José Torres Cárdenas
Jefe de Proyecto

Prof. Luis García Paucar
Cliente

Anexo 9: Acta de Reunión 09 – 17/09/2013

Acta de Reunión N°9

Sección 1. Información General

Nombre del Proyecto	Fecha de Reunión	Hora de Inicio	Lugar
Formalización de los procesos de documentación y evaluación de arquitecturas de software	17/09/2013	18:00	Oficina del cliente – Universidad Peruana de Ciencias Aplicadas
Asistentes	Cargo	Hora de llegada	
César Gonzales Yapapasca	Jefe de proyecto	18:00	
José Torres Cárdenas	Jefe de proyecto	18:00	
Prof. Luis García Paucar	Cliente	18:00	

Sección 2. Temas tratados (agenda)

Tema:	Descripción:
Llevar a cabo un taller de QAW	Se conversó sobre la importancia de generar evidencia del conocimiento generado, por lo que se conversó con el Gerente General y Arquitecto de BitPerfect para poder realizar un taller de Quality Attribute Workshop sobre un proyecto real de dicha empresa.

Sección 3. Acuerdos

Nº	Acuerdo:	Descripción:	Responsable	Fecha Límite
1	Realizar el taller de Quality Attribute Workshop	Se definió la fecha para llevar a cabo el taller de Quality Attribute Workshop.	Jefes de proyecto	19/10/13

César Gonzales Yapapasca
Jefe de Proyecto

José Torres Cárdenas
Jefe de Proyecto

Prof. Luis García Paucar
Cliente

Anexo 10: Acta de Reunión 10 – 24/09/2013

Acta de Reunión N°10

Sección 1. Información General

Nombre del Proyecto	Fecha de Reunión	Hora de Inicio	Lugar
Formalización de los procesos de documentación y evaluación de arquitecturas de software	24/09/2013	18:00	Oficina del cliente – Universidad Peruana de Ciencias Aplicadas
Asistentes	Cargo	Hora de llegada	
Cesar Gonzales Yapapasca	Jefe de proyecto	18:00	
José Torres Cárdenas	Jefe de proyecto	18:00	
Prof. Luis García Peucar	Cliente	18:00	

Sección 2. Temas tratados (agenda)

Tema:	Descripción:
Solucionar un ejercicio del Carnegie Mellon University proporcionado por el Doctor Felix Bachmann	Se conversó sobre un ejercicio práctico relacionado a la documentación de una arquitectura de software provisto por el Doctor Felix Bachmann.

Sección 3. Acuerdos

Nº	Acuerdo:	Descripción:	Responsable	Fecha Límite
1	Realizar un análisis sobre el ejercicio proporcionado del Carnegie Mellon University	Se definió la fecha para comenzar a ir generando un análisis sobre el ejercicio proporcionado del Carnegie Mellon University	Jefes de proyecto	08/10/13

César Gonzales Yapapasca
Jefe de Proyecto

José Torres Cárdenas
Jefe de Proyecto

Prof. Luis García Peucar
Cliente

Anexo 11: Informe de Seguimiento – Semana 9

Proyecto	Siglas
PS02 Formalización del proceso de Documentación de una Arquitectura de Software.	FPDAS
Jefes de Proyecto	Cesar Gonzales José Torres
Empresa Virtual	Bankmin
Profesor Gerente	Profesor Rubén Cerdá
Profesor Cliente	Profesor Luis García Paucar

PROFESOR GERENTE	PROFESOR CLIENTE
Se informó a los jefes del proyecto de realizar los informes completos de los papers seleccionados y presentarlos para que puedan ser enviados para su revisión. Los informes terminados y los papers originales seleccionados deben ser remitidos al Profesor Gerente.	
Fecha y Firma Jueves 10 de Octubre del 2013	Fecha y Firma

Anexo 12: Informe de Seguimiento – Semana 10

Proyecto	Siglas
PS02 Formalización del proceso de Documentación de una Arquitectura de Software.	FPDAS
Jefes de Proyecto	Cesar Gonzales José Torres
Empresa Virtual	Bankmin
Profesor Gerente	Profesor Rubén Cerdá
Profesor Cliente	Profesor Luis García Paucar

PROFESOR GERENTE	PROFESOR CLIENTE
Se informó a los jefes del proyecto a realizar el estado del arte para la 26 de Octubre para que pueda ser revisado por el Doctor Mauricio. Se presentó la posibilidad de presentar un avance durante la semana y poder ir dando retrospectivas antes de la fecha de entrega final.	
Fecha y Firma 17 de Octubre del 2013	Fecha y Firma

Anexo 13: Informe de Seguimiento – Semana 11

Proyecto	Siglas
PS02 Formalización del proceso de Documentación de una Arquitectura de Software.	FPDAS
Jefes de Proyecto	Cesar Gonzales José Torres
Empresa Virtual	Bankmin
Profesor Gerente	Profesor Rubén Cerdá
Profesor Cliente	Profesor Luis García Paucar

PROFESOR GERENTE	PROFESOR CLIENTE
	Se realizó el feedback respectivo sobre la plantilla del SAD generada inicialmente. Se acordó la entrega final para el 27 de Octubre
Fecha y Firma	Fecha y Firma 24 de Octubre del 2013

Anexo 14: Informe de Seguimiento – Semana 13

Proyecto	Siglas
PS02 Formalización del proceso de Documentación de una Arquitectura de Software.	FPDAS
Jefes de Proyecto	Cesar Gonzales José Torres
Empresa Virtual	Bankmin
Profesor Gerente	Profesor Rubén Cerdá
Profesor Cliente	Profesor Luis García Paucar

PROFESOR GERENTE	PROFESOR CLIENTE
	Revisión de los documentos generados a la fecha. Se propuso realizar algunos cambios a las plantillas generadas a la fecha para poder ser distribuidas con los interesados.
Fecha y Firma	Fecha y Firma 07 de Noviembre del 2013

Anexo 15: Informe de Seguimiento – Semana 14

Proyecto	Siglas
PS02 Formalización del proceso de Documentación de una Arquitectura de Software.	FPDAS
Jefes de Proyecto	Cesar Gonzales José Torres
Empresa Virtual	Bankmin
Profesor Gerente	Profesor Rubén Cerdá
Profesor Cliente	Profesor Luis García Paucar

PROFESOR GERENTE	PROFESOR CLIENTE
	Revisión del ejemplo de Documentación de Arquitectura de Software sobre el proyecto uReader. Las observaciones serán enviadas mediante correo para ser levantadas a la brevedad.
Fecha y Firma	Fecha y Firma 15 de Noviembre del 2013

Anexo 16: Reporte de monitoreo – Semana 1

Sección 1. Información General

Nombre del Proyecto	Fecha de Reporte
Formalización de procesos de documentación y evaluación de arquitecturas de software	24/03/2013
Membro del Equipo	
• Cesar Gonzales Yapapasca • Jose Torres Cardenas	

Sección 2. Cronograma

Item de proyecto	Reporte a la fecha	
Elaboración de Project Charter	Reunión con el cliente para sentar bases del proyecto.	19/03/2013
	Redacción del Project Charter sobre la plantilla de uso para Talleres de Proyecto.	24/03/2013
	Porcentaje Completado	70 %

Sección 3. Hitos

Hitos
• Elaboración de Project Charter

Hitos de Proyecto	Inicio Planeado	Inicio Real	Fin Planeado	Fin Real	Porcentaje Completado
Elaboración de Project Charter	18/03/2013		26/03/2013		70%

Sección 4. Riesgos

Riesgos del Proyecto
Demora en la comunicación con el cliente.

Factor de Riesgo	Mitigación del Riesgo
Disponibilidad del cliente.	Realizar un seguimiento al cliente.

Sección 5. Cambios del Proyecto

Cambios del Proyecto
Ninguno

Descripción del Cambio	Acciones ante el Cambio

Sección 6. Problemas del Proyecto

Problemas del Proyecto
Ninguno

Descripción del Problema	Acciones Correctivas

Anexo 17: Reporte de monitoreo – Semana 2

Sección 1. Información General

Nombre del Proyecto	Fecha de Reporte
Formalización de procesos de documentación y evaluación de arquitecturas de software	31/03/2013
Membro del Equipo	
<ul style="list-style-type: none">• Cesar Gonzales Yapapasca• Jose Torres Cardenas	

Sección 2. Cronograma

Item de proyecto	Reporte a la fecha	
Corrección del Project Charter	Se realizaron las modificaciones sobre el documento siguiendo las observaciones dadas por el profesor gerente.	100%
	Porcentaje Completado	100 %

Sección 3. Hitos

Hitos
<ul style="list-style-type: none">• Elaboración de Project Charter

Hitos de Proyecto	Inicio Planeado	Inicio Real	Fin Planeado	Fin Real	Porcentaje Completado
Elaboración de Project Charter	18/03/2013		26/03/2013		80%

Sección 4. Riesgos

Riesgos del Proyecto
Demora en la comunicación con el cliente. Demora en las revisiones del profesor gerente.

Factor de Riesgo	Mitigación del Riesgo
Disponibilidad del cliente.	Realizar un seguimiento al cliente.
Disponibilidad del profesor gerente.	Realizar un seguimiento al profesor gerente.

Sección 5. Cambios del Proyecto

Cambios del Proyecto
Observaciones sobre el contenido en el marco teórico.

Descripción del Cambio	Acciones ante el Cambio
Inclusión del propósito del proyecto.	Modificaciones del marco teórico del Project Charter.

Sección 6. Problemas del Proyecto

Problemas del Proyecto
Ninguno

Descripción del Problema	Acciones Correctivas

Anexo 18: Reporte de monitoreo – Semana 3

Sección 1. Información General

Nombre del Proyecto	Fecha de Reporte
Formalización de procesos de documentación y evaluación de arquitecturas de software	07/04/2013
Membro del Equipo	
<ul style="list-style-type: none">• Cesar Gonzales Yapapasca• Jose Torres Cardenas	

Sección 2. Cronograma

Item de proyecto	Reporte a la fecha	
Correcciones finales del Project Charter	Se realizaron las modificaciones sobre el documento siguiendo las observaciones dadas por el profesor gerente.	100%
	Porcentaje Completado	100 %

Sección 3. Hitos

Hitos
<ul style="list-style-type: none">• Elaboración de Project Charter

Hitos de Proyecto	Inicio Planeado	Inicio Real	Fin Planeado	Fin Real	Porcentaje Completado
Elaboración de Project Charter	18/03/2013		09/04/2013		100%

Sección 4. Riesgos

Riesgos del Proyecto

Demora en la comunicación con el cliente.
Demora en las revisiones del profesor gerente.

Factor de Riesgo	Mitigación del Riesgo
Disponibilidad del cliente.	Realizar un seguimiento al cliente.
Disponibilidad del profesor gerente.	Realizar un seguimiento al profesor gerente.

Sección 5. Cambios del Proyecto

Cambios del Proyecto

Observaciones sobre el contenido en el marco teórico.

Descripción del Cambio	Acciones ante el Cambio
Correcciones sobre las observaciones realizadas.	Modificaciones del marco teórico del Project Charter.

Sección 6. Problemas del Proyecto

Problemas del Proyecto

Ninguno

Descripción del Problema	Acciones Correctivas

Anexo 19: Reporte de monitoreo – Semana 4

Sección 1. Información General

Nombre del Proyecto	Fecha de Reporte
Formalización de procesos de documentación y evaluación de arquitecturas de software	14/03/2013
Membro del Equipo	
<ul style="list-style-type: none">• Cesar Gonzales Yapapasca• Jose Torres Cardenas	

Sección 2. Cronograma

Item de proyecto	Reporte a la fecha	Porcentaje Completado	Exposición del Project Charter
Exposición del Project Charter	Se realizará la presentación para la exposición a realizar frente al comité de evaluación	100%	Exposición del Project Charter
	Porcentaje Completado	100 %	

Sección 3. Hitos

Hitos
<ul style="list-style-type: none">• Elaboración de Project Charter

Hitos de Proyecto	Inicio Planeado	Inicio Real	Fin Planeado	Fin Real	Porcentaje Completado
Exposición del Project Charter	18/04/2013		18/04/2013		100%

Sección 4. Riesgos

Riesgos del Proyecto

Demora en la comunicación con el cliente.
Demora en las revisiones del profesor gerente.

Factor de Riesgo	Mitigación del Riesgo
Disponibilidad del cliente.	Realizar un seguimiento al cliente.
Disponibilidad del profesor gerente.	Realizar un seguimiento al profesor gerente.

Sección 5. Cambios del Proyecto

Cambios del Proyecto

Ninguno

Descripción del Cambio	Acciones ante el Cambio

Sección 6. Problemas del Proyecto

Problemas del Proyecto

Ninguno

Descripción del Problema	Acciones Correctivas

Anexo 20: Reporte de monitoreo – Semana 5

Sección 1. Información General

Nombre del Proyecto	Fecha de Reporte
Formalización de procesos de documentación y evaluación de arquitecturas de software	21/03/2013
Miembro del Equipo	
<ul style="list-style-type: none">• Cesar Gonzales Yapapasca• Jose Torres Cardenas	

Sección 2. Cronograma

Ítem de proyecto	Reporte a la fecha	
Diagramación y Caracterización de QAW	Se realizará la diagramación y caracterización de QAW.	100%
	Porcentaje Completado	100 %

Sección 3. Hitos

Hitos
<ul style="list-style-type: none">• Diagramación y Caracterización de QAW terminada

Hitos de Proyecto	Inicio Planeado	Inicio Real	Fin Planeado	Fin Real	Porcentaje Completado
<ul style="list-style-type: none">• Diagramación y Caracterización de QAW terminada	15/04/2013		21/04/2013		100%

Sección 4. Riesgos

Riesgos del Proyecto
Demora en la comunicación con el cliente. Demora en las revisiones del profesor gerente.

Factor de Riesgo	Mitigación del Riesgo
Disponibilidad del cliente.	Realizar un seguimiento al cliente.
Disponibilidad del profesor gerente.	Realizar un seguimiento al profesor gerente.

Sección 5. Cambios del Proyecto

Cambios del Proyecto

Descripción del Cambio	Acciones ante el Cambio

Sección 6. Problemas del Proyecto

Problemas del Proyecto
Ninguno

Descripción del Problema	Acciones Correctivas

Anexo 21: Reporte de monitoreo – Semana 6

Sección 1. Información General

Nombre del Proyecto	Fecha de Reporte
Formalización de procesos de documentación y evaluación de arquitecturas de software	28/03/2013
Membro del Equipo	
<ul style="list-style-type: none">• Cesar Gonzales Yapapasca• Jose Torres Cardenas	

Sección 2. Cronograma

Item de proyecto	Reporte a la fecha	
Diagramación y Caracterización de ADD	Se realizaró la diagramación y caracterización de ADD.	100%
	Porcentaje Completado	100 %

Sección 3. Hitos

Hitos
<ul style="list-style-type: none">• Diagramación y Caracterización de ADD terminada

Hitos de Proyecto	Inicio Planeado	Inicio Real	Fin Planeado	Fin Real	Porcentaje Completado
<ul style="list-style-type: none">• Diagramación y Caracterización de ADD terminada	22/04/2013		28/04/2013		100%

Sección 4. Riesgos

Riesgos del Proyecto
Demora en la comunicación con el cliente. Demora en las revisiones del profesor gerente.

Factor de Riesgo	Mitigación del Riesgo
Disponibilidad del cliente.	Realizar un seguimiento al cliente.
Disponibilidad del profesor gerente.	Realizar un seguimiento al profesor gerente.

Sección 5. Cambios del Proyecto

Cambios del Proyecto
Ninguno

Descripción del Cambio	Acciones ante el Cambio

Sección 6. Problemas del Proyecto

Problemas del Proyecto
Ninguno

Descripción del Problema	Acciones Correctivas

Anexo 22: Reporte de monitoreo – Semana 7

Sección 1. Información General

Nombre del Proyecto	Fecha de Reporte
Formalización de procesos de documentación y evaluación de arquitecturas de software	25/05/2013
Membro del Equipo	
<ul style="list-style-type: none">• Cesar Gonzales Yapapasca• Jose Torres Cardenas	

Sección 2. Cronograma

Item de proyecto	Reporte a la fecha	
Revisión de la Diagramación y Caracterización de ADD	Se realizaró la revisión de la diagramación y caracterización de ADD.	100%
	Porcentaje Completado	100 %

Sección 3. Hitos

Hitos
<ul style="list-style-type: none">• Revisión de la Diagramación y Caracterización de ADD terminada

Hitos de Proyecto	Inicio Planeado	Inicio Real	Fin Planeado	Fin Real	Porcentaje Completado
<ul style="list-style-type: none">• Revisión de la Diagramación y Caracterización de ADD terminada	29/04/2013		03/05/2013		100%

Sección 4. Riesgos

Riesgos del Proyecto
Demora en la comunicación con el cliente. Demora en las revisiones de los documentos.

Factor de Riesgo	Mitigación del Riesgo
Disponibilidad del cliente.	Realizar un seguimiento al cliente.
Disponibilidad del profesor gerente.	Realizar un seguimiento al profesor gerente.

Sección 5. Cambios del Proyecto

Cambios del Proyecto
Ninguno

Descripción del Cambio	Acciones ante el Cambio

Sección 6. Problemas del Proyecto

Problemas del Proyecto
Ninguno

Descripción del Problema	Acciones Correctivas

Anexo 23: Reporte de monitoreo – Semana 9

Sección 1. Información General

Nombre del Proyecto	Fecha de Reporte
Formalización de procesos de documentación y evaluación de arquitecturas de software	19/05/2013
Miembro del Equipo	
<ul style="list-style-type: none">• Cesar Gonzales Yapapasca• Jose Torres Cardenas	

Sección 2. Cronograma

Item de proyecto	Reporte a la fecha	
Corrección de la Diagramación y Caracterización de ADD	Levantamiento de las observaciones recibidas de QA	100%
	Porcentaje Completado	100 %

Sección 3. Hitos

Hitos
<ul style="list-style-type: none">• Corrección de la Diagramación y Caracterización de ADD

Hitos de Proyecto	Inicio Planeado	Inicio Real	Fin Planeado	Fin Real	Porcentaje Completado
<ul style="list-style-type: none">• Corrección de la Diagramación y Caracterización de ADD terminada	13/05/2013		17/05/2013		100%

Sección 4. Riesgos

Riesgos del Proyecto
Demora en la comunicación con el cliente. Demora en las revisiones del profesor gerente.

Factor de Riesgo	Mitigación del Riesgo
Disponibilidad del cliente.	Realizar un seguimiento al cliente.
Disponibilidad del profesor gerente.	Realizar un seguimiento al profesor gerente.

Sección 5. Cambios del Proyecto

Cambios del Proyecto
Ninguno

Descripción del Cambio	Acciones ante el Cambio

Sección 6. Problemas del Proyecto

Problemas del Proyecto
Ninguno

Descripción del Problema	Acciones Correctivas

Anexo 24: Reporte de monitoreo – Semana 10

Sección 1. Información General

Nombre del Proyecto	Fecha de Reporte
Formalización de procesos de documentación y evaluación de arquitecturas de software	26/03/2013
Membro del Equipo	
<ul style="list-style-type: none">• Cesar Gonzales Yapapasca• Jose Torres Cardenas	

Sección 2. Cronograma

Item de proyecto	Reporte a la fecha	
Ejemplificación de Escenarios - QAW	Se realizaron ejemplos basados en la plantilla generada para los procesos de Captura de Atributos de Calidad (QAW)	100%
	Porcentaje Completado	100 %

Sección 3. Hitos

Hitos
<ul style="list-style-type: none">• Ejemplificación de Escenarios - QAW

Hitos de Proyecto	Inicio Planeado	Inicio Real	Fin Planeado	Fin Real	Porcentaje Completado
<ul style="list-style-type: none">• Ejemplificación de Escenarios - QAW	20/05/2013		24/05/2013		100%

Sección 4. Riesgos

Riesgos del Proyecto

Demora en la comunicación con el cliente.
Demora en las revisiones del profesor gerente.

Factor de Riesgo	Mitigación del Riesgo
Disponibilidad del cliente.	Realizar un seguimiento al cliente.
Disponibilidad del profesor gerente.	Realizar un seguimiento al profesor gerente.

Sección 5. Cambios del Proyecto

Cambios del Proyecto

Ninguno

Descripción del Cambio	Acciones ante el Cambio

Sección 6. Problemas del Proyecto

Problemas del Proyecto

Ninguno

Descripción del Problema	Acciones Correctivas

Anexo 25: Reporte de monitoreo – Semana 1

Sección 1. Información General

Nombre del Proyecto	Fecha de Reporte
Formalización de procesos de documentación y evaluación de arquitecturas de software	18/08/2013
Membro del Equipo	
<ul style="list-style-type: none">• Cesar Gonzales Yapapasca• Jose Torres Cardenas	

Sección 2. Cronograma

Item de proyecto	Reporte a la fecha	
Documentación de Arquitecturas de Software	Actualización del status de los entregables con el cliente.	13/08/2013
Preparación del plan de trabajo para el ciclo académico 2013 02	Armar el plan de trabajo para el ciclo 2013 02	17/08/2013

Sección 3. Hitos

Hitos
<ul style="list-style-type: none">• Presentación del Plan de Trabajo para el ciclo académico 2013 02

Hitos de Proyecto	Inicio Planeado	Inicio Real	Fin Planeado	Fin Real	Porcentaje Completado
Presentación del Plan de Trabajo para el ciclo académico 2013 02	13/08/2013		17/08/2013		100%

Sección 4. Riesgos

Riesgos del Proyecto

Factor de Riesgo	Mitigación del Riesgo
Disponibilidad del cliente.	Realizar un seguimiento al cliente.
Disponibilidad del profesor gerente.	Realizar un seguimiento al profesor gerente.

Sección 5. Cambios del Proyecto

Cambios del Proyecto

Ninguno

Descripción del Cambio	Acciones ante el Cambio

Sección 6. Problemas del Proyecto

Problemas del Proyecto

Ninguno

Descripción del Problema	Acciones Correctivas

Anexo 26: Reporte de monitoreo – Semana 2

Sección 1. Información General

Nombre del Proyecto	Fecha de Reporte
Formalización de procesos de documentación y evaluación de arquitecturas de software	25/08/2013
Membro del Equipo	
<ul style="list-style-type: none">• Cesar Gonzales Yapapasca• Jose Torres Cardenas	

Sección 2. Cronograma

Item de proyecto	Reporte a la fecha	
Revisión de artefactos EPF del Doctor Chapagne	Se revisó los artefactos del Doctor Chapagne y se armó un plan de trabajo sobre los puntos a tratar.	70%
Revisión de proyecto open source Arch-E del Doctor Chapagne	Pendiente	

Sección 3. Hitos

Hitos
<ul style="list-style-type: none">• Entrega y aceptación de artefactos de Documentación de Arquitecturas de Software

Hitos de Proyecto	Inicio Planeado	Inicio Real	Fin Planeado	Fin Real	Porcentaje Completado
Entrega y aceptación de artefactos de Documentación de Arquitecturas de Software			02/09/2013		70%

Sección 4. Riesgos

Riesgos del Proyecto

Factor de Riesgo	Mitigación del Riesgo
Disponibilidad del cliente.	Realizar un seguimiento al cliente.
Disponibilidad del profesor gerente.	Realizar un seguimiento al profesor gerente.

Sección 5. Cambios del Proyecto

Cambios del Proyecto
Ninguno

Descripción del Cambio	Acciones ante el Cambio
Revisión de artefactos EPF del Doctor Chapagne	Se incorporó la tarea dentro del plan de trabajo.
Revisión de proyecto open source Arch-E del Doctor Chapagne	

Sección 6. Problemas del Proyecto

Problemas del Proyecto
Ninguno

Descripción del Problema	Acciones Correctivas
Revisión de artefactos generados ciclo académico 2013 01	10 horas

Anexo 27: Reporte de monitoreo – Semana 3

Sección 1. Información General

Nombre del Proyecto	Fecha de Reporte
Formalización de procesos de documentación y evaluación de arquitecturas de software	01/09/2013
Membro del Equipo	
<ul style="list-style-type: none">• Cesar Gonzales Yapapasca• Jose Torres Cardenas	

Sección 2. Cronograma

Item de proyecto	Reporte a la fecha	
Realización del feedback del documento EFP del Doctor Champagne	Se realizó el feedback correspondiente en Ingles sobre el documento original, señalando todas las modificaciones	100%
Pruebas y análisis del proyecto Arche-core del Doctor Champagne	Se revisó el proyecto en Github y luego de instalar las dependencias y preparar el entorno de desarrollo se determinó que no era viable la realización del feedback correspondiente	100%
Ejemplo de Software Architecture Documentation	Se continuó con el avance de la documentación de ejemplo.	80%

Sección 3. Hitos

Hitos
<ul style="list-style-type: none">• Entrega y aceptación de artefactos de Documentación de Arquitecturas de Software

Hitos de Proyecto	Inicio Planeado	Inicio Real	Fin Planeado	Fin Real	Porcentaje Completado
Entrega y aceptación de artefactos de Documentación de Arquitecturas de Software			02/09/2013		80%

Sección 4. Riesgos

Riesgos del Proyecto

Factor de Riesgo	Mitigación del Riesgo
Disponibilidad del cliente.	Realizar un seguimiento al cliente.
Disponibilidad del profesor gerente.	Realizar un seguimiento al profesor gerente.

Sección 5. Cambios del Proyecto

Cambios del Proyecto

Ninguno

Descripción del Cambio	Acciones ante el Cambio

Sección 6. Problemas del Proyecto

Problemas del Proyecto

Ninguno

Descripción del Problema	Acciones Correctivas

Anexo 28: Reporte de monitoreo – Semana 4

Sección 1. Información General

Nombre del Proyecto	Fecha de Reporte
Formalización de procesos de documentación y evaluación de arquitecturas de software	08/09/2013
Miembro del Equipo	
<ul style="list-style-type: none">• Cesar Gonzales Yapapasca• Jose Torres Cardenas	

Sección 2. Cronograma

Item de proyecto	Reporte a la fecha	Porcentaje
Realización del feedback del Deployment Software Design	Se realizó el feedback correspondiente en Ingles sobre el documento original, señalando todas las modificaciones. Además se agregó anexó un excel de cambios con respecto al original.	100%
Ejemplo de Software Architecture Documentation	Se continuó con el avance de la documentación de ejemplo.	90%

Sección 3. Hitos

Hitos
<ul style="list-style-type: none">• Entrega y aceptación de artefactos de Documentación de Arquitecturas de Software

Hitos de Proyecto	Inicio Planeado	Inicio Real	Fin Planeado	Fin Real	Porcentaje Completado
Entrega y aceptación de artefactos de Documentación de Arquitecturas de Software			12/09/2013		90%

Sección 4. Riesgos

Riesgos del Proyecto

Factor de Riesgo	Mitigación del Riesgo
Disponibilidad del cliente.	Realizar un seguimiento al cliente.
Disponibilidad del profesor gerente.	Realizar un seguimiento al profesor gerente.

Sección 5. Cambios del Proyecto

Cambios del Proyecto
Ninguno

Descripción del Cambio	Acciones ante el Cambio

Sección 6. Problemas del Proyecto

Problemas del Proyecto
Ninguno

Descripción del Problema	Acciones Correctivas

Anexo 29: Reporte de monitoreo – Semana 5

Sección 1. Información General

Nombre del Proyecto	Fecha de Reporte
Formalización de procesos de documentación y evaluación de arquitecturas de software	15/09/2013
Membro del Equipo	
<ul style="list-style-type: none">Cesar Gonzales YapapascaJose Torres Cárdenas	

Sección 2. Cronograma

Ítem de proyecto	Reportea la fecha	
Modificación del feedback del Software Design Deployment Package	Se realizó la modificación del feedback correspondiente en Ingles sobre el documento original, señalando todas las modificaciones.	100%
Realización del feedback del Requirement Analysis Deployment Package	Se realizó el feedback correspondiente en Ingles sobre el documento original, señalando todas las modificaciones. Además se agregó anexó un excel de cambios con respecto al original.	50%
Ejemplo de Software Architecture Documentation	Se continuó con el avance de la documentación de ejemplo.	100%

Sección 3. Hitos

Hitos
<ul style="list-style-type: none">Entrega y aceptación de artefactos de Documentación de Arquitecturas de Software

Hitos de Proyecto	Inicio Planeado	Inicio Real	Fin Planeado	Fin Real	Porcentaje Completado
Entrega y aceptación de artefactos de Documentación de Arquitecturas de Software			15/09/2013		100%

Sección 4. Riesgos

Riesgos del Proyecto

Factor de Riesgo	Mitigación del Riesgo
Disponibilidad del cliente.	Realizar un seguimiento al cliente.
Disponibilidad del profesor gerente.	Realizar un seguimiento al profesor gerente.

Sección 5. Cambios del Proyecto

Cambios del Proyecto
Ninguno

Descripción del Cambio	Acciones ante el Cambio

Sección 6. Problemas del Proyecto

Problemas del Proyecto
Ninguno

Descripción del Problema	Acciones Correctivas

Anexo 30: Reporte de monitoreo – Semana 6

Sección 1. Información General

Nombre del Proyecto	Fecha de Reporte
Formalización de procesos de documentación y evaluación de arquitecturas de software	22/09/2013
Miembro del Equipo	
<ul style="list-style-type: none">Cesar Gonzales YapapascaJose Torres Cardenas	

Sección 2. Cronograma

Item de proyecto	Reporte a la fecha	
Presentación de Quality Attribute Workshop	Se realizó la presentación de Quality Attribute Workshop utilizando un ejemplo real de software de la empresa Bitperfect.	100%

Sección 3. Hitos

Hitos
<ul style="list-style-type: none">Entrega y realización del Quality Attribute Workshop

Hitos de Proyecto	Inicio Planeado	Inicio Real	Fin Planeado	Fin Real	Porcentaje Completado
Entrega y realización del Quality Attribute Workshop			19/09/2013		100%

Sección 4. Riesgos

Riesgos del Proyecto

Factor de Riesgo	Mitigación del Riesgo
Disponibilidad del cliente.	Realizar un seguimiento al cliente.
Disponibilidad del profesor gerente.	Realizar un seguimiento al profesor gerente.

Sección 5. Cambios del Proyecto

Cambios del Proyecto
Ninguno

Descripción del Cambio	Acciones ante el Cambio

Sección 6. Problemas del Proyecto

Problemas del Proyecto
Ninguno

Descripción del Problema	Acciones Correctivas

Anexo 31: Reporte de monitoreo – Semana 7

Sección 1. Información General

Nombre del Proyecto	Fecha de Reporte
Formalización de procesos de documentación y evaluación de arquitecturas de software	29/09/2013
Miembro del Equipo	
<ul style="list-style-type: none">• Cesar Gonzales Yapapasca• Jose Torres Cardenas	

Sección 2. Cronograma

Item de proyecto	Reporte a la fecha	
Modificación del feedback del Requirement Analysis Deployment	Se realizó el feedback correspondiente en Ingles sobre el documento original, señalando todas las modificaciones. Además se agregó anexó un excel de cambios con respecto al original.	100%
Envio de las sugerencias al profesor Champagne	Se realizó el envío de las sugerencias de los cambios en los Packages.	100%

Sección 3. Hitos

Hitos

Hitos de Proyecto	Inicio Planeado	Inicio Real	Fin Planeado	Fin Real	Porcentaje Completado
Entrega y aceptación de todas las sugerencias de cambios de los Packages para el profesor Champagne			27/09/2013		100%

Sección 4. Riesgos

Riesgos del Proyecto

Factor de Riesgo	Mitigación del Riesgo
Disponibilidad del cliente.	Realizar un seguimiento al cliente.
Disponibilidad del profesor gerente.	Realizar un seguimiento al profesor gerente.

Sección 5. Cambios del Proyecto

Cambios del Proyecto

Ninguno

Descripción del Cambio	Acciones ante el Cambio

Sección 6. Problemas del Proyecto

Problemas del Proyecto

Ninguno

Descripción del Problema	Acciones Correctivas

Anexo 32: Reporte de monitoreo – Semana 9

Sección 1. Información General

Nombre del Proyecto	Fecha de Reporte
Formalización de procesos de documentación y evaluación de arquitecturas de software	13/10/2013
Membro del Equipo	
<ul style="list-style-type: none">• Cesar Gonzales Yapapasca• Jose Torres Cardenas	

Sección 2. Cronograma

Item de proyecto	Reporte a la fecha	
Informes de Papers	Se realizó el informe de los 3 papers seleccionados.	100%
Memoria Parcial	Se realizó la actualización de la memoria parcial.	100%

Sección 3. Hitos

Hitos

Hitos de Proyecto	Inicio Planeado	Inicio Real	Fin Planeado	Fin Real	Porcentaje Completado
Entrega de Informes de Papers			10/10/2013		100%
Entrega de Memoria Parcial TP2			10/10/2013		100%

Sección 4. Riesgos

Riesgos del Proyecto

Factor de Riesgo	Mitigación del Riesgo
Disponibilidad del cliente.	Realizar un seguimiento al cliente.
Disponibilidad del profesor gerente.	Realizar un seguimiento al profesor gerente.

Sección 5. Cambios del Proyecto

Cambios del Proyecto

Ninguno

Descripción del Cambio	Acciones ante el Cambio

Sección 6. Problemas del Proyecto

Problemas del Proyecto

Ninguno

Descripción del Problema	Acciones Correctivas

Anexo 33: Reporte de monitoreo – Semana 10

Sección 1. Información General

Nombre del Proyecto	Fecha de Reporte
Formalización de procesos de documentación y evaluación de arquitecturas de software	20/10/2013
Miembro del Equipo	
<ul style="list-style-type: none">Cesar Gonzales YapapascaJose Torres Cardenas	

Sección 2. Cronograma

Ítem de proyecto	Reporte a la fecha	
Plantilla SAD	Se creó una plantilla para la documentación de una arquitectura de software.	50%
Estado del Arte	Se creó el documento del estado del arte	50%

Sección 3. Hitos

Hitos

Hitos de Proyecto	Inicio Planeado	Inicio Real	Fin Planeado	Fin Real	Porcentaje Completado
Plantilla SAD	15/10/2013		27/10/2013		50%
Estado del Arte	15/10/2013		27/10/2013		50%

Sección 4. Riesgos

Riesgos del Proyecto

Factor de Riesgo	Mitigación del Riesgo
Disponibilidad del cliente.	Realizar un seguimiento al cliente.
Disponibilidad del profesor gerente.	Realizar un seguimiento al profesor gerente.

Sección 5. Cambios del Proyecto

Cambios del Proyecto
Ninguno

Descripción del Cambio	Acciones ante el Cambio

Sección 6. Problemas del Proyecto

Problemas del Proyecto
Ninguno

Descripción del Problema	Acciones Correctivas

Anexo 34: Reporte de monitoreo – Semana 11

Sección 1. Información General

Nombre del Proyecto	Fecha de Reporte
Formalización de procesos de documentación y evaluación de arquitecturas de software	27/10/2013
Miembro del Equipo	
<ul style="list-style-type: none">• Cesar Gonzales Yapasasca• Jose Torres Cardenas	

Sección 2. Cronograma

Item de proyecto	Reporte a la fecha	
Plantilla SAD	Se creó una plantilla para la documentación de una arquitectura de software.	100%
Estado del Arte	Se creó el documento del estado del arte	100%

Sección 3. Hitos

Hitos

Hitos de Proyecto	Inicio Planeado	Inicio Real	Fin Planeado	Fin Real	Porcentaje Completado
Plantilla SAD	15/10/2013		27/10/2013		100%
Estado del Arte	15/10/2013		27/10/2013		100%

Sección 4. Riesgos

Riesgos del Proyecto

Factor de Riesgo	Mitigación del Riesgo
Disponibilidad del cliente.	Realizar un seguimiento al cliente.
Disponibilidad del profesor gerente.	Realizar un seguimiento al profesor gerente.

Sección 5. Cambios del Proyecto

Cambios del Proyecto
Ninguno

Descripción del Cambio	Acciones ante el Cambio

Sección 6. Problemas del Proyecto

Problemas del Proyecto
Ninguno

Descripción del Problema	Acciones Correctivas

Anexo 35: Reporte de monitoreo – Semana 12

Sección 1. Información General

Nombre del Proyecto	Fecha de Reporte
Formalización de procesos de documentación y evaluación de arquitecturas de software	03/11/2013
Membro del Equipo	
<ul style="list-style-type: none">• Cesar Gonzales Yapapasca• Jose Torres Cardenas	

Sección 2. Cronograma

Item de proyecto	Reporte a la fecha	
Congreso Internacional	Se realizaron las actividades de asistencias para los ponentes y se realizó una pequeña presentación de lo alcanzado a la fecha hacia los especialistas del SEI.	100%

Sección 3. Hitos

Hitos

Hitos de Proyecto	Inicio Planeado	Inicio Real	Fin Planeado	Fin Real	Porcentaje Completado
Congreso Internacional	29/10/2013		31/10/2013		100%

Sección 4. Riesgos

Riesgos del Proyecto

Factor de Riesgo	Mitigación del Riesgo
Disponibilidad del cliente.	Realizar un seguimiento al cliente.
Disponibilidad del profesor gerente.	Realizar un seguimiento al profesor gerente.

Sección 5. Cambios del Proyecto

Cambios del Proyecto

Ninguno

Descripción del Cambio	Acciones ante el Cambio

Sección 6. Problemas del Proyecto

Problemas del Proyecto

Ninguno

Descripción del Problema	Acciones Correctivas

Anexo 36: Reporte de monitoreo – Semana 13

Sección 1. Información General

Nombre del Proyecto	Fecha de Reporte
Formalización de procesos de documentación y evaluación de arquitecturas de software	10/11/2013
Membro del Equipo	
<ul style="list-style-type: none">• Cesar Gonzales Yapapasca• Jose Torres Cardenas	

Sección 2. Cronograma

Item de proyecto	Reporte a la fecha	
Ejemplo SAD	Se avanzó con el ejemplo de Documentación de una Arquitectura de Software. La aplicación en análisis fue uReader.	50%

Sección 3. Hitos

Hitos

Hitos de Proyecto	Inicio Planeado	Inicio Real	Fin Planeado	Fin Real	Porcentaje Completado
Ejemplo SAD	04/11/2013		17/11/2013		50%

Sección 4. Riesgos

Riesgos del Proyecto

Factor de Riesgo	Mitigación del Riesgo
Disponibilidad del cliente.	Realizar un seguimiento al cliente.
Disponibilidad del profesor gerente.	Realizar un seguimiento al profesor gerente.

Sección 5. Cambios del Proyecto

Cambios del Proyecto
Ninguno

Descripción del Cambio	Acciones ante el Cambio

Sección 6. Problemas del Proyecto

Problemas del Proyecto
Ninguno

Descripción del Problema	Acciones Correctivas

Anexo 37: Reporte de monitoreo – Semana 14

Sección 1. Información General

Nombre del Proyecto	Fecha de Reporte
Formalización de procesos de documentación y evaluación de arquitecturas de software	17/11/2013
Membro del Equipo	
<ul style="list-style-type: none">• Cesar Gonzales Yapapasca• Jose Torres Cardenas	

Sección 2. Cronograma

Item de proyecto	Reporte a la fecha	
Ejemplo SAD	Se terminó con el ejemplo de Documentación de una Arquitectura de Software. La aplicación en análisis fue uReader.	100%

Sección 3. Hitos

Hitos

Hitos de Proyecto	Inicio Planeado	Inicio Real	Fin Planeado	Fin Real	Porcentaje Completado
Ejemplo SAD	04/11/2013		17/11/2013		100%

Sección 4. Riesgos

Riesgos del Proyecto

Factor de Riesgo	Mitigación del Riesgo
Disponibilidad del cliente.	Realizar un seguimiento al cliente.
Disponibilidad del profesor gerente.	Realizar un seguimiento al profesor gerente.

Sección 5. Cambios del Proyecto

Cambios del Proyecto
Ninguno

Descripción del Cambio	Acciones ante el Cambio

Sección 6. Problemas del Proyecto

Problemas del Proyecto
Ninguno

Descripción del Problema	Acciones Correctivas

Anexo 38: Certificado de Aprobación - QA



Anexo 39: Plantilla Anexo Análisis Enfoque Arquitectural

1 ESCENARIO

Escenario	<Código y texto del escenario.>			
Atributo(s)	<Atributo(s) de Calidad con el que este escenario está involucrado.>			
Ambiente	<Importantes suposiciones sobre el ambiente en el cual el sistema se aloja, y las condiciones relevantes en el cual el escenario toma lugar.>			
Estímulo	<Una declaración precisa del estímulo del atributo de calidad (i.e., función invocada, fallo, ejecución de un hilo, modificación,...) contextualizado en el escenario.>			
Respuesta	<Una declaración precisa de la respuesta del atributo de calidad (i.e., tiempo de respuesta, medición de dificultad o modificación).>			
Decisiones Arquitecturales	Sensibilidad	Trade Off	Riesgo	No Riesgo
<Decisiones arquitecturales relevantes con el escenario que afecten a la respuesta del atributo de calidad.>	<Punto de Sensibilidad>	<Punto de trade off>	<# Riesgo>	<# de No Riesgo>
...				
Razonamiento	<Racionalidad cualitativa y/o cuantitativa del por qué el listado de decisiones arquitecturales contribuyen a alcanzar cada requerimiento de atributo de calidad expresado en el escenario.>			

Diagrama arquitectural	<i><Diagrama o diagramas de las vistas arquitecturales comentadas con información arquitectural que soporte el razonamiento anterior, acompañado de texto que lo explique si se considera necesario.></i>
Listado de los Puntos de Sensibilidad	<i><Listar aquí los puntos de sensibilidad relacionados al escenario en análisis. Considera el código y el nombre.></i>
Listado de los Puntos de Trade off	<i><Listar aquí los puntos de trade off relacionados al escenario en análisis. Considera el código y el nombre.></i>
Listado de los Riesgos Identificados	<i><Listar aquí los riesgos identificados relacionados al escenario en análisis. Considera el código y el nombre.></i>
Listado de los No Riesgos Identificados	<i><Listar aquí los no riesgos identificados relacionados al escenario en análisis. Considera el código y el nombre.></i>

Considerar lo siguiente:

- *Punto de sensibilidad:* *<Es una propiedad de uno o más componentes (y/o relaciones de componentes) que es crítica para alcanzar la respuesta deseada de un atributo de calidad.>*
- *Punto de trade off:* *<Es una propiedad que afecta a más de un atributo y que es punto de sensibilidad para más de un atributo. Involucra las decisiones más críticas que se realizan en una arquitectura por las que se deben manejar con mucho cuidado.>*
- *Riesgos:* *<Expresan los potenciales problemas de las decisiones arquitecturales.>*
- *No Riesgos:* *<Buenas decisiones que se basan en suposiciones que frecuentemente se hallan detalladas en la arquitectura.>*

Anexo 40: Plantilla Documento Arquitectura Full

1 RESUMEN EJECUTIVO

<Se proporciona un breve resumen de los aspectos más importantes del documento, las decisiones que fueron tomadas y las conclusiones a las que se llegaron.>

1 INTRODUCCIÓN

11 Acrónimos, Abreviaturas y Siglas

<Se brindan definiciones de los acrónimos, abreviaturas o siglas de términos usados en el presente documento que necesiten de alguna explicación para su correcta interpretación.>

Siglas / Abreviaturas / Acrónimos	Significado
<ABCDF>	<Breve definición>

12 Definiciones

<Se brindan las definiciones de los términos usados en el documento, como términos genéricos o específicos (por ejemplo, términos relacionados al negocio y/o la organización a la cual pertenece el documento). Es importante también señalar la fuente de donde se obtiene la definición de los términos.>

13 Resumen del Documento

<Se realiza una descripción de las principales partes que conforman este documento, principalmente las vistas en las que ha sido estructurada y presentada la arquitectura del sistema. Además se señala para que stakeholder/interesado está dirigido cada vista arquitectural.>

Vista Arquitectural	Descripción	Interesados
<Nombre de la Vista>	<Breve descripción de la vista, la cual será detallada en los puntos siguientes>	<Lista de interesados a quienes está dirigida la vista>

1 VISIÓN GENERAL DEL SISTEMA

11 Descripción

<Se describe brevemente el sistema a analizar para tener un bosquejo general del alcance del mismo y se describe el contexto en cual se desarrolló el proyecto. Se explica también el contexto general donde se haya situado el sistema y se recomienda acompañarlo con un diagrama de contexto.>

<Recordar que un diagrama de contexto típicamente muestra la parte del sistema como un cuadro único y distingible en medio de otros cuadros que representan las entidades externas. Entre ellos se visualizan líneas que muestran las relaciones entre las partes del sistema y las entidades externas.>

12 Funcionalidades

<Se listan los requerimientos funcionales principales del sistema y se describen brevemente cada uno>

13 Requerimientos de Atributos de Calidad

<Se describen uno a uno los escenarios agrupados por Atributos de Calidad. Estos escenarios son los generados en el QAW (Quality Attribute Workshop).>

- <Atributo de Calidad (Performance / Seguridad / Confiabilidad / etc.) >
QAS #1: <Descripción del escenario de atributo de calidad>
QAS #2: <Descripción del escenario de atributo de calidad>
- <Atributo de Calidad (Performance / Seguridad / Confiabilidad / etc.) >
QAS #3: <Descripción del escenario de atributo de calidad>
QAS #4: <Descripción del escenario de atributo de calidad>

14 Restricciones

<Listar todas las restricciones del sistemas tales como las limitaciones o fronteras bajo las cuales el sistema debe ser creado. Estas restricciones también son llamadas restricciones de diseño. Pueden incluir hardware y/o software a usar, equipo de trabajo y/o lenguajes predeterminados>

1 VISTA DE MÓDULOS

<Los estilos de vistas modulares más conocidos son:

- *Descomposición*: es usado para mostrar la estructura de módulos y submódulos (incluyendo las relaciones entre estos).
- *De Uso*: es usado para mostrar las relaciones de dependencia entre los módulos.
- *Generalización*: es usado para indicar las relaciones de especialización entre los módulos.
- *Por Capas*: es usado para describir las relaciones de "permiso de uso" de una forma restringida entre grupos de módulos llamados capas.
- *Por Aspectos*: es usado para describir módulos particulares llamados "aspectos" que son responsables de preocupaciones transversales.
- *Modelo de Datos*: es usado para mostrar las relaciones entre las entidades de datos.>

11 Nombre de la Vista

111 Representación Básica

<Agregar aquí diagramas (o representaciones no gráficas) que muestren los elementos y sus relaciones en esta vista. Indicar el lenguaje o la notación empleada. Siempre es necesario agregar una leyenda para cada diagrama mostrado.>

112 Catálogo de Elementos

<Esta sección puede ser organizada como un diccionario donde cada entrada es un elemento de la Representación Básica. Para cada elemento se provee información adicional y las propiedades que los lectores necesitan conocer que no están expresados en la Representación Básica. Opcionalmente, se puede agregar especificaciones de interfaces y diagramas de comportamiento (por ejemplo, diagramas de secuencia UML, cuadros de estado).>

112.1 Elementos y sus propiedades

<Esta sección nombre cada elemento en la vista y lista sus propiedades. A manera de sugerencia, las propiedades mencionadas deben estar asociadas al estilo de vista en uso.>

1111 Relaciones y sus propiedades

<Esta sección tiene las relaciones que se presentan entre los elementos en la vista. Dichas relaciones suelen estar presentes en la Representación Básica. En caso existan relaciones no presentes en la Representación Básica o si existen excepciones de lo que se presenta se deben mencionar en esta sección. En caso de no existir relaciones esta sección va vacía.>

112 Guía de Variabilidad

<Describir aquí cualquier mecanismo de variabilidad usado en la parte del sistema mostrada en esta vista, acompañado del cómo y cuándo (tiempo de construcción, tiempo de despliegue, tiempo de ejecución) estos mecanismos serán empleados. Ejemplos de variabilidad incluyen: componentes opcionales (i.e., plug-ins, add-ons); réplicas configurables de componentes y conectores; selección de diferentes implementaciones de un elemento de diferentes proveedores; valores parametrizados en valores de construcción (build flags), archivos de propiedades (.properties), archivos de configuración inicial (.ini), u otros archivos de configuración.>

113 Racionalidad

<Describa aquí la racionalidad de cualquier decisión de diseño significativa cuyo alcance está limitado en esta vista. También describe cualquier alternativa de diseño representativa que haya sido rechazada. Esta sección también incluye suposiciones, restricciones, resultados de análisis y experimentos, y requerimientos arquitecturales significativos que afecten la vista.>

114 Vistas Relacionadas

<Agregar aquí las referencias a las vistas padre y a cualquier vista hijo (por ejemplo vistas refinadas) en caso existan.>

1 VISTAS DE COMPONENTES Y CONECTORES (C&C)

<Los estilos de vistas Componentes y Conectores más conocidos son:

- *Llamada-Retorno: estilo en el cual los componentes interactúan a través de invocaciones síncronas de las capacidades provistas por otros componentes.*
- *Flujo de Datos: estilo en el cual la computación es manejada por el flujo de datos a través del sistema.*
- *Basado en Eventos: estilo en el cual los componentes interactúan a través de eventos asíncronos o mensajes.*
- *Repositorios: estilo en el cual los componentes interactúan a través de grandes colecciones de data persistente y compartida.>*

11 Nombre de la Vista

111 Representación Básica

<Agregar aquí diagramas (o representaciones no gráficas) que muestren los elementos y sus relaciones en esta vista. Indicar el lenguaje o la notación empleada. Siempre es necesario agregar una leyenda para cada diagrama mostrado.>

112 Catálogo de Elementos

<Esta sección puede ser organizada como un diccionario donde cada entrada es un elemento de la Representación Básica. Para cada elemento se provee información adicional y las propiedades que los lectores necesitan conocer que no están expresados en la Representación Básica. Opcionalmente, se puede agregar especificaciones de interfaces y diagramas de comportamiento (por ejemplo, diagramas de secuencia UML, cuadros de estado).>

112.1 Elementos y sus propiedades

<Esta sección nombre cada elemento en la vista y lista sus propiedades. A manera de sugerencia, las propiedades mencionadas deben estar asociadas al estilo de vista en uso.>

1111 Relaciones y sus propiedades

<Esta sección tiene las relaciones que se presentan entre los elementos en la vista. Dichas relaciones suelen estar presentes en la Representación Básica. En caso existan relaciones no presentes en la Representación Básica o si existen excepciones de lo que se presenta se deben mencionar en esta sección. En caso de no existir relaciones esta sección va vacía.>

112 Guía de Variabilidad

<Describir aquí cualquier mecanismo de variabilidad usado en la parte del sistema mostrada en esta vista, acompañado del cómo y cuándo (tiempo de construcción, tiempo de despliegue, tiempo de ejecución) estos mecanismos serán empleados. Ejemplos de variabilidad incluyen: componentes opcionales (i.e., plug-ins, add-ons); réplicas configurables de componentes y conectores; selección de diferentes implementaciones de un elemento de diferentes proveedores; valores parametrizados en valores de construcción (build flags), archivos de propiedades (.properties), archivos de configuración inicial (.ini), u otros archivos de configuración.>

113 Racionalidad

<Describa aquí la racionalidad de cualquier decisión de diseño significativa cuyo alcance está limitado en esta vista. También describe cualquier alternativa de diseño representativa que haya sido rechazada. Esta sección también incluye suposiciones, restricciones, resultados de análisis y experimentos, y requerimientos arquitecturales significativos que afecten la vista.>

114 Vistas Relacionadas

<Agregar aquí las referencias a las vistas padre y a cualquier vista hijo (por ejemplo vistas refinadas) en caso existan.>

2 VISTAS DE ASIGNACIÓN

<Los estilos de vistas de Asignación más conocidos son:

- *Despliegue: describe el mapeo entre los componentes y conectores de software y el hardware de la plataforma de computación en el cual el software se ejecuta.*
- *Instalación: describe el mapeo entre los componentes de software y las estructuras en el sistema de archivos del ambiente de producción.*
- *Asignación de Trabajo: describe el mapeo entre los componentes de software y las personas, equipos o unidades de trabajo organizacionales asignados al desarrollo de estos módulos. >*

11 Nombre de la Vista

111 Representación Básica

<Agregar aquí diagramas (o representaciones no gráficas) que muestren los elementos y sus relaciones en esta vista. Indicar el lenguaje o la notación empleada. Siempre es necesario agregar una leyenda para cada diagrama mostrado.>

112 Catálogo de Elementos

<Esta sección puede ser organizada como un diccionario donde cada entrada es un elemento de la Representación Básica. Para cada elemento se provee información adicional y las propiedades que los lectores necesitan conocer que no están expresados en la Representación Básica. Opcionalmente, se puede agregar especificaciones de interfaces y diagramas de comportamiento (por ejemplo, diagramas de secuencia UML, cuadros de estado).>

112.1 Elementos y sus propiedades

<Esta sección nombre cada elemento en la vista y lista sus propiedades. A manera de sugerencia, las propiedades mencionadas deben estar asociadas al estilo de vista en uso.>

112.2 Relaciones y sus propiedades

<Esta sección tiene las relaciones que se presentan entre los elementos en la vista. Dichas relaciones suelen estar presentes en la Representación Básica. En caso existan relaciones no presentes en la Representación Básica o si existen excepciones de lo que se presenta se deben mencionar en esta sección. En caso de no existir relaciones esta sección va vacía.>

111 Guía de Variabilidad

<Describir aquí cualquier mecanismo de variabilidad usado en la parte del sistema mostrada en esta vista, acompañado del cómo y cuándo (tiempo de construcción, tiempo de despliegue, tiempo de ejecución) estos mecanismos serán empleados. Ejemplos de variabilidad incluyen: componentes opcionales (i.e., plug-ins, add-ons); réplicas configurables de componentes y conectores; selección de diferentes implementaciones de un elemento de diferentes proveedores; valores parametrizados en valores de construcción (build flags), archivos de propiedades (.properties), archivos de configuración inicial (.ini), u otros archivos de configuración.>

112 Racionalidad

<Describa aquí la racionalidad de cualquier decisión de diseño significativa cuyo alcance está limitado en esta vista. También describe cualquier alternativa de diseño representativa que haya sido rechazada. Esta sección también incluye suposiciones, restricciones, resultados de análisis y experimentos, y requerimientos arquitecturales significativos que afecten la vista.>

113 Vistas Relacionadas

<Agregar aquí las referencias a las vistas padre y a cualquier vista hijo (por ejemplo vistas refinadas) en caso existan.>

2 ANEXOS

<Se adjuntan todos los anexos que han sido referenciados en la documentación de arquitectura de software>

Anexo 41: Plantilla Documento Arquitectura Light

1 RESUMEN EJECUTIVO

<Se proporciona un breve resumen de los aspectos más importantes del documento, las decisiones que fueron tomadas y las conclusiones a las que se llegaron.>

1 INTRODUCCIÓN

1.1 Acrónimos, Abreviaturas y Siglas

<Se brindan definiciones de los acrónimos, abreviaturas o siglas de términos usados en el presente documento que necesiten de alguna explicación para su correcta interpretación.>

Siglas / Abreviaturas / Acrónimos	Significado
<i><ABCD></i>	<i><Breve definición></i>

1.2 Definiciones

<Se brindan las definiciones de los términos usados en el documento, como términos genéricos o específicos (por ejemplo, términos relacionados al negocio y/o la organización a la cual pertenece el documento). Es importante también señalar la fuente de donde se obtiene la definición de los términos.>

1.3 Resumen del Documento

<Se realiza una descripción de las principales partes que conforman este documento, principalmente las vistas en las que ha sido estructurada y presentada la arquitectura del sistema. Además se señala para qué stakeholder/interesado está dirigido cada vista arquitectural>

Vista Arquitectural	Descripción	Interesados
<i><Nombre de la Vista></i>	<i><Breve descripción de la vista, la cual será detallada en los puntos siguientes></i>	<i><Lista de interesados a quienes está dirigida la vista></i>

1 VISIÓN GENERAL DEL SISTEMA

11 Descripción

<Se describe brevemente el sistema a analizar para tener un bosquejo general del alcance del mismo y se describe el contexto en cual se desarrolló el proyecto. Se explica también el contexto general donde se haya situado el sistema y se recomienda acompañarlo con un diagrama de contexto.>

<Recordar que un diagrama de contexto típicamente muestra la parte del sistema como un cuadro único y distingible en medio de otros cuadros que representan las entidades externas. Entre ellos se visualizan líneas que muestran las relaciones entre las partes del sistema y las entidades externas.>

12 Funcionalidades

<Se listan los requerimientos funcionales principales del sistema y se describen brevemente cada uno.>

13 Requerimientos de Atributos de Calidad

<Se describen uno a uno los escenarios agrupados por Atributos de Calidad. Estos escenarios son los generados en el QAW (Quality Attribute Workshop).>

- <Atributo de Calidad (Performance / Seguridad / Confiabilidad / etc.) >
QAS #1: <Descripción del escenario de atributo de calidad>
QAS #2: <Descripción del escenario de atributo de calidad>
- <Atributo de Calidad (Performance / Seguridad / Confiabilidad / etc.) >
QAS #3: <Descripción del escenario de atributo de calidad>
QAS #4: <Descripción del escenario de atributo de calidad>

14 Restricciones

<Listar todas las restricciones del sistemas tales como las limitaciones o fronteras bajo las cuales el sistema debe ser creado. Estas restricciones también son llamadas restricciones de diseño. Pueden incluir hardware y/o software a usar, equipo de trabajo y/o lenguajes predeterminados>

1 VISTA DE MÓDULOS

<Los estilos de vistas modulares más conocidos son:

- *Descomposición*: es usado para mostrar la estructura de módulos y submódulos (incluyendo las relaciones entre estos).
- *De Uso*: es usado para mostrar las relaciones de dependencia entre los módulos.
- *Generalización*: es usado para indicar las relaciones de especialización entre los módulos.
- *Por Capas*: es usado para describir las relaciones de "permiso de uso" de una forma restringida entre grupos de módulos llamados capas.
- *Por Aspectos*: es usado para describir módulos particulares llamados "aspectos" que son responsables de preocupaciones transversales.
- *Modelo de Datos*: es usado para mostrar las relaciones entre las entidades de datos.>

11 Nombre de la Vista

111 Representación Básica

<Agregar aquí diagramas (o representaciones no gráficas) que muestren los elementos y sus relaciones en esta vista. Indicar el lenguaje o la notación empleada. Siempre es necesario agregar una leyenda para cada diagrama mostrado.>

112 Catálogo de Elementos

<Esta sección puede ser organizada como un diccionario donde cada entrada es un elemento de la Representación Básica. Para cada elemento se provee información adicional y las propiedades que los lectores necesitan conocer que no están expresados en la Representación Básica. Opcionalmente, se puede agregar especificaciones de interfaces y diagramas de comportamiento (i.e, diagramas de secuencia UML, cuadros de estado).>

112.1 Elementos y sus propiedades

<Esta sección nombre cada elemento en la vista y lista sus propiedades. A manera de sugerencia, las propiedades mencionadas deben estar asociadas al estilo de vista en uso.>

1111 Relaciones y sus propiedades

<Esta sección tiene las relaciones que se presentan entre los elementos en la vista. Dichas relaciones suelen estar presentes en la Representación Básica. En caso existan relaciones no presentes en la Representación Básica o si existen excepciones de lo que se presenta se deben mencionar en esta sección. En caso de no existir relaciones esta sección va vacía.>

112 Racionalidad

<Describa aquí la racionalidad de cualquier decisión de diseño significativa cuya alcance está limitado en esta vista. También describe cualquier alternativa de diseño representativa que haya sido rechazada. Esta sección también incluye suposiciones, restricciones, resultados de análisis y experimentos, y requerimientos arquitecturales significativos que afecten la vista.>

2 VISTAS DE COMPONENTES Y CONECTORES (C&C)

<Los estilos de vistas Componentes y Conectores más conocidos son:

- *Llamada-Retorno: estilo en el cual los componentes interactúan a través de invocaciones síncronas de las capacidades provistas por otros componentes.*
- *Flujo de Datos: estilo en el cual la computación es manejada por el flujo de datos a través del sistema.*
- *Basado en Eventos: estilo en el cual los componentes interactúan a través de eventos asíncronos o mensajes.*
- *Repositorios: estilo en el cual los componentes interactúan a través de grandes colecciones de data persistente y compartida.*>

2.1 Nombre de la Vista

2.11 Representación Básica

<Agregar aquí diagramas (o representaciones no gráficas) que muestren los elementos y sus relaciones en esta vista. Indicar el lenguaje o la notación empleada. Siempre es necesario agregar una leyenda para cada diagrama mostrado.>

111 Catálogo de Elementos

<Esta sección puede ser organizada como un diccionario donde cada entrada es un elemento de la Representación Básica. Para cada elemento se provee información adicional y las propiedades que los lectores necesitan conocer que no están expresados en la Representación Básica. Opcionalmente, se puede agregar especificaciones de interfaces y diagramas de comportamiento (i.e., diagramas de secuencia UML, cuadros de estado).>

1111 Elementos y sus propiedades

<Esta sección nombre cada elemento en la vista y lista sus propiedades. A manera de sugerencia, las propiedades mencionadas deben estar asociadas al estilo de vista en uso.>

1112 Relaciones y sus propiedades

<Esta sección tiene las relaciones que se presentan entre los elementos en la vista. Dichas relaciones suelen estar presentes en la Representación Básica. En caso existan relaciones no presentes en la Representación Básica o si existen excepciones de lo que se presenta se deben mencionar en esta sección. En caso de no existir relaciones esta sección va vacía.>

112 Racionalidad

<Describa aquí la racionalidad de cualquier decisión de diseño significativa cuya alcance está limitado en esta vista. También describe cualquier alternativa de diseño representativa que haya sido rechazada. Esta sección también incluye suposiciones, restricciones, resultados de análisis y experimentos, y requerimientos arquitecturales significativos que afecten la vista.>

1 VISTAS DE ASIGNACIÓN

<Los estilos de vistas de Asignación más conocidos son:

- Despliegue: describe el mapeo entre los componentes y conectores de software y el hardware de la plataforma de computación en el cual el software se ejecuta.
- Instalación: describe el mapeo entre los componentes de software y las estructuras en el sistema de archivos del ambiente de producción.
- Asignación de Trabajo: describe el mapeo entre los componentes de software y las personas, equipos o unidades de trabajo organizacionales asignados al desarrollo de estos módulos. >

11 Nombre de la Vista

111 Representación Básica

<Agregar aquí diagramas (o representaciones no gráficas) que muestren los elementos y sus relaciones en esta vista. Indicar el lenguaje o la notación empleada. Siempre es necesario agregar una leyenda para cada diagrama mostrado.>

112 Catálogo de Elementos

<Esta sección puede ser organizada como un diccionario donde cada entrada es un elemento de la Representación Básica. Para cada elemento se provee información adicional y las propiedades que los lectores necesitan conocer que no están expresados en la Representación Básica. Opcionalmente, se puede agregar especificaciones de interfaces y diagramas de comportamiento (i.e., diagramas de secuencia UML, cuadros de estado).>

112.1 Elementos y sus propiedades

<Esta sección nombre cada elemento en la vista y lista sus propiedades. A manera de sugerencia, las propiedades mencionadas deben estar asociadas al estilo de vista en uso.>

1111 Relaciones y sus propiedades

<Esta sección tiene las relaciones que se presentan entre los elementos en la vista. Dichas relaciones suelen estar presentes en la Representación Básica. En caso existan relaciones no presentes en la Representación Básica o si existen excepciones de lo que se presenta se deben mencionar en esta sección. En caso de no existir relaciones esta sección va vacía.>

112 Racionalidad

<Describa aquí la racionalidad de cualquier decisión de diseño significativa cuyo alcance está limitado en esta vista. También describe cualquier alternativa de diseño representativa que haya sido rechazada. Esta sección también incluye suposiciones, restricciones, resultados de análisis y experimentos, y requerimientos arquitecturales significativos que afecten la vista.>

2 ANEXOS

<Se adjuntan todos los anexos que han sido referenciados en la documentación de arquitectura de software>

Anexo 42: Ejemplo SAD Adventure Builder

1. Guía de documentación

1.1. Alcance y resumen

Este documento describe la arquitectura de la aplicación de referencia **Adventure Builder**, el cual es parte del programa BluePrints de Sun Microsystems. Adventure Builder es una aplicación de comercio electrónico (e-commerce) que ofrece paquetes de viaje para vacacionistas a través del internet. Emplea servicios web para interactuar con proveedores externos como bancos, aerolíneas, hoteles y otros proveedores.

La arquitectura descrita aquí fue recopilada de los artefactos de implementación de Adventure Builder, pero se hicieron algunos cambios para hacer la documentación de la arquitectura y del sistema en sí ser más interesantes para el propósito del documento.

1.2. ¿Cómo está organizado el documento?

El presente documento de arquitectura está organizado en las siguientes secciones:

- **Guía de la documentación:** provee información sobre el documento y la audiencia propósito. Provee una vista general de la estructura del documento.
- **Cómo es documentada una Vista:** describe cómo usar la plantilla para documentar una Vista de una arquitectura.
- **Visión general del sistema:** provee una descripción breve de la aplicación Adventure Builder, sus funcionalidades y requerimientos de atributos de calidad.
- **Vistas:** Varias vistas de arquitectura son descritas, cada una en secciones separadas. Cada Vista describe una estructura diferente del sistema. Existen vistas modulares, componente y conector (C&C), y vistas de asignación. Las Vistas son la parte más importante del documento. Información más detallada sobre cada Vista puede encontrarse en su respectiva sección.
- **Relación entre las Vistas:** Específica como los elementos en una Vista se relacionan con otras Vistas para pares específicos de Vistas.
- **Racionalidad:** Provee información sobre la arquitectura de software. Describe los antecedentes y rationalidad para una arquitectura de software. Explica las restricciones e influencias que llevaron a la arquitectura actual, y describe los mayores enfoques arquitecturales que fueron utilizados.
- **Directorio:** contiene un glosario de términos técnicos y acrónimos usados en la documentación que pueden no ser familiar para los lectores. También contiene un listado de los materiales referenciados.

Los lugares donde la documentación está incompleta están marcados con un "TODO". Si una sección está marcada como "N/A", significa que los contenidos de dicha sección no aplican o no son necesarios.

1. Cómo es documentada una Vista

Toda las Vistas de la arquitectura están descritas siguiendo la misma estructura definida en la plantilla de una Vista.

2. Visión general del sistema

2.1. Aplicación de referencia Adventure Builder

El sistema usado como ejemplo en este documento de arquitectura es una versión adaptada de la aplicación Adventure Builder, la cual fue desarrollada en el contexto del programa Java BluePrints en Sun Microsystems. Esta aplicación fue escogida porque su funcionalidad es sencilla de entender, es de código abierto (open source), y otros artefactos que están públicos para su descarga.

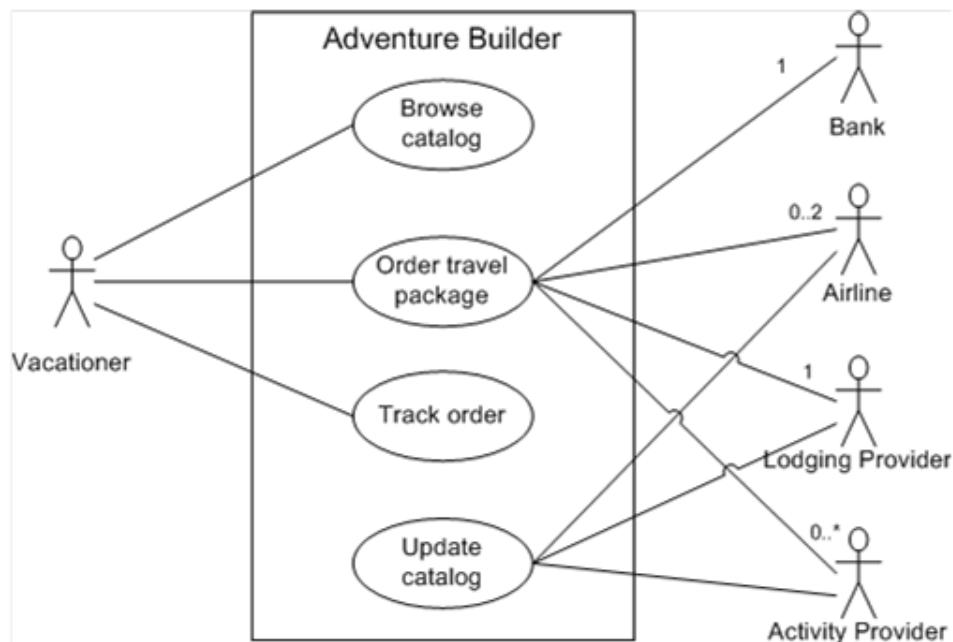
2.2. Funcionalidades

Adventure Builder es una empresa ficticia que vende paquetes de aventura para vacacionistas a través de internet. El sistema provee de cuatro casos de uso básico (UC):

- **UC1.** El usuario puede visitar la página web de Adventure Builder y navegar en el catálogo de paquetes de viaje, que incluye viajes a destinos específicos, opciones de alojamiento, y actividades que pueden ser compradas con anticipación. Actividades como ciclismo, pesca, clases de surf, globo aerostático y buzo. El usuario puede seleccionar el transporte, lugar, y varias actividades para construir su paquete de aventura de viaje.
- **UC2.** El usuario puede colocar una orden de compra de un paquete. Para procesar esta orden, el sistema tiene que interactuar con varias entidades externas. Así un banco requiere aprobar el pago del cliente, aerolíneas requieren proveer de la información de vuelos, lugares de alojamiento reservarán las habitaciones, y centros que proveen actividades agendarán las actividades seleccionadas por el usuario.

UC3. Luego de que la orden esté colocada, el usuario puede regresar para revisar el estado de su orden. Esto es necesario porque algunas interacciones con

- entidades externas son procesadas en segundo plano y pueden tomar horas o días en concretarse.
- UC4. El sistema interno periódicamente interactúa con los proveedores externos (transporte, hospedaje, centros de actividades) para actualizar el catálogo con las ofertas más recientes.



1.1. Requerimientos de Atributos de Calidad / Quality Attribute Requirements

Los escenarios de atributos de calidad están listados a continuación, separados según atributo de calidad.

- **Capacidad de modificación [Modifiability]**

- QAS1. Un nuevo socio del negocio (aerolínea, hospedaje, centro de actividades) que use su propio interface de servicios web es agregado al sistema en un esfuerzo no mayor de 10 días-persona para la implementación. El objetivo del negocio es tener una fácil integración con los socios del negocio.

- **Rendimiento [Performance]**

- QAS2. Un usuario coloca una orden para un paquete de aventura en el website consumidor. El usuario luego es notificado en la pantalla de que la orden ha sido enviada exitosamente y que está siendo procesada, esto en menos de 5 segundos.
- QAS3. Bajo 500 clicks de usuarios revisando el catálogo de paquetes de aventura seguida de una distribución de alrededor 1 minuto, el sistema está operando a condiciones normales. El máximo de latencia para mostrar la primera página de contenido debe ser menor a 5 segundos, la latencia promedio para lo mismo en menos de 2 segundos.

- **Confiabilidad [Reliability]**

- QAS4. El Website Consumidor envía la petición de la orden de compra al centro de procesamiento de órdenes (Order Processing Center). El OPC procesa la solicitud pero no responde al Website Consumidor en cinco segundos, entonces el Website Consumidor reenvía la petición al OPC. El OPC recibe la petición duplicada, pero el consumidor no es cargado dos veces, la data permanece en un estado consistente, y el Consumer Website es notificado que la petición original fue exitosa en un 100%.

- **Seguridad [Security]**

- QAS5. La aprobación del crédito y procesamiento de compras son solicitados para una nueva orden. En el 100% de los casos, la transacción es completada de manera segura y no puede ser rebotada por ninguna de las partes. Los objetivos de negocio son proveer a los clientes y socios la confianza en que la seguridad y de cumplir con las obligaciones contractuales, legales y regulatorias para mantener una transacción segura.
- QAS6. El OPC experimenta una lluvia de peticiones a través del servicio web que no corresponden a ninguna orden vigente. En el 100% de los casos, el sistema detecta la actividad anormal, notifica al administrador del sistema, y continúa con la petición del servicio en modo degradado.

- **Disponibilidad [Availability]**

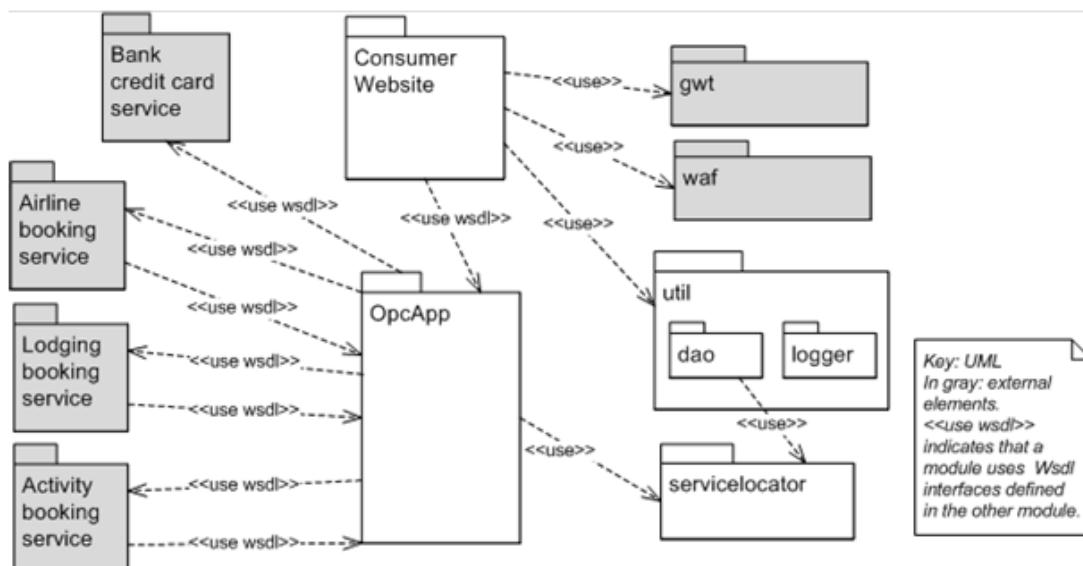
- QAS7. El Website Consumidor está disponible para el usuario 24x7. Si una instancia de la aplicación OPC falla, la falta es detectada y es notificada al administrador del sistema en 30 segundos. El sistema continúa tomando peticiones de órdenes, otra instancia del OPC es creada y la data permanece en un estado consistente.

1. Vistas

1.1. Vistas Modular / Module Views

1.1.1. Vista de Usos de alto nivel / Top Level Module Uses View

1.1.1.1. Presentación básica / Primary Presentation



1.1.1.1. Catálogo de elementos / Element Catalog

Elemento	Descripción
ConsumerWebsite	<p>La interfaz de usuario basada en web de Adventure Builder es implementado en este módulo. La interfaz de usuario permite que el usuario navegue el catálogo de paquetes de viajes, realice una nueva orden de compra y pueda rastrear el estado de las órdenes existentes. Este módulo crea las órdenes de compra basadas en las entradas del usuario y las transfiere a OpcApp para su procesamiento. Este usa una implementación del patrón Modelo Vista Controlador llamado Web Application Framework (waf). El modelo es implementado usando Entity beans, el controlador es implementado usando servlets, y la vista es una colección de JSPs y páginas estáticas de HTML. Parte del código del lado del cliente es implementado usando el framework GWT.</p>
OpcApp	<p>OpcApp es la Aplicación del Centro de Procesamiento de Órdenes. La lógica de negocio de Adventure Builder es implementado en este módulo. Esto provee las funcionalidades siguientes:</p> <ul style="list-style-type: none"> ▪ Aceptar las peticiones de orden de compra del ConsumerWebsite para que sean procesadas por el Servicio Web de Compra de Orden. ▪ Proveer un mecanismo para que la Página Web del Consumidor pueda consultar el estado actual de una orden de compra al Servicio Web de Rastreo de Órdenes. ▪ Comunicarse con proveedores externos para procesar y mantener el estado de una orden de compra. ▪ Tras la finalización del proceso de orden de compra, mandar un email al cliente sobre el éxito o falla del proceso. <p>Una vista refinada de descomposición de módulos y una vista refinada de usos de módulos están disponibles más adelante.</p>
utils	<p>Este módulo contiene utilidades usadas por el sistema Adventure Builder.</p>

dao	Este módulo contiene utilidades de Data Access Object (DAO), tales como fábrica DAO. Este no contiene las clases actuales DAO que acceden a la base de datos -- estas clases están dentro del módulo ConsumerWebsite.
logger	Este módulo contiene utilidades de rastreo y debug. El nombre original fue rastreador
servicelocator	Este módulo es una implementación del patrón de diseño ServiceLocator
gwt	Google Web Toolkit (gwt) (gwt) es un framework de código abierto para desarrollo de aplicaciones web basadas en Ajax.
waf	Waf son las siglas de Framework de Aplicaciones Web (en inglés). Este es un framework Model View Controller similar a Struts. Este permite especificar en archivos de configuración las pantallas web y acciones que están asociadas a los clicks del usuario sobre elementos específicos de la pantalla. El archivo de configuración tiene el mapeo de pantallas y acciones hacia clases de Java. El framework provee el motor para mostrar las pantallas apropiadas e invocar las acciones apropiadas.
Bank credit card service	Este módulo representa un servicio externo provisto por un Banco socio para validar las transacciones de tarjeta de crédito.
Airline booking service	Este módulo representa un servicio externo provisto por una aerolínea socia para reservar viajes aéreos.
Lodging booking service	Este módulo representa un servicio externo provisto por una compañía de alojamiento socia para la reserva de cuartos de hotel.
Activity booking service	Este módulo representa un servicio externo provisto por una compañía proveedora de actividades para reservar diferentes actividades.

1.1.1.1. Diagrama de contexto / Context Diagram

N/A

1.1.1.2. Guía de variabilidad / Variability Guide

Ver Guía de Variabilidad de OPC Module Uses View.

Ver Guía de Variabilidad de Top Level SOA View.

Ver Guía de Variabilidad de Consumer Website Multi-tier View.

1.1.1.3. Racionalidad / Rationale

WAF

El framework WAF fue escogido porque este facilita la implementación del código del Website Consumer proveyendo clases plantillas para el uso del patrón MVC. Para un usuario de operación dado, el desarrollador implementa una clase de acción (controlador) y páginas JSP que corresponden a las pantallas de usuario (Vistas). El desarrollador también usa archivos de configuración para proveer un mapeo configurable entre las acciones, clases de acciones, eventos y pantallas. La infraestructura de WAF puede luego tomar pedidos de HTTP e invocar a las clases de acciones y las pantallas JSP.

WAF también provee soporte para comunicación basada en eventos e internacionalización.

Una alternativa de WAF fue usar el Spring framework. Spring era una solución más robusta y rica desde un punto de vista técnica, pero este fue rechazado porque el equipo de desarrollo no está familiarizado con Spring y muy familiarizado con WAF.

GWT

El framework GWT fue elegido por las siguientes razones:

- Es código abierto, el cual nos permite ir debajo del capó y arreglar las cosas cuando se necesiten.
- Provee un ambiente de desarrollo rico con rastreo/debug poderoso, integración con IDE y administración de build.
- Se integra bien con otras tecnologías frontend debido a que no usa estándares propietarios.
- Ofrece funcionalidades poderosas para la construcción de widgets.
- Existen muchas librerías de extensión disponibles para componentes widget.
- Código es escrito en Java (y traducido a JavaScript durante el build), el cual es un lenguaje OO que es familiar para la mayoría del equipo.
- Código es compilado en Javascript, el cual está disponible en un porcentaje muy alto de navegadores.
- Tiene una comunidad de soporte razonablemente grande y es respaldado por un gran jugador en la industria (Google).

Esta tabla muestra la comparativa de análisis de GWT y tecnologías competentes basadas en necesidades específicas de nuestro proyecto.

	Flex	GWT	OpenLaszlo	Silverlight
Lenguaje de Desarrollo	ActionScript + MXML (Basado en ECMAScript 4)	Java + JavaScript	LZX / JavaScript	Any .net language + XAML
Ambiente de Desarrollo	Eclipse Flex Builder	Eclipse	(?)	Visual Studio
Code-build-run tiempo*	Bueno (~3s para una aplicación simple)	Bueno (~4s para una aplicación simple)	Muy lento (30s+ para recompilar una página de "Hola Mundo")	(?)
Eficiencia de Desarrollo	Bueno (declarative UI elements, data binding, visual designer)	Promedio (instancias procedurales detalladas)	Bueno (declarativo, data binding)	Bueno (lo mismo que Flex; Silverlight Blend se compara a Flash MX)
Herramientas de Debug	Eclipse debugger	Eclipse debugger	Integrated debugger (no se pudo probar)	Visual Studio Debugger
Frameworks de Pruebas Automatizadas	Flex automation testing API	JUnit / Selenium	(?)	NUnit

Gráficos / Animación	Poderoso motor de animación Flash	Promedio, puede ser extendido con scriptaculous, ext-js	Depende del tiempo de renderización (Flash/DHTML)	Ver Flex
Librerías de widget	Muy completo (debido a la madurez de la tecnología)	Muy completo (Librerías de extensión Javascript)	Promedio (inmaduro)	Lo mismo de Flash (librerías de terceros)
Comunidad de Desarrollo	Bien establecido	Bien establecido	(?)	Menor que Flex pero creciendo
Performance de Ejecución del Cliente*	Muy bueno (código nativo)	Promedio (interpretado, excepto en Chrome)	Un poco más lento que Flex (Capa Laszlo intermedia)	Muy bueno (código nativo)
Comunicación con Servidor	AMF, REST, SOAP	GWT-RPC, REST, SOAP, JSON	(?)	(?)
Soporte de Internacionalización	Promedio. Localización en tiempo de ejecución de Flex para caracteres latinos. No soporte nativo	GWT localización API provee una solución.	Debajo del promedio. Solo lenguajes europeos acentuados.	Pobre, mismo que flex + falta de fonts.

Portabilidad	Requiere Flash player	Requiere Javascript, no hay necesidad de plugin para los navegadores principales	Tiempo de ejecución agnóstico, actualmente soporta DHTML (proyectos futuros para Java ME, iPhone, Silverlight?)	Requiere Silverlight plugin (no soporte para Linux)
Soporte Móvil	Gran número de plataformas (todo Windows mobile, mayoría de Symbian, 50+ teléfonos celulares)	Muy limitado	Depende del tiempo de ejecución (Flex o GWT)	Soportado por 1.0, Windows mobile y Nokia S60

*Todos las pruebas de performance fueron ejecutadas en el mismo hardware: Pentium M 1.7GHz, 2GB RAM, Win XP Pro SP2

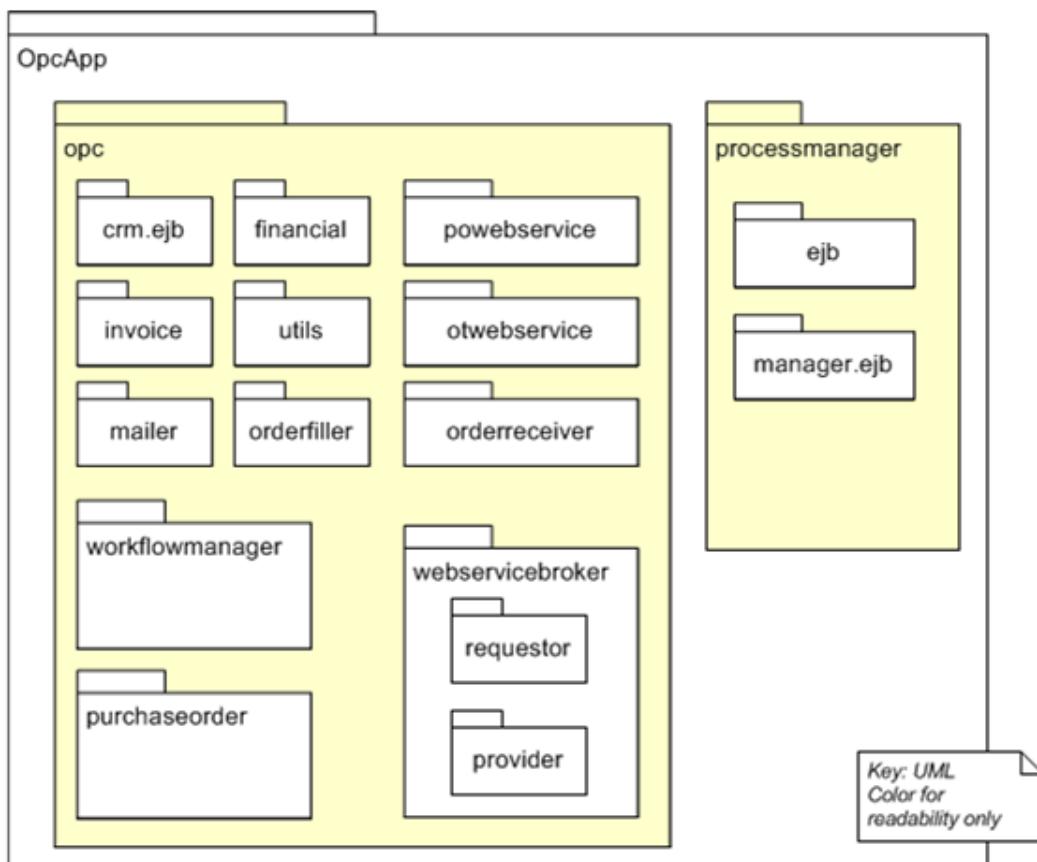
1.1.1.1. Vistas relacionadas / Related Views

Refinamiento de OpcApp mostrando descomposición: OPC Module Decomposition View

Refinamiento de OpcApp mostrando uso de dependencias: OPC Module Uses View

1.1.2. Vista de Descomposición del módulo OPC / OPC Module Decomposition View

1.1.2.1. Presentación básica / Primary Presentation



1.1.1.1. Catálogo de elementos / Element Catalog

Elemento	Descripción
OpcApp	<p>OpcApp es la Aplicación del Centro de Procesamiento de Órdenes. La lógica de negocio de Adventure Builder es implementado en este módulo. Esto provee las funcionalidades siguientes:</p> <ul style="list-style-type: none"> • Aceptar las peticiones de orden de compra del ConsumerWebsite para que sean procesadas por el Servicio Web de Compra de Orden. • Proveer un mecanismo para que la Página Web del Consumidor pueda consultar el estado actual de una orden de compra al Servicio Web de Rastreo de Órdenes. • Comunicarse con proveedores externos para procesar y mantener el estado de una orden de compra. • Tras la finalización del proceso de orden de compra, mandar un email al cliente sobre el éxito o falla del proceso. <p>Una vista refinada de descomposición de módulos y una vista refinada de usos de módulos están disponibles más adelante.</p>
opc	<p>Este paquete contiene toda la lógica de procesamiento de las órdenes, incluyendo flujo de trabajo, colas internas usadas para la comunicación entre los elementos e interacción con servicios web externos.</p>
crm.ejb	<p>Este es Administrador basado en la relación con los clientes (CRM en inglés). El trabajo de este módulo es enviar un correo una vez que una orden haya sido completamente y satisfactoriamente procesada. Este es implementado como un bean basado en mensajes. En el futuro este módulo puede albergar información adicional acerca de los clientes, lo cual podría asistir en proveer a los clientes de una mejor experiencia. Este podría incluir cosas como el historial de compras de un cliente en particular o enviar correos periódicamente acerca de nuevas ofertas.</p>

factura	Este paquete contiene una estructura de datos que alberga información que OPC usa para comunicarse con proveedores externos. La estructura de datos también contiene el estado de una orden en la factura (para mayor información del ciclo de vida de una orden ver el diagrama de estados en el OPC C&C view).
remitente	El remitente es un módulo de ayuda y su responsabilidad primaria es enviar correos usando el API de Java Mail. Este es provisto de un mensaje y direcciones de correos para enviar los correos. En el futuro este módulo será movido al paquete utils fuera de opc.
financiero	El módulo financiero es responsable por verificar y cobrar a la tarjeta de crédito del cliente. Para este propósito, este llama a los servicios web externos provistos por el banco. La verificación de la tarjeta de crédito pasa de una manera síncrona y la aplicación OPC espera a que el servicio web externo responda para proseguir. Si la respuesta del servicio de banca no es positiva, la aplicación OPC no procesará la orden.
utils	Este paquete contiene clases de utilidad que son usadas por la aplicación OPC, tal como código de unión para el API JMS. En el futuro, este módulo será movido al paquete útil fuera de OPC.
orderfiller	Este módulo lee una cola interna de peticiones de órdenes. Cuando una orden arriba, este descompone la orden en peticiones a los diferentes proveedores envueltos. Estas peticiones son enviadas en formato XML a las colas internas.
powebbservice	Este módulo provee un servicio web que es usado por ConsumerWebsite para comunicar los detalles de una orden de compra a OPC para procesarla. La interfaz del servicio web es OpcPurchaseOrderService: <ul style="list-style-type: none"> • Documentación de Interfaz de OpcPurchaseOrderService

otwebservice	<p>Este módulo provee un servicio web que es usado por ConsumerWebsite para preguntar el estado de una orden proveyendo el id del orden. La interfaz del servicio web es OpcOrderTrackingService:</p> <ul style="list-style-type: none"> • Documentación de Interfaz de OpcOrderTrackingService
orderreceiver	<p>El orderreceiver ayuda en la persistencia de la orden de compra en base de datos relacional.</p>
administrador de flujo de trabajo	<p>El administrador de flujo de trabajo coordina el procesamiento de una orden de compra y rastrea el estado de una orden de compra a través de su ciclo de vida. El administrador de flujo de trabajo media la interacción a lo largo de los otros módulos internos del OPC de tal forma que estos no saben de la existencia de cada uno de los otros.</p> <p>Para mayor detalle ver el administrador de flujo de trabajo de Module Uses View</p>
purchaseorder	<p>Este paquete contiene las clases que corresponden en memoria a las entidades de data requeridas para crear una orden de compra. Por cada entidad de data hay un POJO y una entidad bean. Los POJOs son usados a través de la aplicación como objetos de transferencias de datos. Las entidades de data en este paquete son:</p> <ul style="list-style-type: none"> • Actividad • Tarjeta de Crédito • Alojamiento • Orden de Compra • Transporte • Información de Contacto • Dirección <p>Ver la data Model para una descripción de lo que cada entidad representa.</p>

webservicebroker	El webservicebroker es responsable de la interacción vía servicios web con la aerolínea, transporte y proveedores de actividades. Este módulo es dividido en 2 submódulos descritos abajo.
solicitante	Contiene clases que pueden invocar servicios web externos provistos por los socios de aerolínea, transporte y proveedores de actividades. Este también contiene un bean basado en mensajes que puede recibir mensajes enviados a través de una cola interna. Estos mensajes contienen exactamente las peticiones a ser enviados a los servicios web externos.
proveedor	Este módulo provee un servicio web que es visible para los socios de aerolínea, transporte y proveedores de actividades. Este servicio web es implementado en un bean de sesión y es usado por socios externos para enviar los resultados del procesamiento de solicitudes procesados por estos. Las llamadas a este servicio web son reenviadas a una cola interna.
administrador de procesos	El administrador de procesos es usado por el módulo otwebserice para traer de la base datos las órdenes de compra y sus estados actualizados. Esto es también usado por el administrador de flujo de trabajo para traer las órdenes colocadas con el proveedor externo y persistir sus estados. Este módulo contiene los submódulos (ejb y administrador.ejb) descritos luego.
ejb	Contiene una sesión bean que ofrece operaciones para traer las órdenes y actualizar el estado de una orden dada (ambas órdenes de compra adventure y las órdenes colocadas con los socios proveedores).
administrador.ejb	Contiene un bean entity para persistir una orden de compra. La entidad bean usa persistencia administrada por contenedor (CMP en

1.1.1.1. [Diagrama de contexto / Context Diagram](#)

N/A

1.1.1.2. [Guia de variabilidad / Variability Guide](#)

N/A

1.1.1.3. [Racionalidad / Rationale](#)

La elección de EJBs en la implementación, incluyendo sesiones beans, beans basado en mensajes y entidades bean está basado en:

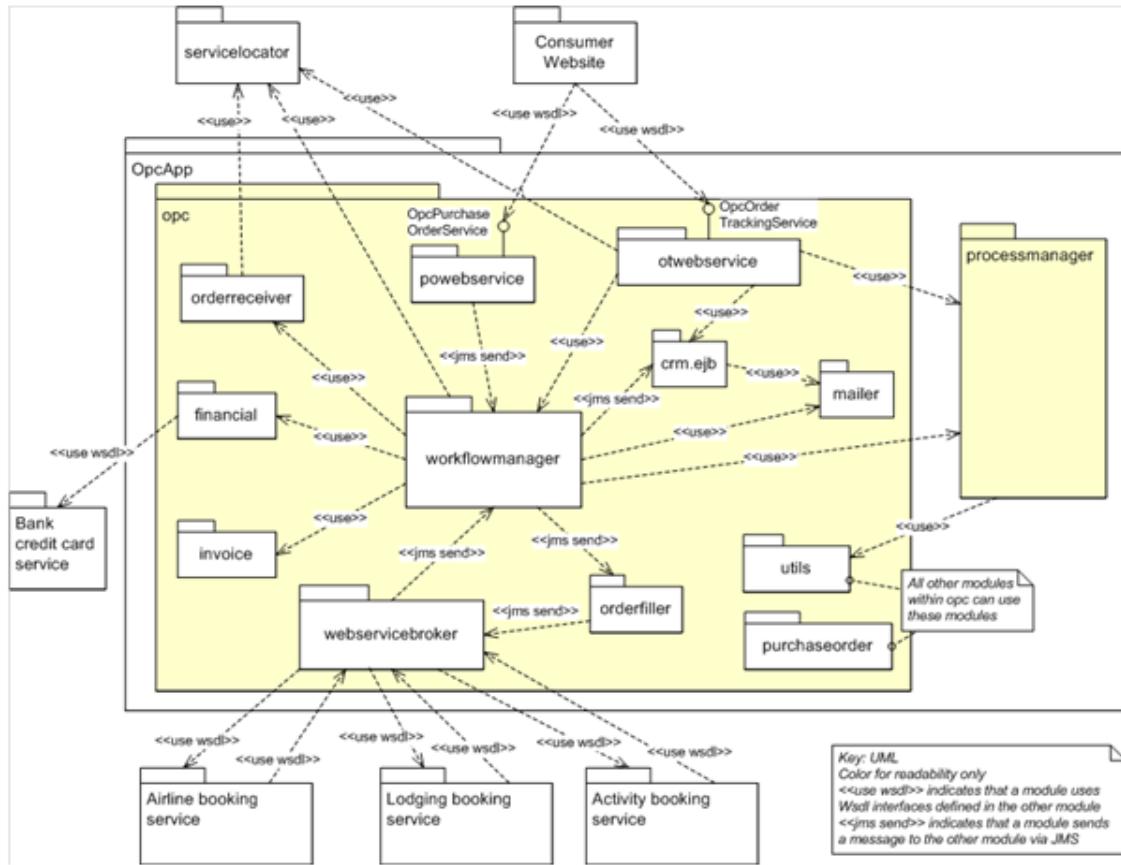
- Los desarrolladores están familiarizados con desarrollo en EJB y desarrollo basado en componentes.
- Estos componentes EJB altamente modulares promueven el reúso.

1.1.1.4. [Vistas relacionadas / Related Views](#)

- Vista Padre: Top Level Module Uses View
- Refinamiento del administrador del flujo de trabajo: Module Uses View
- OPC Module Uses View

1.1.1. Vista de Usos del módulo OPC / OPC Module Uses View

1.1.1.1. Presentación básica / Primary Presentation



1.1.1.1. Catálogo de elementos / Element Catalog

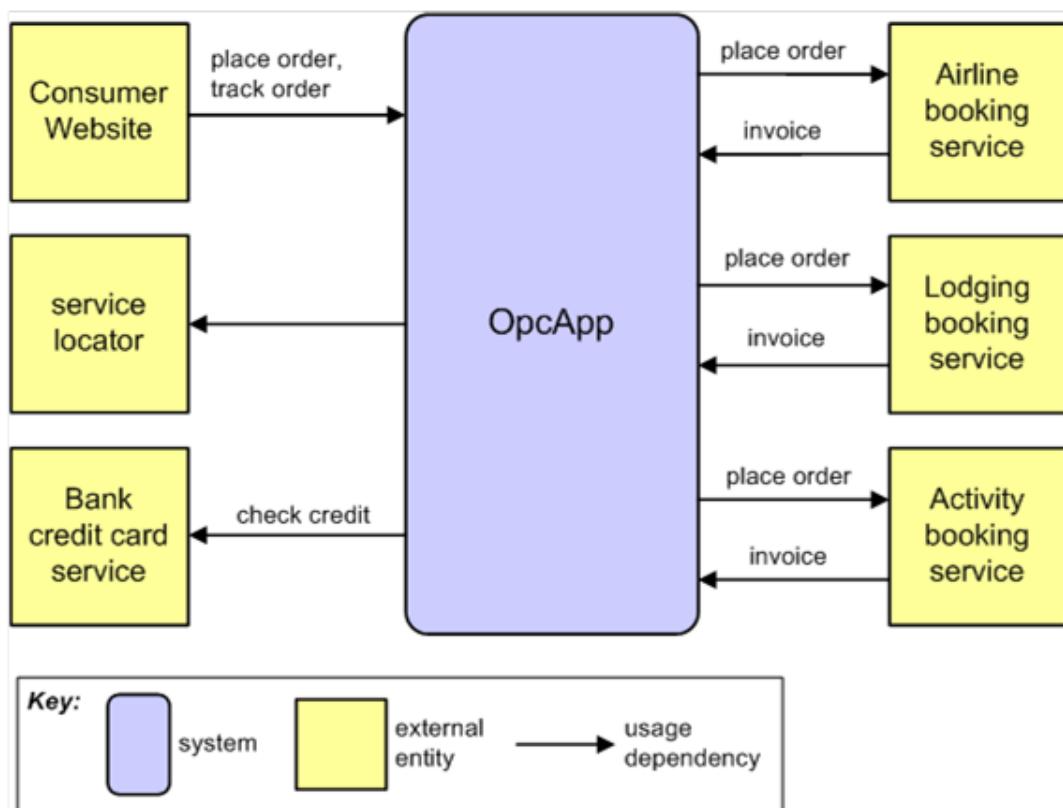
Elemento	Descripción
OpcApp	<p>OpcApp es la Aplicación del Centro de Procesamiento de Órdenes. La lógica de negocio de Adventure Builder es implementado en este módulo. Esto provee las funcionalidades siguientes:</p> <ul style="list-style-type: none"> • Aceptar las peticiones de orden de compra del ConsumerWebsite para que sean procesadas por el Servicio Web de Compra de Orden. • Proveer un mecanismo para que la Página Web del Consumidor pueda consultar el estado actual de una orden de compra al Servicio Web de Rastreo de Órdenes. • Comunicarse con proveedores externos para procesar y mantener el estado de una orden de compra. • Tras la finalización del proceso de orden de compra, mandar un email al cliente sobre el éxito o falla del proceso.
opc	<p>Este paquete contiene toda la lógica de procesamiento de las órdenes, incluyendo flujo de trabajo, colas internas usadas para la comunicación entre los elementos e interacción con servicios web externos.</p>
crm.ejb	<p>Este es Administrador basado en la relación con los clientes (CRM en inglés). El trabajo de este módulo es enviar un correo una vez que una orden haya sido completamente y satisfactoriamente procesada. Este es implementado como un bean basado en mensajes. En el futuro este módulo puede albergar información adicional acerca de los clientes, lo cual podría asistir en proveer a los clientes de una mejor experiencia. Este podría incluir cosas como el historial de compras de un cliente en particular o enviar correos periódicamente acerca de nuevas ofertas.</p>

factura	Este paquete contiene una estructura de datos que alberga información que OPC usa para comunicarse con proveedores externos. La estructura de datos también contiene el estado de una orden en la factura (para mayor información del ciclo de vida de una orden ver el diagrama de estados en el OPC C&C view).
remitente	El remitente es un módulo de ayuda y su responsabilidad primaria es enviar correos usando el API de Java Mail. Este es provisto de un mensaje y direcciones de correos para enviar los correos. En el futuro este módulo será movido al paquete utils fuera de opc.
financiero	El módulo financiero es responsable por verificar y cobrar a la tarjeta de crédito del cliente. Para este propósito, este llama a los servicios web externos provistos por el banco. La verificación de la tarjeta de crédito pasa de una manera síncrona y la aplicación OPC espera a que el servicio web externo responda para proseguir. Si la respuesta del servicio de banca no es positiva, la aplicación OPC no procesará la orden.
utils	Este paquete contiene clases de utilidad que son usadas por la aplicación OPC, tal como código de unión para el API JMS. En el futuro, este módulo será movido al paquete útil fuera de OPC.
orderfiller	Este módulo lee una cola interna de peticiones de órdenes. Cuando una orden arriba, este descompone la orden en peticiones a los diferentes proveedores envueltos. Estas peticiones son enviadas en formato XML a las colas internas.
powebbservice	<p>Este módulo provee un servicio web que es usado por ConsumerWebsite para comunicar los detalles de una orden de compra a OPC para procesarla. La interfaz del servicio web es OpcPurchaseOrderService:</p> <ul style="list-style-type: none"> • Documentación de interfaz de OpcPurchaseOrderService

otwebservice	<p>Este módulo provee un servicio web que es usado por ConsumerWebsite para preguntar el estado de una orden proveyendo el id del orden. La interfaz del servicio web es OpcOrderTrackingService:</p> <ul style="list-style-type: none"> • Documentación de interfaz de OpcOrderTracking Service
orderreceiver	El orderreceiver ayuda en la persistencia de la orden de compra en base de datos relacional.
administrador de flujo de trabajo	<p>El administrador de flujo de trabajo coordina el procesamiento de una orden de compra y rastrea el estado de una orden de compra a través de su ciclo de vida. El administrador de flujo de trabajo media la interacción a lo largo de los otros módulos internos del OPC de tal forma que estos no saben de la existencia de cada uno de los otros.</p> <p>Para mayor detalle ver el administrador de flujo de trabajo de Module Uses View</p>
purchaseorder	<p>Este paquete contiene las clases que corresponden en memoria a las entidades de data requeridas para crear una orden de compra. Por cada entidad de data hay un POJO y una entidad bean. Los POJOs son usados a través de la aplicación como objetos de transferencias de datos. Las entidades de data en este paquete son:</p> <ul style="list-style-type: none"> • Actividad • Tarjeta de Crédito • Alojamiento • Orden de Compra • Transporte • Información de Contacto • Dirección

webservicebroker	El webservicebroker es responsable de la interacción vía servicios web con la aerolínea, transporte y proveedores de actividades. Este módulo es dividido en 2 submódulos descritos abajo.
solicitante	Contiene clases que pueden invocar servicios web externos provistos por los socios de aerolínea, transporte y proveedores de actividades. Este también contiene un bean basado en mensajes que puede recibir mensajes enviados a través de una cola interna. Estos mensajes contienen exactamente las peticiones a ser enviados a los servicios web externos.
proveedor	Este módulo provee un servicio web que es visible para los socios de aerolínea, transporte y proveedores de actividades. Este servicio web es implementado en un bean de sesión y es usado por socios externos para enviar los resultados del procesamiento de solicitudes procesados por estos. Las llamadas a este servicio web son reenviadas a una cola interna.
administrador de procesos	El administrador de procesos es usado por el módulo otwebservice para traer de la base datos las órdenes de compra y sus estados actualizados. Esto es también usado por el administrador de flujo de trabajo para traer las órdenes colocadas con el proveedor externo y persistir sus estados. Este módulo contiene los submódulos (ejb y administrador.ejb) descritos luego.
ejb	Contiene una sesión bean que ofrece operaciones para traer las órdenes y actualizar el estado de una orden dada (ambas órdenes de compra adventure y las órdenes colocadas con los socios proveedores).
manager.ejb	Contiene un bean entity para persistir una orden de compra. La entidad bean usa persistencia administrada por contenedor (CMP en inglés).

1.1.1.1. [Diagrama de contexto / Context Diagram](#)



1.1.1.2. [Guía de variabilidad / Variability Guide](#)

Agregar o remover un banco

El sistema permite agregar o remover bancos socios manteniendo un registro de los servicios externos (elemento de 'registro de servicio' en la <Top Level SOA View>). Un banco nuevo tiene que implementar la interfaz de CreditCardService. En tiempo de ejecución los nuevos bancos son identificados por OPC pidiendo el registro de los servicios externos que implementan esta interfaz.

Agregar o remover un proveedor de aerolínea, alojamiento o actividad

El sistema permite agregar un proveedor socio manteniendo un registro de los servicios externos (elemento de 'registro de servicio' en la <Top Level SOA View>).

- Una nueva aerolínea tiene que implementar la interfaz AirlinePOService. En tiempo de ejecución las nuevas aerolíneas son identificadas por OPC pidiendo el registro de los servicios externos que implementan esta interfaz.
- Un nuevo proveedor de alojamiento tiene que implementar la interfaz LodgingPOService. En tiempo de ejecución los nuevos proveedores de alojamiento son identificados por OPC pidiendo el registro de los servicios externos que implementan esta interfaz.
- Un nuevo proveedor de actividad tiene que implementar la interfaz ActivityPOService. En tiempo de ejecución los nuevos proveedores de actividad son identificados por OPC pidiendo el registro de los servicios externos que implementan esta interfaz.

El módulo de webservicebroker provee una capa de abstracción dentro de OPC que contacta todos los proveedores socios externos. Otros módulos de OPC no se preocupan de los cambios del conjunto de proveedores disponibles. Otros módulos sólo conocen la identidad del nuevo proveedor.

1.1.1.1. Racionalidad / Rationale

Ver Racionalidad de OPC Runtime Refinement View

1.1.1.2. Vistas relacionadas / Related Views

Vista Padre: Top Level Module Uses View

Refinamiento del Administrador de Flujo de Trabajo: Workflowmanager Module Uses View

OPC Module Decomposition View

1.1.1. Documentación de Interfaces

1.1.1.1. Documentación de Interfaz OpcPurchaseOrderService

1.1.1.1.1. Identidad de Interfaz

Esta es una interfaz de un servicio web SOAP nombrado OpcPurchaseOrderService. El propósito principal de este servicio web es permitir a los clientes colocar una orden de compra.

1.1.1.1.2. Recursos

String submitPurchaseOrder (PurchaseOrder poObject)

Sube una orden de compra completa para un paquete de aventura. Esta operación retorna el ID de la orden como un String. La operación simplemente checa si el argumento de PurchaseOrder es válido. Si es válido, la orden es creada en la base de datos y esta operación retorna al solicitante. El procesamiento actual de la orden de compra es llevada a cabo en segundo plano. Si existe algún error imprevisto durante el proceso de la orden de compra después que esta haya retornado al solicitante, el error es manejado en otro lado y no es la responsabilidad de esta interfaz.

Pre-condiciones

- El argumento de PurchaseOrder no debe ser nulo.
- Los siguientes componentes del argumento de PurchaseOrder no deben ser nulos: user-id, email-id, locale, order date, shipping information, billing information, total price, credit card information, head count information, start date, end date and departure city

Post-condiciones

- Una llamada exitosa a esta interfaz retomará un ID único de orden de compra.

Restricciones de uso

- Autorización: puede ser solo llamado por usuarios que hayan iniciado sesión.

- **Acceso concurrente:** No hay ninguna limitación en el número de accesos concurrentes de esta interfaz. El servicio está implementado usando EJBs y las llamadas concurrentes a este son manejadas por el contenedor EJB.

Manejo de Error

- **InvalidPOException.** El servicio lanza esta excepción si la orden de compra es nula o si alguno de los componentes que constituyen la orden de compra es nulo.
- **ProcessingException.** Si el proceso de orden de compra falla después de que este haya pasado su validación (por ejemplo si existe un InvalidPOException), se lanza esta excepción. Esto indica que la orden de compra es válida pero algo salió mal mientras se grababa la orden.

boolean cancelPurchaseOrder(String orderId)

Cancela la orden de compra dada. La operación retorna true si la orden fue cancelada satisfactoriamente; retorna falso cuando el procesamiento de orden de compra está en un estado donde algunas reservaciones están ya confirmadas y no pueden ser canceladas.

Pre-condiciones

- El argumento de orderId no es nulo y corresponde a una orden de compra existente.

Pos-condiciones

- Si la operación retorna true, la orden es cancelada y todas las reservaciones correspondientes a esta son canceladas. Cargos a la tarjeta de crédito del cliente, si alguna fue hecha, son también canceladas.
- Si la operación retorna false, el estado de la orden permanece sin alterarse.

Restricciones de uso

- *Autorización: Esta operación puede ser solo llamada por un usuario que haya iniciado sesión. Un usuario cliente puede solo cancelar las órdenes que él creó. Un usuario representante de ventas puede cancelar cualquier orden.*

Manejo de error

- **InvalidPOException:** El servicio lanza esta excepción si el orderId es nulo o el orden correspondiente no existe.
- **ProcessingException:** Se lanza si el orden existe pero ya está cancelado.

1.1.1.1.1. Tipos de Datos y Constantes

Tipo PurchaseOrder - Este tipo es usado como un argumento para la operación submitPurchaseOrder. Es una estructura de datos que contiene toda la información necesaria para colocar una orden. Los atributos de un objeto PurchaseOrder son listados aquí:

```
String pold  
String userId  
String emailId  
DateTime orderDate  
DateTime startDate  
Activity[] activities  
ContactInfo billingInfo  
ContactInfo shippingInfo  
CreditCard creditCard  
String departureCity  
Transportation departureFlightInfo  
Transportation returnFlightInfo  
Lodging  
DateTime endDate  
int headCount  
String locale  
float totalPrice
```

Tipo Activity. Este tipo es usado para describir cada actividad que el usuario escoge como parte de la orden de compra.

```
String activityId  
DateTime endDate  
int headCount  
String location  
String name  
float price  
DateTime startDate
```

Tipo ContactInfo. Este tipo es usado para guardar la información de contacto del usuario.

```
Address address  
String email  
String familyName  
String givenName  
String phone
```

Tipo Address. Este tipo es usado para guardar la dirección del usuario.

```
String city  
String country  
String postalCode  
String state  
String streetName1  
String streetName2
```

Tipo CreditCard. Este tipo es usado para guardar la información de la tarjeta de crédito del usuario.

```
String cardExpiryDate  
String cardNumber  
String cardType
```

Tipo Transporation. Este tipo es usado para guardar la información la aerolínea y otra información relacionada al transporte de la orden de compra.

```
String carrier  
DateTime departureDate  
String departureTime  
String destination  
int headCount  
String origin  
float price  
String transportationId  
String travelClass
```

Tipo Lodging. Este tipo es usado para guardar información de alojamiento de la orden de compra.

```
DateTime endDate  
String location  
String lodgingId  
String name  
int noNights  
int noRooms  
float pricePerNight  
DateTime startDate
```

1.1.1.1.1. Manejo de Errores

Todas las operaciones en esta interfaz pueden elevar la siguiente excepción adicionalmente a las excepciones específicas de cada operación:

RemoteException. El solicitante recibe esta excepción cuando existe un problema de comunicación con el proveedor de servicio implementando esta interfaz.

1.1.1.1.1. Variabilidad

N/A

1.1.1.1.2. Características de Atributos de Calidad

Escalabilidad: ver discusión en Racionalidad de Deployment View.

Capacidad de Modificación: Este es un servicio asíncrono que resulta en poco acoplamiento entre la página web del cliente y el centro de procesamiento de órdenes. Además, la interfaz SOAP definida en WSDL puede ser versionada y más de una versión puede ser soportada.

1.1.1.1.3. Razonamiento y Problemas de Diseño

Granularidad de grano grueso de servicio

Actualmente solo exponemos colocar/guardar una orden como un servicio web único que incluye colocar una orden para actividades, transporte y alojamiento. No proveemos de operaciones separadas para colocar una orden solo para vuelos, almacenamiento o actividades. Además, esta interfaz es menos flexible en términos de componer diferentes tipos de órdenes. Por otro lado, subir una orden de compra completa envuelve una única llamada.

Uso de JAX-RPC para pasar parámetros

Debido a que la página del consumidor y el centro de procesamiento de órdenes residen en el mismo Enterprise, evitamos el uso de procesamiento de XML complejos y pasamos parámetros como objetos Java. Esto hace la interfaz un poco menos interoperable pero simplifica implementación.

Publicación del servicio web como WSDL

Este servicio web es publicado como un WSDL en un bien conocido lugar pues no está disponible para uso público en general. Es solo usado por la página web del consumidor. La opción para usar SOAP en lugar de Java RMI o llamadas directas a EJB es motivada por la posibilidad de reemplazar la implementación de la página web del consumidor con una tecnología diferente (por ejemplo .NET); SOAP puede proveer la interoperabilidad en ese caso.

Uso de tipo de terminal EJB

Escogimos el tipo de terminal EJB porque el centro de procesamiento de órdenes está implementado usando un conjunto de beans de sesión.

1.1.1.1. Guía de uso

```
Context ic = new InitialContext();
Service opcPurchaseOrderSvc = (Service)
ic.lookup("java:comp/env/service/OpcPurchaseOrderService");
PurchaseOrderIntf port = (PurchaseOrderIntf)
opcPurchaseOrderSvc.getPort(PurchaseOrderIntf.class);
String orderId = port.submitPurchaseOrder(myPurchaseOrder);
```

1.1.1.2. Interfaz de Documentación OpcOrderTrackingService

1.1.1.2.1. Identidad de Interfaz

Esta es la interfaz del servicio web SOAP nombrado **OpcOrderTrackingService**. El propósito principal de este servicio web es rastrear el estado de una orden de compra con un id de orden dado.

1.1.1.1. Recursos

OrderDetails getOrderDetails(String orderId)

Pre-condiciones

- El argumento orderId no debe ser nulo
- El orderId debe ser un id de orden válido.

Post-condiciones

- Retornar los detalles de la orden de compra, el cual incluye el estado de cada uno de los pedidos de aerolínea (ambos vuelos de ida y vuelta), pedidos de reservaciones de hotel y pedidos de actividades.

Restricciones de uso

- **Autorización:** Solo un usuario con sesión iniciada está autorizado a llamar esta operación. Un usuario cliente puede solo obtener información acerca de las órdenes que este creó. Un usuario representante de ventas pueden obtener información acerca de cualquier orden.
- **Acceso concurrente:** No hay limitación en el número de accesos concurrentes a esta interfaz. El servicio es implementado usando EJBs y las llamadas concurrentes a este son manejadas por el contenedor EJB.

1.1.1.2. Tipos de Datos y Constantes

- **Tipo OrderDetails:** Esta estructura de datos contiene PurchaseOrder con su estado. Atributos:
PurchaseOrder PO
String status

El tipo PurchaseOrder es descrito en la documentación de interfaz de OpcPurchaseOrderService.

1.1.1.1.1. Manejo de Errores

Las excepciones elevadas por esta interfaz son:

- **OrderNotFoundException**: El servicio lanza esta excepción si no encuentra un orden de compra con el id de orden dado o el id de orden fue nulo o mal formado.
- **RemoteException**: La solicitante recibe un RemoteException cuando existe un problema de comunicación con el proveedor de servicio implementando esta interfaz.

1.1.1.1.2. Variabilidad

N/A

1.1.1.1.3. Características de Atributos de Calidad

Escalabilidad: ver discusión en Racionalidad de Deployment View.

Performance: la operación debe retornar los resultados en menos de 1 segundo.

1.1.1.1.4. Razonamiento y Problemas de Diseño

Enfoque Java-to-WSDL para el diseño de la interfaz

Debido a que el website del consumidor y el centro de procesamiento de órdenes residen en el mismo enterprise, preferimos la facilidad de desarrollo provista por el compilador JavaToWSDL sobre la estabilidad de una descripción de WSDL mantenido manualmente. Sin embargo, esto puede resultar en reescribir los clientes de este servicio cada vez que hacemos un cambio en esta interfaz.

Uso de JAX-RPC para pasar parámetros

Debido a que la página del consumidor y el centro de procesamiento de órdenes residen en el mismo Enterprise, evitamos el uso de procesamiento de XML complejos y pasamos parámetros como objetos Java. Esto hace la interfaz un poco menos interoperable pero simplifica implementación.

Publicación del servicio web como WSDL

Este servicio web es publicado como un WSDL en un bien conocido lugar pues no está disponible para uso público en general. Es solo usado por la página web del consumidor. La opción para usar SOAP en lugar de Java RMI o llamadas directas a EJB es motivada por la posibilidad de reemplazar la implementación de la página web del consumidor con una tecnología diferente (por ejemplo .NET); SOAP puede proveer la interoperabilidad en ese caso.

Uso de tipo de terminal EJB

Escogimos el tipo de terminal EJB porque el centro de procesamiento de órdenes está implementado usando un conjunto de beans de sesión.

1.1.1.1.1. Guía de uso

```
Context ic = new InitialContext();
OpcOrderTrackingService opcOrderTrackingSvc =
    (OpcOrderTrackingService)
ic.lookup("java:comp/env/service/OpcOrderTrackingService");
OrderTrackingIntf port = (OrderTrackingIntf)
opcOrderTrackingSvc.getOrderTrackingIntfPort();
OrderDetails orderDetails = port.getOrderDetails(orderId);
```

1.1.1.1. Vista de Usos del módulo Workflowmanager / Workflowmanager Module

Uses View

1.1.1.1. [Presentación básica / Primary Presentation](#)

1.1.1.2. [Catálogo de elementos / Element Catalog](#)

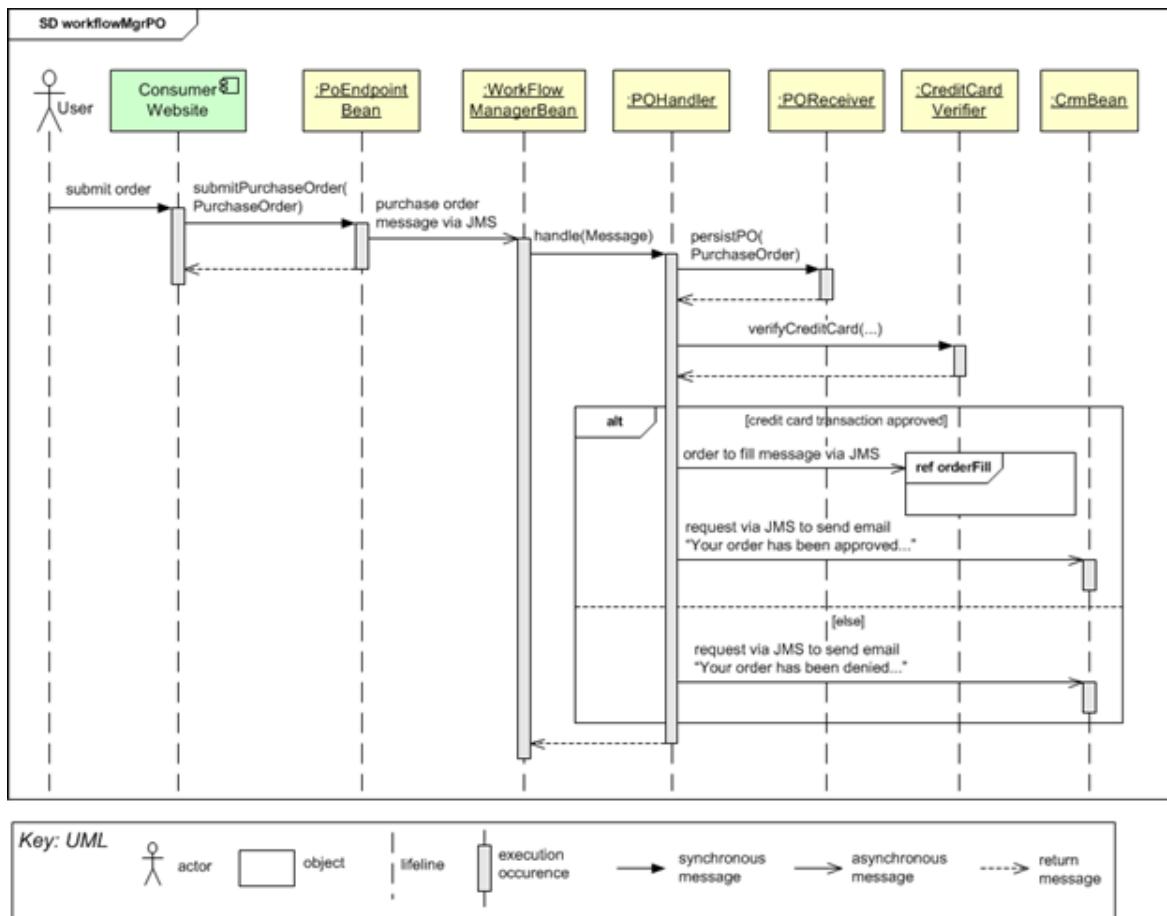
Para una descripción de elementos no listados abajo ver *Catálogo de Elementos de OPC Module*
Uses View.

Elemento	Descripción
PoEndPointBean	Esta clase es un bean de sesión sin estado. Este provee la implementación para el servicio web OpcPurchaseOrderService. Cuando este recibe la orden de compra, verifica que todos los argumentos son válidos y si está todo bien lanza la orden al administrador de flujo de trabajo enviando un mensaje a la cola JMS.
WorkFlowManagerBean	Esta clase es un bean basado en mensajes. Este es activado cuando existe un mensaje en la cola. Este procesa 2 clases de mensajes: <ul style="list-style-type: none">■ mensaje de orden de compra: en este caso este llama a la clase PoHandler. El diagrama de secuencia de abajo muestra las interacciones envueltas en el proceso de orden de compra.■ mensaje factura: este es un mensaje que viene de uno de los proveedores externos en respuesta de una reserva de orden. WorkFlowManagerBean llama a la clase InvoiceHandler para que se haga caso de estos mensajes. WorkFlowManagerBean también coloca un temporizador con el contenedor EJB para que este sea activado periódicamente para verificar el estado de todas las órdenes pendientes.

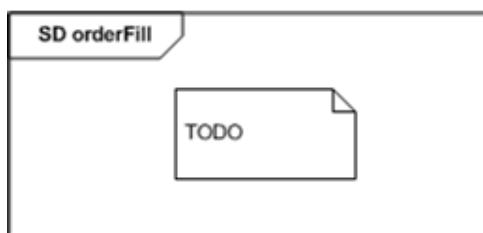
PoHandler	<p>La clase PoHandler no es visible fuera del paquete de administrador de flujo de trabajo. Este actúa como un delegado de la clase WorkFlowManagerBean para administrar cualquier requerimiento de orden de compra. Cuando procesa un pedido, este primero usa POReceiver y el administrador de procesos para insertar la orden en la base de datos en el estado Pendiente. Luego, este llama a CreditCardVerifier sincrónicamente para cargar la tarjeta de crédito del cliente. Si la tarjeta de crédito es correcta, manda un mensaje a la cola JMS para que sea procesada por OrderFillerBean. Finalmente, envía otro mensaje JMS a CrmBean, el cual creará y enviará un correo electrónico al cliente informando acerca del estado de la orden.</p>
InvoiceHandler	<p>El InvoiceHandler no es visible fuera del paquete de administrador de flujo de trabajo. Este actúa como un delegado de la clase WorkflowManagerBean para administrar cualquier factura que este reciba de cualquier proveedor. Cuando una factura es recibida, InvoiceHandler básicamente usa un administrador de procesos para actualizar el estado de la orden correspondiente.</p> <p>Una orden de paquete de aventura puede consistir de:</p> <ul style="list-style-type: none"> ■ 0 ó 1 reserva de hotel ■ 0, 1 ó 2 vuelos de aerolíneas (vuelos de ida y vuelta) ■ 0 o más ítems de actividades <p>Cuando InvoiceHandler recibe la última factura confirmando la finalización de la reserva, este envía un mensaje JMS a CrmBean para notificar al cliente vía correo electrónico.</p>
CrmBean	<p>Esta clase es un bean basado en mensajes. Este es responsable de manejar la comunicación con los clientes vía correo electrónico.</p>
CreditCardVerifier	<p>Esta clase es usada para verificar la información de crédito para el usuario. Este contacta a un servicio web externo para conseguir la información bancaria para el número de tarjeta de crédito especificado.</p>

OrderFillerBean	Esta clase es un mensaje basado en mensajes. Es responsable de enviar todos los pedidos de reserva a todos los proveedores de aerolíneas, alojamiento y actividades involucrados en una orden de compra dada.
POReceiver	La responsabilidad del destinatario de orden de compra es crear una entidad bean de orden de compra y persistirla en una base de datos relacional. El POReceiver es implementado como una clase Java y este crea una entidad bean de PurchaseOrder.
Invoice	Esta clase de "objeto de valor" almacena la data para una factura que arriba de cualquier proveedor externo. Este tiene un método para proveer una representación en XML de la factura.

1.1.1.1. Diagrama de secuencia / Sequence diagram - purchase order
within workflowmanager



1.1.1.1. Diagrama de secuencia / Sequence diagram - order filling



1.1.1.2. Diagrama de contexto / Context Diagram

N/A

1.1.1.3. Guia de variabilidad / Variability Guide

SMTP host, contenidos del cuerpo del mensaje, asunto, direcciones del remitente y direcciones de "respuesta a" para todos los mensajes de correo

electrónico enviados por CrmBean son configurables vía archivos de textos que pueden ser cambiados en tiempo de ejecución, pero requiere reiniciar la aplicación para tomar efecto. Estos archivos son implementados usando soporte estándar Java i18n.

Ver también Guía de Variabilidad de OPC Runtime Refinement View

1.1.1.1. [Racionalidad / Rationale](#)

Ver Racionalidad de OPC Runtime Refinement View

1.1.1.2. [Vistas relacionadas / Related Views](#)

Vista Padre: OPC Module Uses View

1.1.2. Data Model

1.1.2.1. [Presentación básica / Primary Presentation](#)

1.1.2.2. [Catálogo de elementos / Element Catalog](#)

Elemento	Descripción
UserAccount	Un usuario final de la aplicación AdventureBuilder. Guardamos el id de correo electrónico, contraseña e información de contacto.
Transportation	Cada entrada de transporte es un vuelo disponible para reservar en nuestros paquetes de viajes. Para cada uno, nosotros registramos: nombre, aeropuertos de arribo y partida, días y tiempo, nombre de la aerolínea, número de vuelo, tarifa, clase de cabina.
Lodging	Un hotel, casa de huéspedes o B&B que pueden ser usados para alojamiento en nuestros paquetes de viaje. Por cada tipo de alojamiento, nosotros registramos nombre, descripción, información de locación, descripción de cuarto, tarifas.
Package	Un paquete de viaje disponible en nuestro catálogo. Un paquete especifica alojamiento y una lista de actividades. Los atributos de un paquete incluyen nombre, descripción, tarifa por persona,

Category	Una categoría de paquetes de viaje de aventura. Ejemplos: paquetes de isla, aventuras de montaña. Esta categorización ayuda al usuario a navegar a través de nuestro catálogo de paquetes. La data de una categoría consiste de un nombre, descripción y una imagen representativa para mostrar al usuario.
Activity	Una actividad de aventura disponible. Ejemplos: buceo, pesca, observar aves, canotaje, surfing. Las actividades son disponibles en paquetes selectos. La información guardada por cada actividad incluye: nombre, descripción, tarifa y una imagen representativa para mostrar al usuario.
ActivityInPackage	Esta entidad representa la relación muchos a muchos entre actividad y paquete. Este simplemente lista las actividades en cada paquete.

1.1.1.1. [Diagrama de contexto / Context Diagram](#)

N/A

1.1.1.2. [Guia de variabilidad / Variability Guide](#)

N/A

1.1.1.3. [Racionalidad / Rationale](#)

N/A

1.1.1.4. [Vistas relacionadas / Related Views](#)

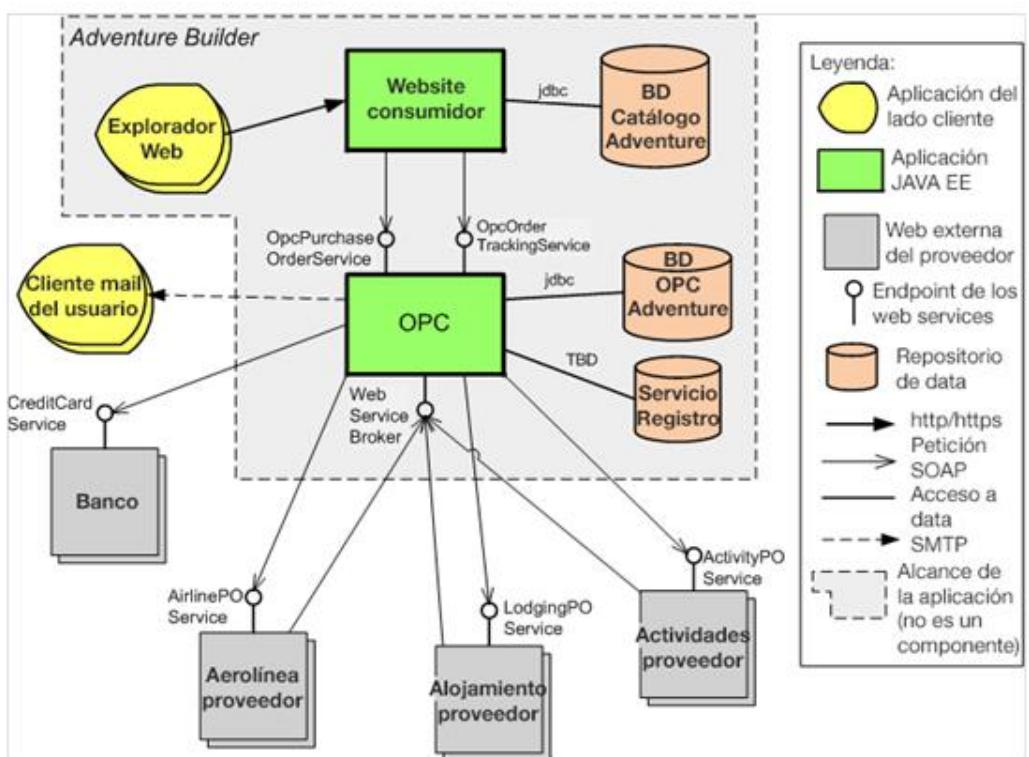
N/A

1.2. Vistas Componente y Conejor / C&C Views

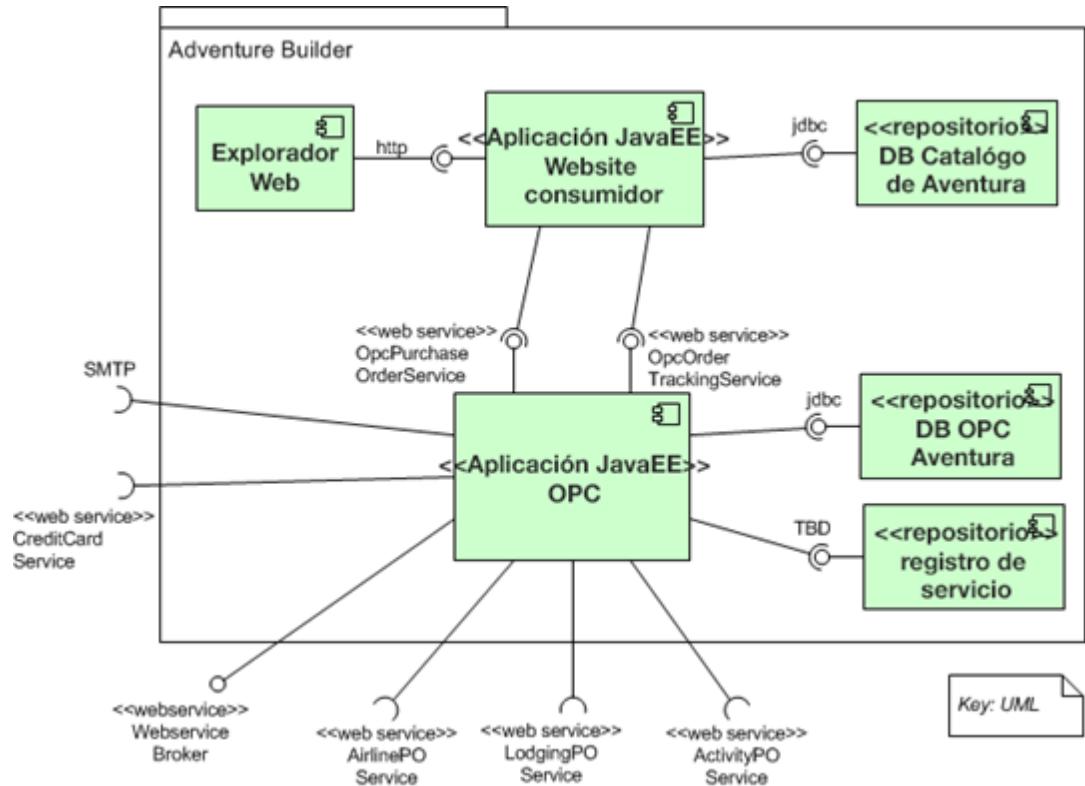
1.2.1. Vista SOA de alto nivel / Top Level SOA View

1.2.1.1. Presentación básica / Primary Presentation

1.2.1.1.1. Notación informal

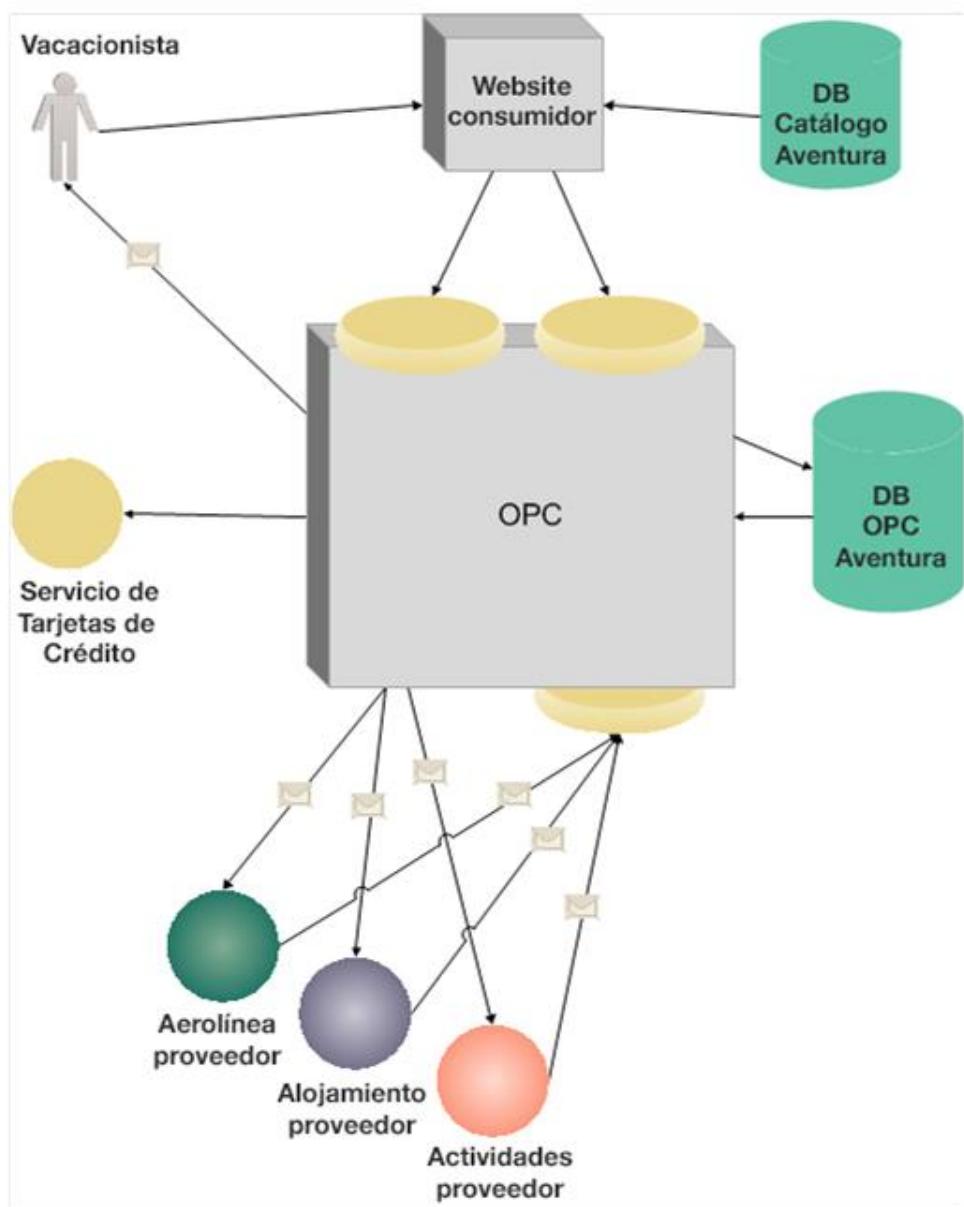


1.1.1.1.1. UML



Key: UML

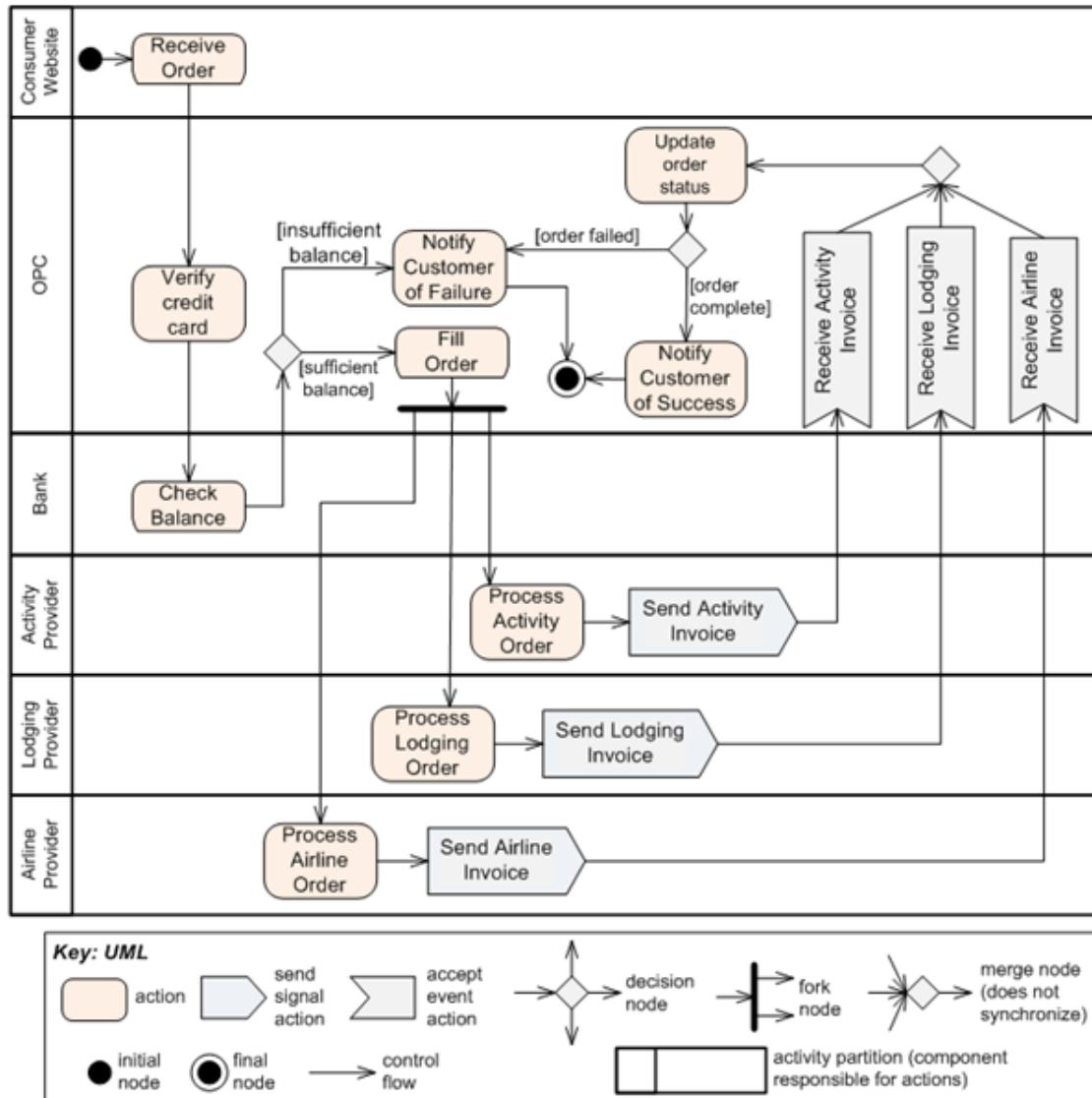
1.1.1.1. Notación soapatterns.org

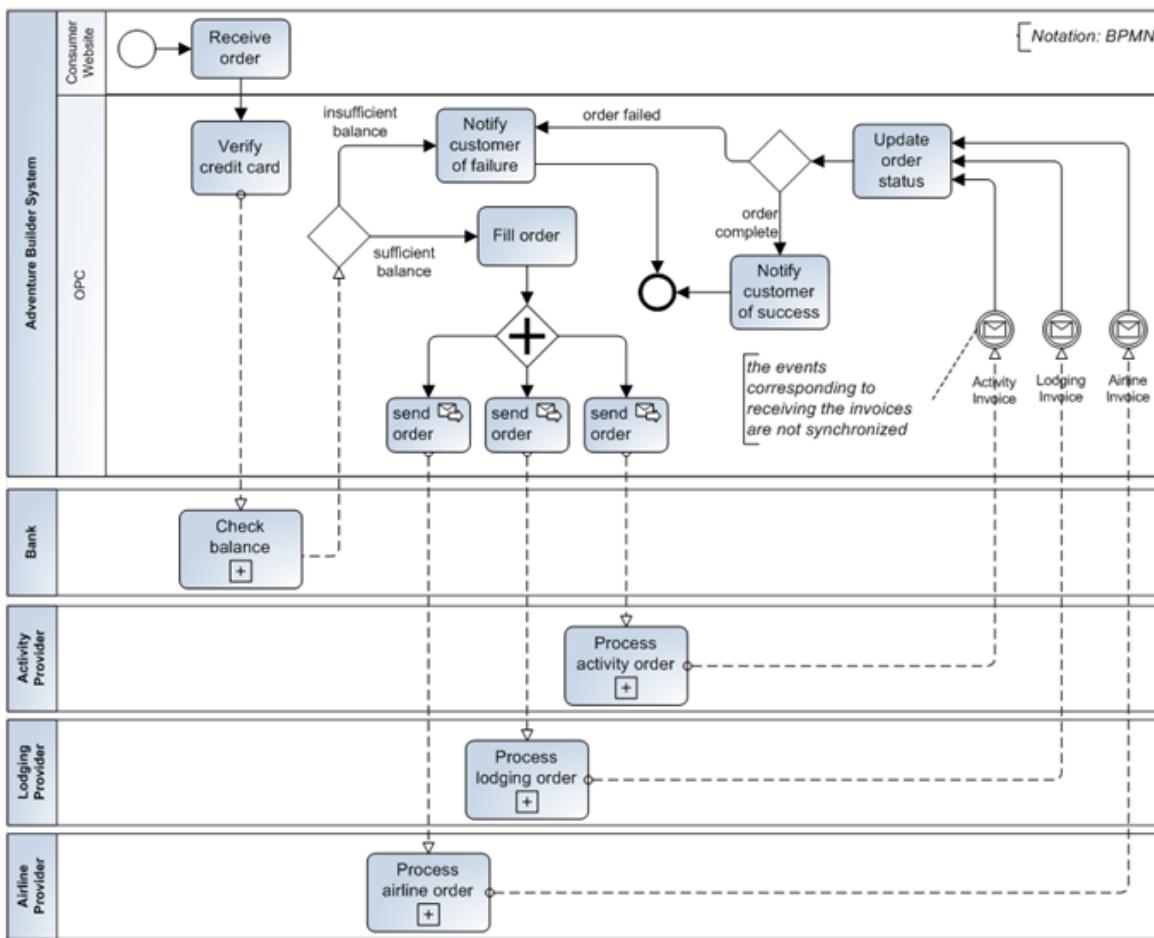


1.1.1.1. Catálogo de elementos / Element Catalog

Elemento	Descripción
Explorador web	<p>Este componente representa la interfaz de usuario de la aplicación en ejecución sobre un explorador web. El website consumidor es una aplicación web 2.0 implementada empleando GWT. Además, el explorador ejecuta código Javascript que hace uso de AJAX para comunicarse con el servidor. Empleando el explorador web, un cliente de Adventure Builder abre un sitio web, navega a través del catálogo existente de paquetes de aventura, ingresa una orden y realiza el seguimiento de las órdenes existente.</p>
Website consumidor	<p>El website consumidor es una aplicación de multicapas implementada usando tecnología Java EE. Es la parte del sistema de Adventure Builder que da la cara al cliente. Es implementado empleando código GWT, así como un número de páginas JSP y HTML, además de componentes estándar del framework WAF. La responsabilidad primaria es de procesar las peticiones http entrantes de los clientes que navegan por el catálogo o que ingresan/siguen una orden. Las peticiones para ingresar o seguir una orden son retransmitidos a la aplicación OPC a través de servicios web SOAP.</p> <p>Para una descripción de los componentes internos del Website consumidor revisar la <Vista multicapas del Website Consumidor></p>
OPC	<p>OPC es la aplicación central de procesamiento de órdenes. Es una aplicación Java EE que se comunica con componentes externos empleando servicios SOAP. Internamente, consiste en EJBs desacoplados que se comunican entre ellos usando principalmente mensajería asíncrona. La arquitectura interna sigue el patrón de diseño de un canal de mensajería. [Hohpe 2003]. Ver la <Vista de OPC C&C> para una descripción de los componentes internos del OPC.</p> <p>La funcionalidad principal del Adventure Builder está implementada en este módulo. Sus funciones mayores son:</p> <ul style="list-style-type: none"> • Aceptar peticiones de compra de órdenes del Website consumidor para su procesamiento. • Completar una orden de compra comunicándose con proveedores externos. • Proveer un mecanismo al Website consumidor para consultar el estado de una orden de compra. • Luego de completar el proceso de orden de compra, enviar un email al cliente reportando su éxito o fallo. <p>OPC provee de las siguientes interfaces, las cuales son expuestas como servicios web SOAP:</p>

El diagrama de actividades UML muestra el proceso de una orden de compra. Hace uso de un patrón de split-and-join para completar una orden. Por simplicidad, el diagrama no muestra la interacción con la base de datos. A continuación es el diagrama equivalente usando la notación BPMN.





1.1.1.1. Catálogo de elementos

Elemento	Descripción
Base de datos del Catálogo de Adventure	Esta es una base de datos relacional que almacena el catálogo de Adventure Builder que contiene varios paquetes de aventura. También almacena información sobre los usuarios para la autenticación de usuarios y autorización. El servidor de base de datos es un Cluster MySQL 7.0 configurado para el uso del motor InnoDB.
Base de datos del OPC de Adventure	Esta base de datos relacional almacena órdenes de compra, facturas provenientes de los proveedores externos e información relacional. El servidor de base de datos es un Cluster MySQL 7.0.

Registro de servicios	Repositorio de datos que trabaja como un registro básico de los servicios externos usados por el OPC. Más específicamente, tiene el nombre, ubicación y la metadata sobre todos los servicios web SOAP ofrecidos por los bancos, aerolíneas, y actividad de los proveedores externos. TBD: uso de la base de datos relacional o archivos basados en XML.
Banco	Este componente representa la aplicación externa hosteada por un banco o administrador de tarjetas de crédito. La aplicación provee de servicios web SOAP para verificar la información crediticia de los clientes.
Proveedor aerolínea	Este componente representa la aplicación externa hosteada por una empresa aerolínea proveedora. La aplicación provee de un servicio web SOAP para reservar un viaje aéreo.
Proveedor de alojamiento	Este componente representa una aplicación externa hosteada por una empresa proveedora de alojamientos. La aplicación provee de servicios web soap para reservar habitaciones y/o estadía.
Proveedor de actividades externas	Este componente representa una aplicación externa hosteada por una empresa proveedora de actividades externas. La aplicación provee de servicios web SOAP para reservar actividades de aventura, tales como clases de surf, climbing, y expediciones.
Cliente de email del usuario	Esta es la bandeja de mensajería del usuario final (vacacionista) quien ubicó una petición de compra de un paquete de aventura. OPC envía notificaciones de email a los usuarios vía SMTP para informar del estado de las órdenes.

1.1.1.1. [Diagrama de contexto / Context Diagram](#)

N/A

1.1.1.2. [Guía de variabilidad / Variability Guide](#)

- Configurabilidad del acceso a la data

Para configurar el almacenamiento de data de forma tal que soporte diferentes proveedores de base de datos, no queremos usar ninguna sentencia de SQL en nuestras consultas que se refieran a un proveedor específico y que puedan provocar problemas cuando se

- requiera una migración. Además usamos el patrón DAO para abstraer clientes de acceder a la data directamente.
- Comunicación a través de Email
 - Todos los parámetros para la comunicación a través de email (nombre del host, puertos y protocolos) deben estar configurados fuera del código a través de un archivo de configuración. Los parámetros son leídos en tiempo de carga de datos.

1.1.1.1. Racionalidad / Rationale

- Se escogió una arquitectura orientada a servicios (SOA) para Adventure Builder porque se requería un alto grado de interoperabilidad entre las entidades internas y externas del sistema. Esto nos brinda una gran flexibilidad en términos plataformas de implementación.
 - Por ejemplo, el componente de proveedor de aerolínea puede estar implementado haciendo uso de Java, PHP, .NET, IBM CICS o cualquier otra tecnología que soporte servicios web SOAP.
 - Internamente, es posible, por ejemplo, implementar y desplegar el Website consumidor usando .NET. OPC no necesita cambiar porque que la comunicación sea a través de servicios web SOAP permite un alto nivel de interoperabilidad.
- Se tiene una separación clara entre las interfaces de usuario y la lógica de negocio. El Website consumidor es la interfaz de usuario y el OPC provee la lógica de negocios y son dos aplicaciones separadas que juntas implementan las funcionalidades descritas en la <descripción del sistema>.
- Se escogió la implementación del registro de servicio usando un repositorio de data simple por la facilidad de implementación. La alternativa de crear un registro UDDI se descartó porque ello traería complejidad que no otorga beneficio, dado que el repositorio es solo consultado internamente (usando el formato estándar es más relevante cuando el registro será accedido por agentes externos).

1.1.1.2. Vistas relacionadas / Related Views

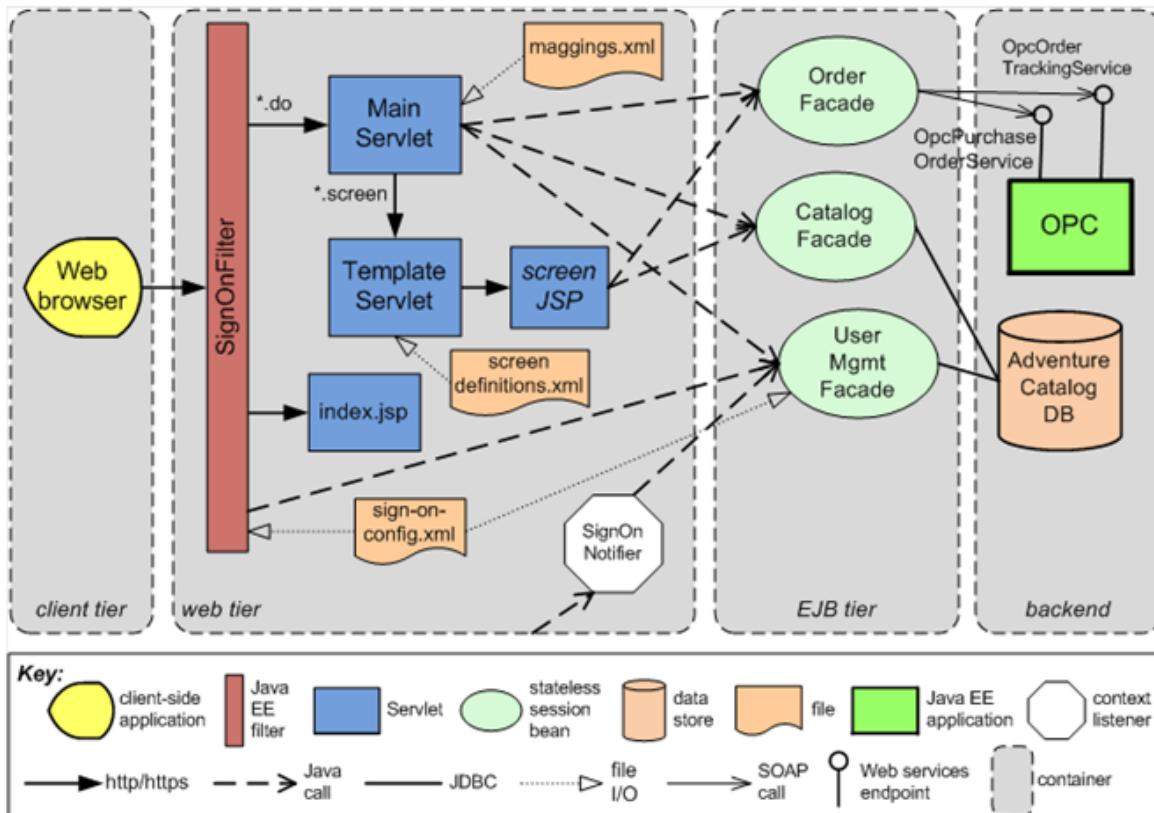
- Refinamiento del Website consumidor: <Vista multicapas del Website consumidor>
- Refinamiento del OPC: <Vista OPC C&C>
- Documentación de interfaces
 - OpcPurchaseOrderService - documentación de interfaz

- OpcOrderTrackingService - documentación de interfaz.

1.1.1. Vista de multi capas del Consumer Website / Consumer Website Multi-tier

View

1.1.1.1. Presentación básica / Primary Presentation



1.1.1.2. Catálogo de elementos / Element Catalog

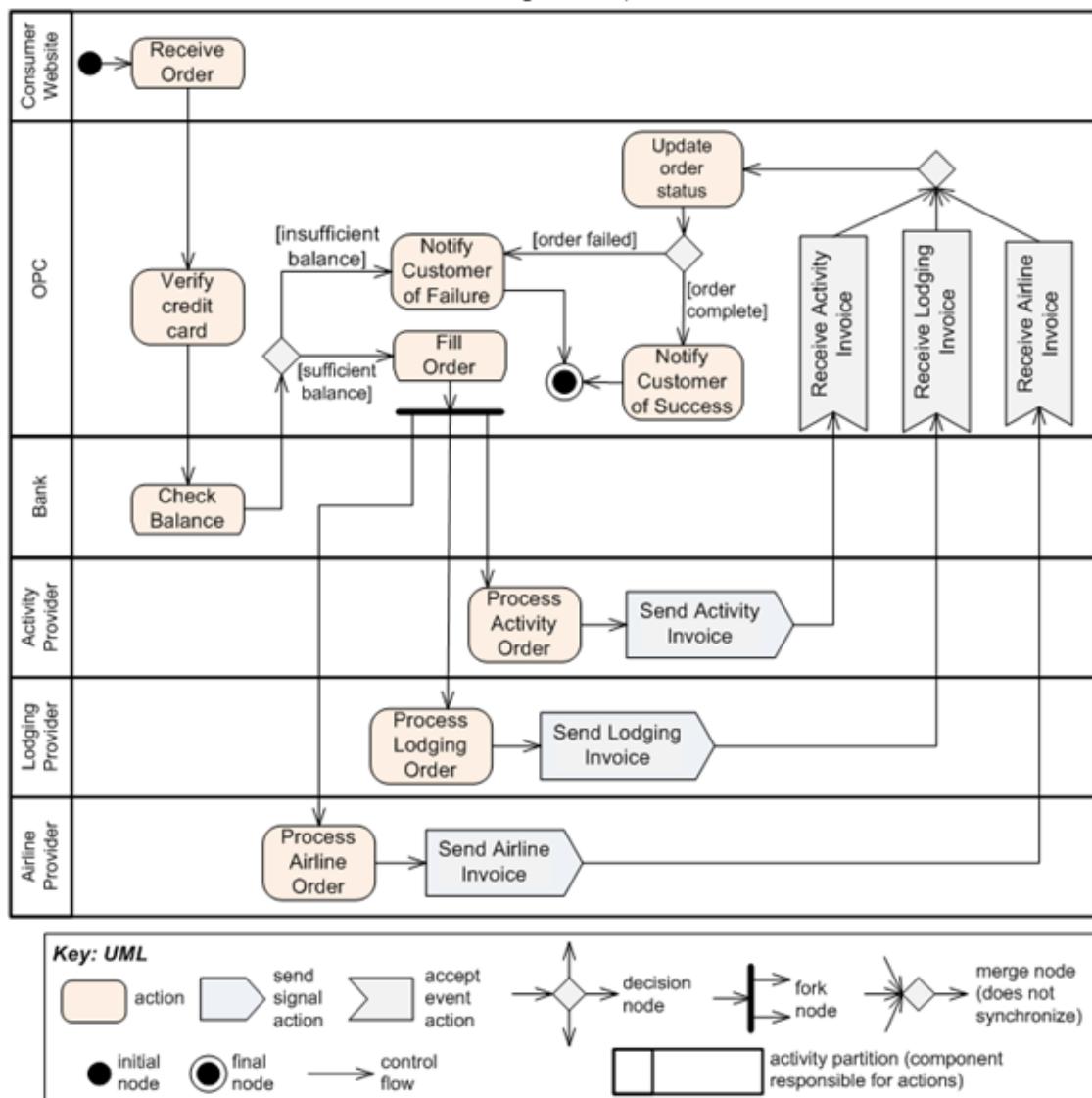
Elemento	Descripción
Explorador web	Este componente representa la interfaz de usuario de la aplicación en ejecución sobre un explorador web. El website consumidor es una aplicación web 2.0 implementada empleando GWT. Además, el explorador ejecuta código Javascript que hace uso de AJAX para comunicarse con el servidor. Empleando el explorador web, un cliente de Adventure Builder abre un sitio web, navega a través del catálogo existente de paquetes de aventura, ingresa una orden y realiza el seguimiento de las órdenes existentes.

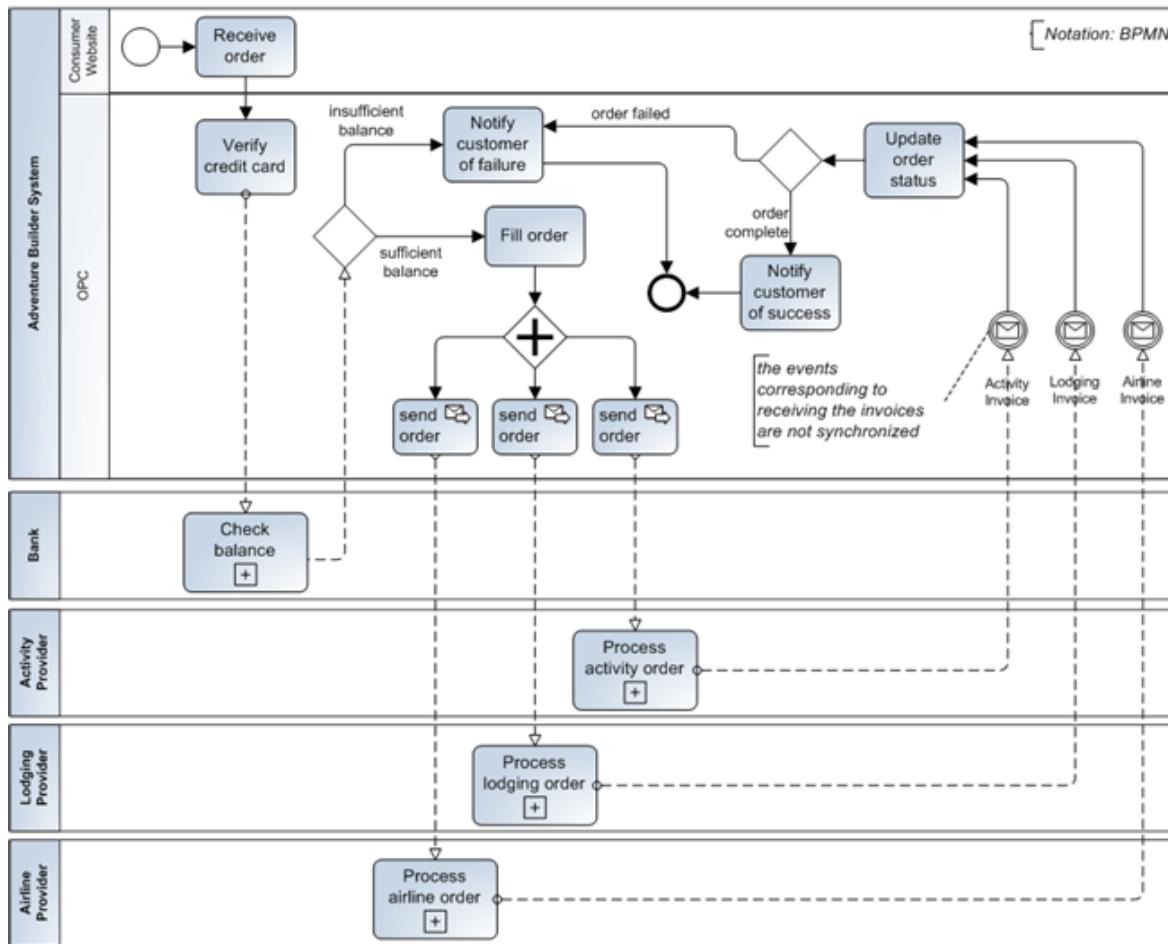
SignOnFilter (Filtro de firma)	Este componente es un filtro Java EE. Intercepta todas las peticiones http (el patrón de url es '/') y revisa si el usuario actual (si alguno) tiene la autorización para realizar la transacción pedida (identificado seleccionando la URL). Si un usuario no conectado realiza la petición que requiere de un usuario conectado, la petición lo redirige a la página de autenticación.
Sign-on-config.xml	Este archivo XML contiene información configurable sobre las restricciones de autenticación y autorización para URLs específicas (i.e, operaciones) en el sistema. También tiene las URLs para la página de 'ingreso' y página de 'error al ingresar'.
MainServlet	<p>Este componente 'servlet' es parte del framework WAF. Corresponde al componente controlador en una implementación MVC con WAF. Procesa las peticiones http a URLs que terminan con ".do" y se corresponde con los clicks del usuario. (e.g. botón de enviar en un formulario). Internamente, este servlet usa una tabla configurable cargada al inicio que mapea cada URL a la clase de acciones para el website consumidor que procesará la solicitud. Estas clases de acción se extienden de la clase HTMLActionSupport. Cada clase de acción específica instanciará POJOs específicos e interactuará con beans tipo facade específicos de la sesión. El mapeo de las URLs es leído del archivo de mappings.xml.</p> <p>Luego de la ejecución de la clase de acción, la petición http es reenviada a una URL que termina con ".screen". El mapeo entre acciones específicas y pantallas es configurable y almacenada en mappings.xml. Llamadas a *.screen son manejadas por el TemplateServlet.</p> <p>MainServlet también es responsable por la demarcación de transacción de la base de datos: si la operación solicitada es de tipo transaccional, este componente comienza una transacción y llama a las operaciones de 'submit' o 'rollback' cuando la petición solicitada es terminada.</p> <p>MainServlet también es responsable por el manejo de excepciones centralizado en la aplicación Website consumidor. Excepciones que aparecen en MainServlet son atrapadas y según un mapeo definido en mappings.xml, la petición http es dirigido a una URL ".screen" específico. (Típicamente una página de error).</p>
Mappings.xml	Archivo xml simple que contiene los mapeos de las URLs (terminadas en ".do") hacia: clases de tipo acción (subclases de HTMLActionSupport) y URLs ".screen". Este archivo también contiene mapeos de excepción de clases Java hacia URLs ".screen".

TemplateServlet	Este servlet es parte del framework WAF. Durante la inicialización, este servlet obtiene los lugares actuales y lee la data configurable i18n del archivo screendefinitions.xml. Otros parámetros configurables están definidos en web.xml (no mostrado en la presentación principal). Cuando un servlet recibe un re envío de una petición http del MainServlet, este usa el mapeo entre los nombres de las pantallas (URLs que terminan con ".screen") y páginas JSP para una vez más reenviar las peticiones a la página JSP apropiada. Estas páginas JSP son llamadas 'screen JSP'.
Screendefinitions.xml	Todas las pantallas en el sitio web tienen la misma estructura: un título, un banner, una barra al lado, un pie de página y un cuerpo. El archivo screendefinitions.xml contiene un mapeo configurable entre los nombres de las pantallas y el set de páginas JSP que componen el título, banner, barra de lado, pie de página y cuerpo de la página.
Screen JSP	Un JSP regular que corresponde a una parte de la página web. La data requerida para completar los campos en una página JSP es enviada en las peticiones http por el MainServlet cuando la acción es ejecutada. Sin embargo, en algunas situaciones una pantalla JSP realiza las llamadas a un bean tipo facade de sesión para obtener data adicional.
Index.jsp	Esta es la página de bienvenida de la aplicación Website consumidor. Básicamente tiene enlaces para autenticarse y otras operaciones básicas.
SignOnNotifier	Este componente es de tipo 'listener'. Más específicamente, es un HttpSessionAttributeListener. Es convocada por el contenedor cuando un atributo en la sesión cambia. Cuando el atributo del usuario autenticado es removido porque el usuario se desconectó o la sesión expiró, este componente remueve los recursos asignados para el usuario.
OrderFacade	Este bean de sesión sin estado es una implementación ligera del patrón 'Session Facade' [Marinescu 2002]. Provee de operaciones que corresponden a la lógica de negocios de los casos de uso relacionados a la compra de una orden. Más específicamente, provee de operaciones para registrar una orden y rastrear el estado de una orden específica. Para seguir con las operaciones, un bean de sesión interactúa con la aplicación OPC a través de servicios web SOAP. Además, el bean de sesión tiene operaciones para añadir o remover ítems del carro de compras del usuario.

CatalogFacade	<p>Este bean de sesión sin estado es una implementación ligera del patrón 'Session Facade' [Marinescu 2002]. Provee de operaciones relacionadas al catálogo de paquetes de aventura. Permite navegar y buscar en el catálogo, y ver el detalle de los paquetes de aventura.</p> <p>Internamente, este bean de sesión emplea clases DAO para obtener información de la BD Catálogo de Adventure.</p>
UserMgmtFacade	<p>Este bean de sesión sin estado es una implementación ligera del patrón 'Session Facade' [Marinescu 2002]. Provee de operaciones relacionadas con la administración de usuarios, que incluye: obtener y actualizar la data del usuario, validación del ID del usuario/password para la autenticación, bloqueo de un usuario luego de un cierto número de fallidos intentos de ingresar. Internamente, el bean de sesión hace uso de clases DAO para obtener información de la BD Catálogo de Adventure.</p>
OPC	<p>OPC es la aplicación central de procesamiento de órdenes. Es una aplicación Java EE que se comunica con componentes externos empleando servicios SOAP. Internamente, consiste en EJBs desacoplados que se comunican entre ellos usando principalmente mensajería asíncrona. La arquitectura interna sigue el patrón de diseño de un canal de mensajería. [Hohpe 2003]. Ver la <Vista de OPC C&C> para una descripción de los componentes internos del OPC.</p> <p>La funcionalidad principal del Adventure Builder está implementada en este módulo. Sus funciones mayores son:</p> <ul style="list-style-type: none"> • Aceptar peticiones de compra de órdenes del Website consumidor para su procesamiento. • Completar una orden de compra comunicándose con proveedores externos. • Proveer un mecanismo al Website consumidor para consultar el estado de una orden de compra. • Luego de completar el proceso de orden de compra, enviar un email al cliente reportando su éxito o fallo. <p>OPC provee de las siguientes interfaces, las cuales son expuestas como servicios web SOAP:</p> <ul style="list-style-type: none"> • OpcPurchaseOrderService - usado para crear una orden de compra. • OpcOrderTrackingService - usado para seguir el estado de una orden de compra. • WebServiceBroker - usado por un proveedor externo para enviar una factura de regreso al OPC. <p>OPC requiere de las siguientes interfaces, las cuales son provistas por los proveedores externos como servicios web SOAP:</p>

El diagrama de actividades UML muestra el proceso de una orden de compra. Hace uso de un patrón de split-and-join para completar una orden. Por simplicidad, el diagrama no muestra la interacción con la base de datos. A continuación es el diagrama equivalente usando la notación BPMN.





BD Catálogo Adventure

Esta es una base de datos relacional que almacena el catálogo de Adventure Builder que contiene varios paquetes de aventura. También almacena información sobre los usuarios para la autenticación y autorización. El servidor de base de datos es un cluster MySQL 7.0 configurado para usar un motor InnoDB.

1.1.1.1. [Diagrama de contexto / Context Diagram](#)

N/A

1.1.1.2. [Guía de variabilidad / Variability Guide](#)

- Internacionalización

- El Website consumidor soporta varios lenguajes. Todos los mensajes de los usuarios son almacenados en un archivo xml con una versión diferente para cada localización (los nombres de estos archivos están basados en la convención de JAVA para archivos localizados)
- Registro
 - Estos son dos parámetros configurables relacionados al registro de usuarios:
 - El número de intento de ingreso sin éxito antes de bloquear la cuenta del usuario temporalmente.
 - El tiempo en minutos que la cuenta permanece bloqueada. Solo después de que el tiempo termine, el usuario podrá intentar ingresar nuevamente.
 - Estos parámetros pueden ser modificados editando el sign-on-config.xml y reiniciando la aplicación.

1.1.1.1. Racionalidad / Rationale

- WAF
 - El framework WAF fue escogido debido a las facilidades de implementación del website consumidor ya que provee de clases plantilla para el empleo del patrón MVC. Para una operación del usuario, el desarrollador implementa una clase acción (Controller) y una página JSP que corresponde a la vista (View). El desarrollador también usa archivos de configuración para proveer un mapeo configurable entre acciones, clases de acciones, eventos y pantallas. La infraestructura WAF puede lego automáticamente tomar peticiones http e invocar las clases acciones y pantallas JSP.
 - WAF provee soporte para comunicación basada en eventos e internacionalización.
 - Como alternativa para el uso de WAF estuvo el framework Spring. Spring es más robusto y solución técnica más rica, pero fue descartado porque el equipo de desarrollo no está familiarizado con Spring pero si con WAF.

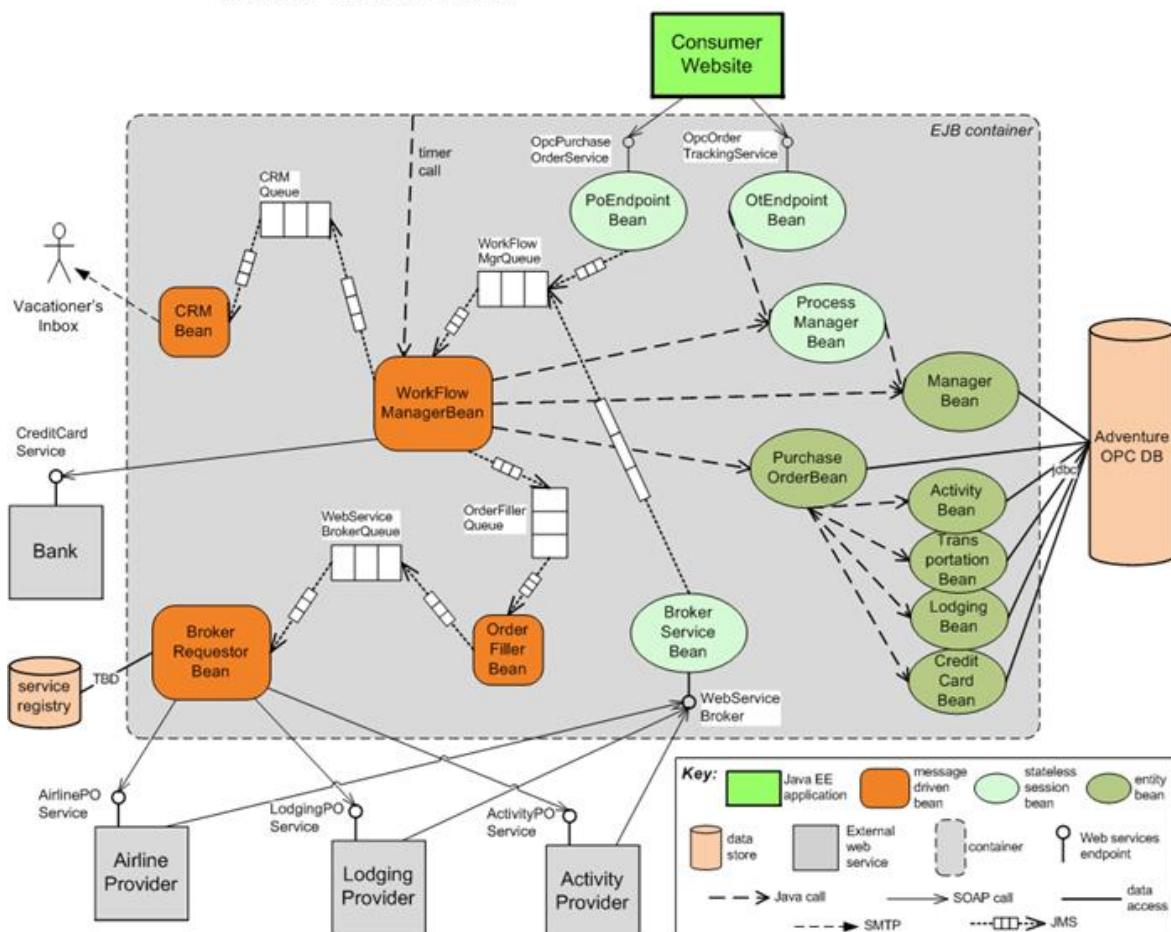
1.1.1.1. Vistas relacionadas / Related Views

- Vista padre: Vista de alto nivel de SOA
- Documentación de interfaz
 - OpcPurchaseOrderService - Documentación de interfaz
 - OpcOrderTrackingService - Documentación de interfaz

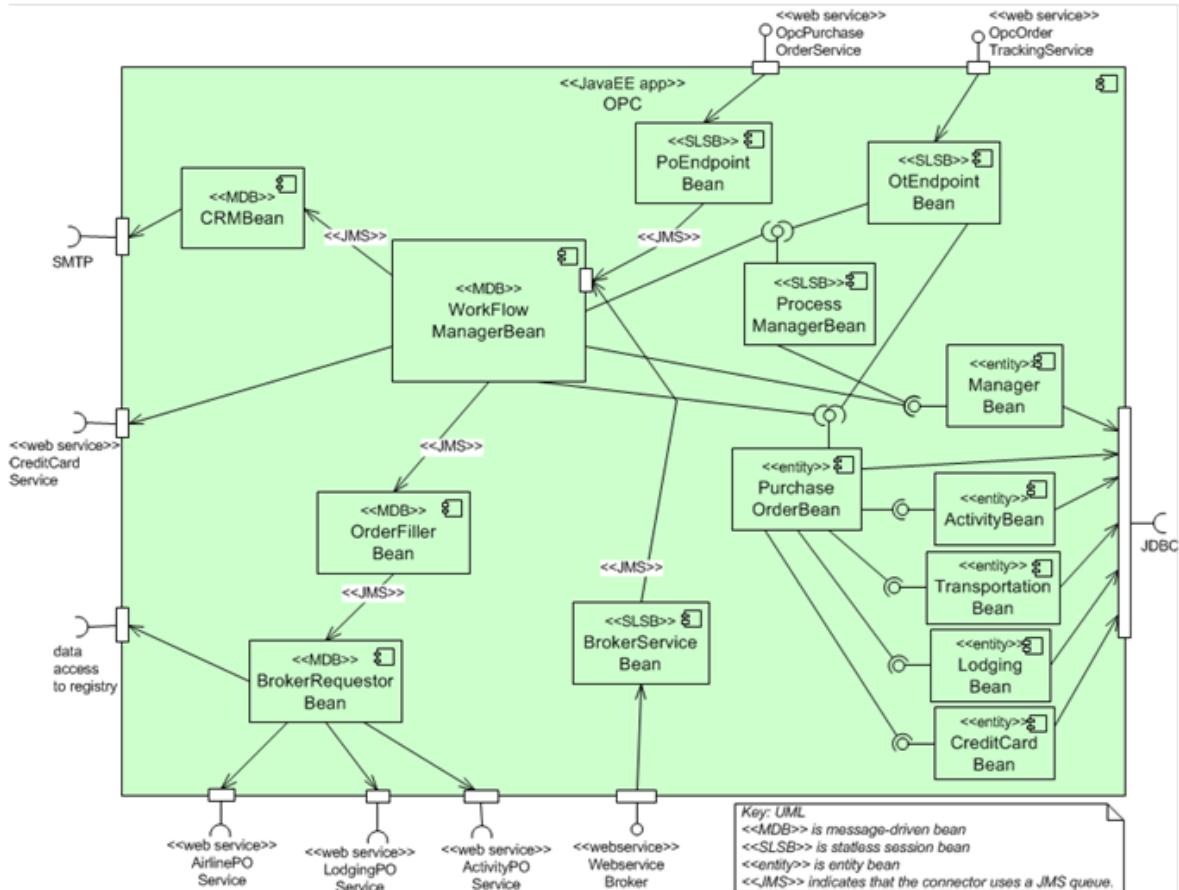
1.1.2. Vista del OPC / OPC C&C View

1.1.2.1. Presentación básica / Primary Presentation

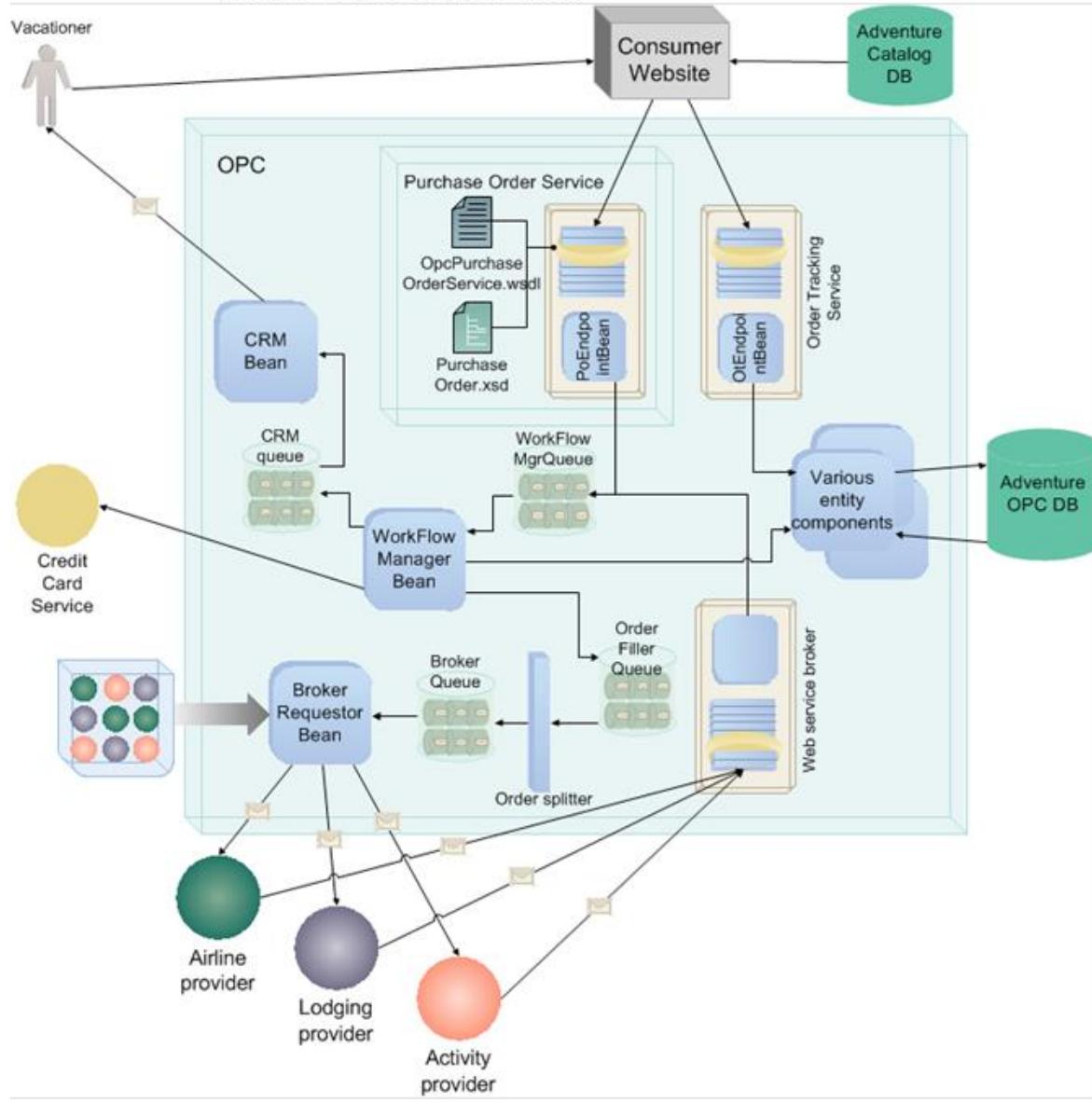
1.1.2.1.1. Notación Informal



1.1.1.1. UML

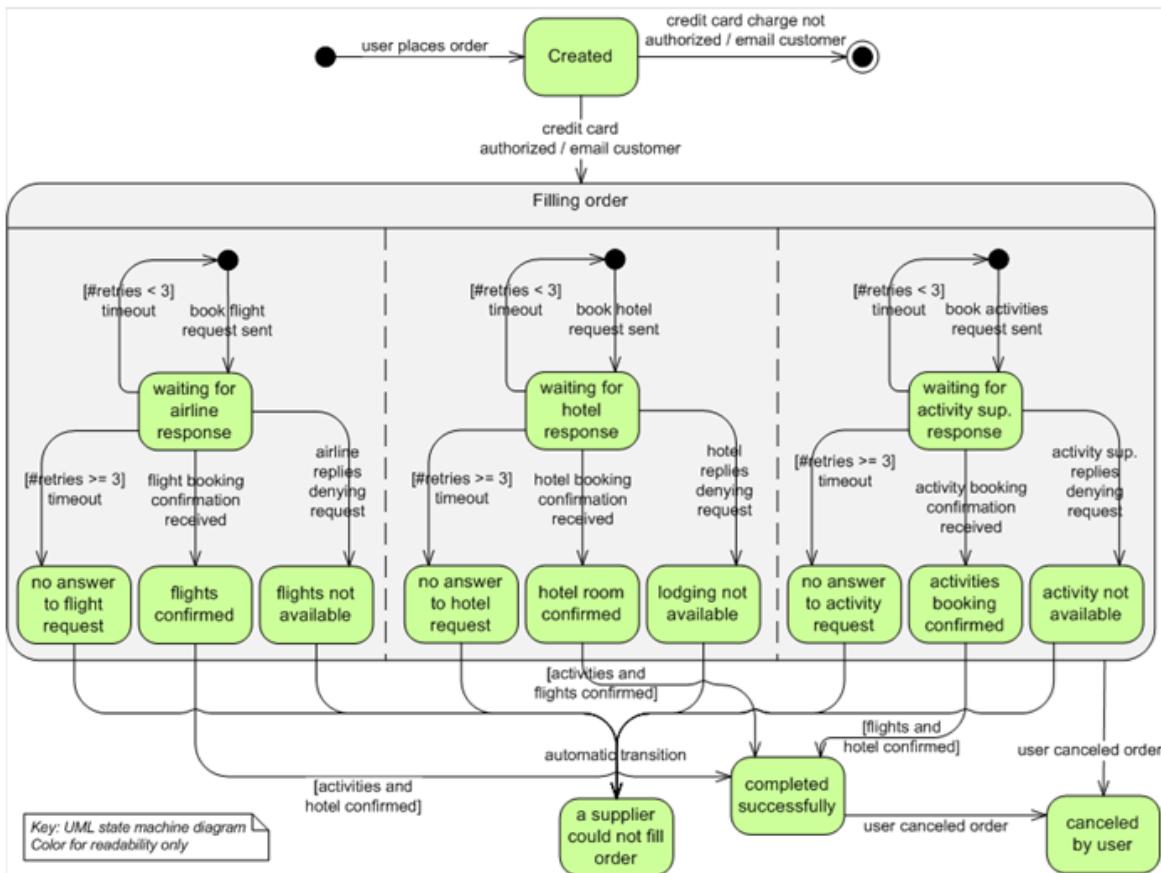


1.1.1.1. Notación soapatterns.org



1.1.1.2. Catálogo de elementos / Element Catalog

Elemento	Descripción
PoEndPointBean	Este bean de sesión sin estado implementa una interfaz de servicios web SOAP llamada OpcPurchaseOrderService. Cuando una petición de orden de compra llega, simplemente valida la orden, y si todo está bien, envía la orden al WorkFlowMgrQueue usando JMS.
OtEndPointBean	Este bean de sesión sin estado implementa una interfaz de servicios web SOAP llamada OpcOrderTrackingService. Peticiones por información sobre una orden son manejados interactuando con ProcessManagerBean. La información es obtenida usando un entity bean de PurchaseOrderBean.
WorkFlowManagerBean	<p>Este bean orientado a mensajes es activado cuando hay un mensaje en WorkFlowMgrQueue. Procesa dos tipos de mensajería:</p> <ul style="list-style-type: none"> • Mensajería de orden de compra. Cuando se procesa dichos mensajes, este componente interactúa con el ProcessManagerBean para insertar la orden en la base de datos en el estado 'Pending'. Luego interactúa sincrónicamente con el proveedor de servicios externos del banco para validar y cargar la tarjeta de crédito del usuario. Si la tarjeta de crédito está bien, se envía un mensaje al OrderFillerQueue JMS queue para ser procesado por OderFillerBean. Finalmente, se envía otro mensaje a CRMQueue, el cual será procesado por CrmBean (envía un email al cliente informando que la orden está siendo procesada). • Mensaje de facturación. Este mensaje proviene de uno de los proveedores externos en respuesta de a una orden de separación. Cuando un mensaje de facturación es recibido, este componente básicamente interactúa con ProcessManagerBean para actualizar el estado de la orden correspondiente. Si el mensaje confirma la última factura que es parte de la orden de empaque, WorkFlowManagerBean envía un mensaje JMS a CrmBean para notificar al usuario a través de un email. <p>WorkFlowManagerBean también coloca un timer con el contenedor de EJB para que periódicamente se revise el estado de las órdenes pendientes.</p> <p>WorkFlowManagerBean es uno de los dos componentes en el sistema de Adventure Builder que actúa como servicio usuario de los servicios externos (el otro es BrokerRequestorBean).</p> <p>El ciclo de vida de una orden de compra puede ser resumido en el siguiente diagrama de máquina-estado en UML.</p>



1.1.1.1. Catálogo de elementos

Elemento	Descripción
PurchaseOrderBean	Este entity bean persiste los detalles de una orden de compra (usuario que realiza la orden, fecha de la orden, precio total, número de participantes, fechas de llegada y salida, ciudad de salida, etc.). El estado de la orden cuando es procesada no es almacenado en este entity bean; sino que es manejado por ManagerBean.
ActivityBean	Este entity bean persiste los detalles de una reserva de actividad (id actividad, localización, precio, fecha/hora, número de participantes)

TransportationBean	Este entity bean persiste los detalles de una reserva de vuelo (origen, destino, empresa, id del vuelo, hora de salida, hora de llegada, clase de viaje, tarifa)
LodgingBean	Este entity bean persiste los detalles de la reserva del alojamiento (id del hotel, fecha de inicio, número de noches, precio por noche, número de cuartos)
CreditCardBean	Este entity bean persiste la información de una tarjeta de crédito (número de la tarjeta de crédito, tipo, fecha de expiración, etc.) de un usuario.
ProcessManagerBean	Este bean de sesión provee operaciones para obtener y actualizar el estado de una orden de compra y el estado de cada orden individual de los proveedores.
ManagerBean	Este bean es usado por WorkFlowManagerBean para persistir el estado de una orden de compra conteniendo el id de la orden, estado de la orden, y el estado individual de cada orden a los proveedores.
OrderFillerBean	Este componente es usado para procesar al objeto purchaseorders. Lee el objeto y lo separa en pequeñas partes, una para cada actividad, transporte y alojamiento. Luego envía cada una de las partes a un webservicebroker que luego las envía a los proveedores externos.
BrokerRequestorBean	Este bean orientado a mensajes realiza peticiones a los proveedores externos. Requiere las siguientes interfaces: <ul style="list-style-type: none"> • AirlinePOService - Usado para enviar purchaseorders a las aerolíneas proveedoras. • ActivityPOService - Usado para enviar purchaseorders a las empresas proveedoras de actividades. • LodgingPOService - Usado para enviar purchaseorders a los proveedores de alojamiento.
BrokerServiceBean	Este componente es la ventana del OPC hacia proveedores externos de actividades, alojamiento y transporte. Provee las siguientes interfaces: <ul style="list-style-type: none"> • WebServiceBroker - Este servicio web es usado por los proveedores externos para subir facturas al OPC. • getInvoice - Esta interfaz puede proveer las facturas enviadas por proveedores externos. • sendRequest - Esta interfaz puede ser usada para contactar proveedores externos y colocar órdenes.

CRMBean	Este componente es usado para el manejo de las relaciones con los clientes. Para esta aplicación se usa solamente para comunicarse con el usuario. Lee los mensajes de una cola y crea los mensajes correspondientes según plantillas ingresadas y las envía al cliente.
Website consumidor	El website consumidor es una aplicación de multicapas implementada usando tecnología Java EE. Es la parte del sistema de Adventure Builder que da la cara al cliente. Es implementado empleando código GWT, así como un número de páginas JSP y HTML, además de componentes estándar del framework WAF. La responsabilidad primaria es de procesar las peticiones http entrantes de los clientes que navegan por el catálogo o que ingresan/siguen una orden. Las peticiones para ingresar o seguir una orden son retransmitidos a la aplicación OPC a través de servicios web SOAP. Para una descripción de los componentes internos del Website consumidor revisar la <Vista multicapas del Website Consumidor>
BD Adventure OPC	Base de datos relacional que almacena las órdenes de compra, facturas provenientes de los proveedores externos e información relacionada. El servidor de la base de datos es MySQL configurado para hacer uso del motor InnoDB.
Service Registry	Repositorio de datos que funciona como un registro de los servicios externos usados por el OPC. Más específicamente, tiene un nombre, ubicación y metadata sobre todos los servicios web SOAP ofrecidos por los socios como bancos, aerolíneas, alojamientos, y centros de actividades. TBD: Uso de base de datos relacional o archivos XML
Banco	Este componente representa la aplicación externa hosteada por un banco o administrador de tarjetas de crédito. La aplicación provee de servicios web SOAP para verificar la información crediticia de los clientes.
Aerolínea proveedor	Este componente representa la aplicación externa hosteada por una empresa aerolínea proveedora. La aplicación provee de un servicio web SOAP para reservar un viaje aéreo.
Alojamiento proveedor	Este componente representa una aplicación externa hosteada por una empresa proveedora de alojamientos. La aplicación provee de servicios web soap para reservar habitaciones y/o estadía.

Centro de actividades proveedor	Este componente representa una aplicación externa hosteada por una empresa proveedora de actividades externas. La aplicación provee de servicios web SOAP para reservar actividades de aventura, tales como clases de surf, climbing, y expediciones.
Cliente de correos del usuario	Esta es la bandeja de mensajería del usuario final (vacacionista) quien ubicó una petición de compra de un paquete de aventura. OPC envía notificaciones de email a los usuarios vía SMTP para informar del estado de las órdenes.

1.1.1.1. [Diagrama de contexto / Context Diagram](#)

Ver el diagrama de contexto de la Vista de Usos del Módulo OPC

1.1.1.2. [Guía de variabilidad / Variability Guide](#)

- Agregar o eliminar un banco
 - El sistema permite añadir o eliminar bancos socios, manteniendo el registro de los servicios externos (elemento 'registro del servicio' en la vista de alto nivel SOA). Un nuevo banco tiene que implementar la interfaz de CreditCardService. En tiempo de ejecución nuevos bancos son identificados por el OPC a través de una consulta al registro por servicios externos que implementan la interfaz.
- Agregar o remover un proveedor de aerolínea, centro de alojamiento o actividades.
 - El sistema permite agregar un socio proveedor manteniendo el registro de los servicios externos (elemento 'registro del servicio' en la vista de alto nivel SOA).
 - Una nueva aerolínea tiene que implementar la interfaz AirlinePOService. En tiempo de ejecución las nuevas aerolíneas son identificadas por el OPC a través de una consulta por servicios externos que se implementa dicha interfaz.
 - Un nuevo centro de alojamiento tiene que implementar la interfaz LodgingPOService. En tiempo de ejecución los nuevos centros de alojamiento son identificadas por el OPC a través de una consulta por servicios externos que se implementa dicha interfaz.

- Un nuevo proveedor de actividades tiene que implementar la interfaz de ActivityPOService. En tiempo de ejecución los nuevos centros de actividades son identificadas por el OPC a través de una consulta por servicios externos que se implementa dicha interfaz.
- El módulo webservicebroker provee una capa de abstracción con el OPC que contacta todos los proveedores externos. Otros módulos de OPC son ajenos a los cambios en el conjunto de proveedores disponibles. Otros módulos solo saben el identificador del nuevo proveedor.
- Configuración EJB
 - Para cada EJB, un pool de instancias de beans son provistos por el servidor de aplicaciones. Existen tres parámetros que pueden ser configurados por separado para cada EJB a través de un descriptor de despliegue:
 - mínimo e inicial número de instancias de beans en el pool.
 - máximo número de instancias de beans en el pool.
 - tiempo de espera para una instancia de inactividad para ser eliminado.

1.1.1.1. **Vistas relacionadas / Related Views**

Vista padre: Vista de Alto Nivel SOA

Documentación de interfaces:

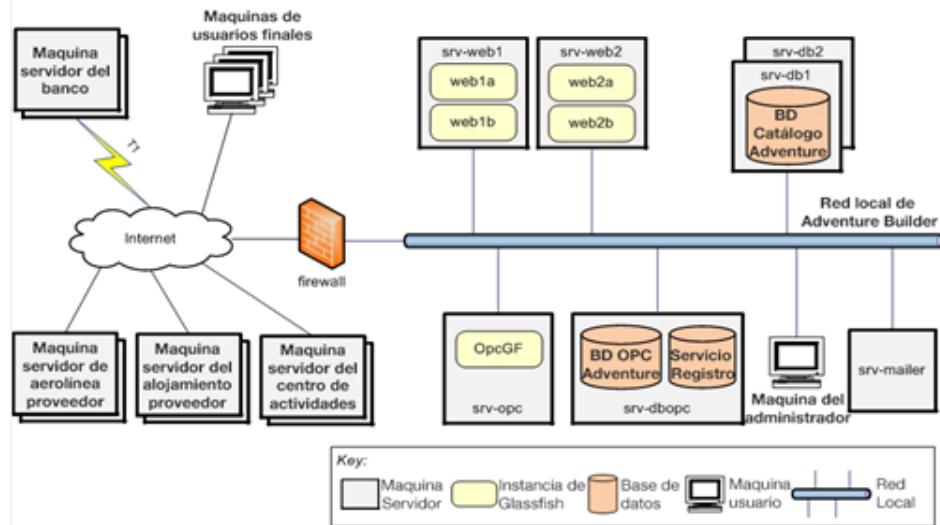
- Documentación de la interfaz OpcPurchaseOrderService
- Documentación de la interfaz OpcOrderTrackingService

1.1. Vistas de Asignación / Allocation Views

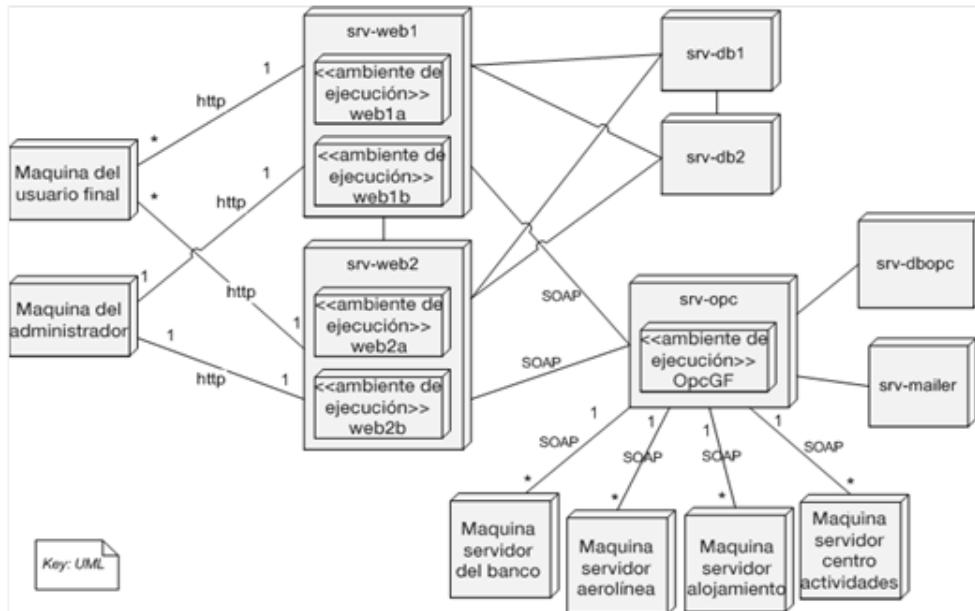
1.1.1. Vista de despliegue / Deployment View

1.1.1.1. Presentación básica / Primary Presentation

1.1.1.1.1. Notación informal



1.1.1.1.2. Notación UML



1.1.1.1. Catálogo de elementos / Element Catalog

Elemento	Descripción
Máquinas de los usuarios finales	Este nodo representa al usuario normal que puede acceder a la aplicación de Adventure Builder empleando una PC. Este emplea una página web para acceder y colocar su orden.
Máquina servidor del banco	Este nodo es externo al sistema. Representa el servidor del banco que permite verificar la cuenta de un usuario.
Máquina servidor de la aerolínea proveedora	Este nodo es externo al sistema. Representa el servidor de una aerolínea proveedora que permite registrar la compra de una orden de transporte.
Máquina servidor del alojamiento proveedor	Este nodo es externo al sistema. Representa el servidor de un alojamiento proveedor que permite registrar la compra de una orden de alojamiento.
Máquina servidor del centro de actividades proveedor	Este nodo es externo al sistema. Representa el servidor de un centro de actividades proveedor que permite registrar la compra de una orden de actividades.
Base de datos - Adventure Catalog	Base de datos relacional que almacena el Adventure Builder Catalog donde se encuentran los planes/paquetes de aventura. Además almacena información para realizar la autenticación y autorización de los usuarios. El servidor de la base de datos es MySQL Cluster 7.0 configurado para usar el motor InnoDB.
Base de datos - Adventure OPC	Base de datos que almacena las órdenes de compra, facturas de los proveedores externos e información relacionada. El servidor de la base de datos es MySQL configurada para usar el motor InnoDB.
Servicios de registro	Repositorio de datos que funciona como un registro de los servicios externos usados por el OPC. Más específicamente, tiene el nombre, ubicación y metadata sobre todos los servicios web SOAP ofrecidos por los bancos, aeropuertos, alojamiento, y proveedores de centro de actividades/actividades externas. TBD: Usar una base de datos o archivos XML

website.ear	Archivo de empresa que es desplegado en un servidor de aplicaciones. Este contiene la aplicación web principal.
opc.ear	Archivo de empresa que es desplegado en un servidor de aplicaciones. Este contiene la aplicación OPC.

1.1.1.1. Racionalidad / Rationale

- **Escalabilidad**

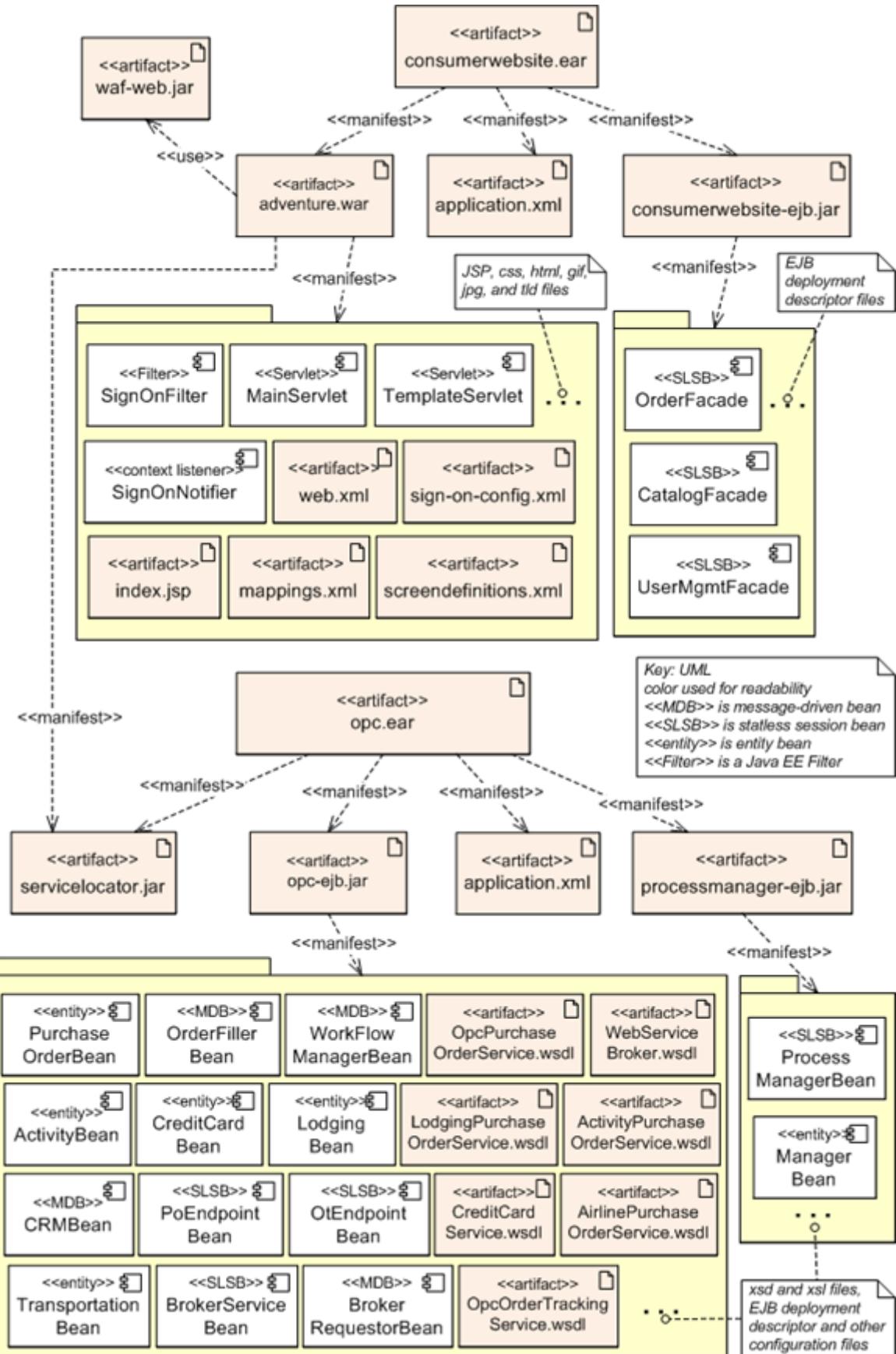
- Todos los componentes que sean servicios web SOAP de los proveedores en el sistema Adventure Builder son implementados usando beans de sesiones sin estado. Ese tipo de componente EJB puede ser replicado a través de máquinas en un cluster y cada máquina puede poner a disposición un pool de dichos EJBs. Esta funcionalidad permite escalar el número de conexiones que pueden ser manejadas. Escalabilidad horizontal es alcanzada agregando más máquinas al cluster. Escalabilidad vertical es también una opción aumentando la capacidad de una máquina y luego aumentar la cantidad de instancias de EJB en el pool de EJBs <[Referencia a OPC C&C View - Variability Guide](#)>. Sin embargo, las transacciones realizadas por los beans de sesiones sin estado requieren de acceso a la base de datos. Dado que la base de datos es difícil de replicar y puede ser un cuello de botella para el rendimiento si el número de llamadas superan las esperadas.

1.1.1.2. Vistas relacionadas / Related Views

N/A

1.1.2. Vista de instalación / Install View

1.1.2.1. Presentación básica / Primary Presentation



Notas:

- En el diagrama, paquetes sin nombre son usados solo para simplificar la representación de dependencias <> manifest > múltiples.
- Solo los artefactos más importantes son mostrados. Usamos '...' para indicar donde existen otros artefactos no presentes en el diagrama. (La caja de comentario indica que artefactos son)

1.1.1.1. Catálogo de elementos / Element Catalog

Elemento	Descripción
Application.xml	Descriptor de despliegue estándar Java EE de archivos ear. Cada archivo ear tiene su propio archivo application.xml con los valores de despliegue correspondientes.
OrderFacade, CatalogFacade, UserMgmtFacade	Ver Vista Consumer Website Multi-tier
servicelocator.jar	Librería para utilidades que implementa el patrón de Service Locator
waf-web.jar	Librería que contiene el Web Application Framework. Corresponde al paquete waf de la vista Top Level Module Uses
SignOnFilter, sign-on-config.xml, MainServlet, mappings.xml, TemplateServlet, screendefinitions.xml, index.jsp, SignOnNotifier	Ver Consumer Website Multi-tier View
opc.ear	El archivo de empresa que representa la aplicación OPC. Es desplegable sobre cualquier servidor de aplicaciones con Java EE.
opc-ejb.jar	Archivo estándar EJB jar que contiene todos los EJBs que comprenden la aplicación OPC, excepto para ProcessManagerBean y ManagerBean.
PoEndPointBean, OtEndPointBean, WorkFlowManagerBean, PurchaseOrderBean, ActivityBean, TransportationBean, LodgingBean, CreditCardBean, OrderFillerBean,	Ver OPC C&C View

OpcPurchaseOrderService.wsdl, WebServiceBroker.wsdl, OpcOrderTrackingService.wsdl	Estos archivos wsdl contienen las especificaciones de la interfaz para los servicios web SOAP que la aplicación OPC requiere. Estos servicios web son implementados y provistos por componentes internos a la aplicación OPC. Estas interfaces se encuentran en OPC C&C View.
LodgingPurchaseOrderService.wsdl , ActivityPurchaseOrderService.wsdl, CreditCardService.wsdl, AirlinePurchaseOrderService.wsdl,	Estos archivos wsdl contienen las especificaciones de la interfaz para los servicios web SOAP que la aplicación OPC requiere. Estos servicios web son implementados y provistos por socios externos. Estas interfaces se encuentran en OPC C&C View.
processmanager-ejb.jar	Archivo EJB estándar que contiene ProcessManagerBean y ManagerBean.
ProcessManagerBean, ManagerBean	Ver OPC C&C View

1.1.1.1. Racionalidad / Rationale

El sistema entero de Adventure Builder es desplegado como dos archivos ear (consumerwebsite.ear and opc.ear). Tener dos archivos ear separados permite desplegar las dos aplicaciones correspondientes en servidores de aplicaciones diferentes y en máquinas diferentes. Estos entornos de ejecución separados pueden ser configurados según las necesidades de procesamiento y comunicación de cada aplicación.

1.1.1.2. Vistas relacionadas / Related Views

N/A

1.1.2. Vista de implementación / Implementation View

2. Relación entre las Vistas

Modelo de datos de Adventure Builder	Vista de alto nivel SOA
UserAccount	Adventure Catalog DB

Transportation	Adventure Catalog DB
Lodging	Adventure Catalog DB
Package	Adventure Catalog DB
Category	Adventure Catalog DB
Activity	Adventure Catalog DB
ActivityInPackage	Adventure Catalog DB
PurchaseOrder	Adventure OPC DB
OrderStatusHistory	Adventure OPC DB
Document	Adventure OPC DB
ActivityPurchaseOrder	Adventure OPC DB
AirlineOrder	Adventure OPC DB
LodgingOrder	Adventure OPC DB
N/A	all others

Vista de Usos de alto nivel	Vista de alto nivel SOA
paquete: Adventure Builder	límite de la aplicación: Adventure Builder
Website consumidor	Website consumidor
OpcApp	OPC
N/A	Banco, aerolíneas, hospedaje, centro de actividades
N/A	Servicios CreditCard, Airline, Lodging, Activity, OPCOrderTrackingService, OPCPurchaseOrderService
N/A	AdventureCatalogDB, OPCDB
N/A	Navegador web

Vista de Descomposición del módulo OPC	Vista C&C de OPC
paquete: OpcApp	<<Aplicación JEE>> componente: OPC
paquetes: workflow manager, powbservice, processmanager, orderreceiver, invoice, purchaseorder	componente: workflowmanager
paquetes: webservicebroker, purchaseorder	componente: webservicebroker
paquetes: crm.ejb, mailer	componente: crm
paquetes: orderreceiver, invoice	componente: orderfiller

1. Directorio

1.1. Glosario y acrónimos

- Proveedor de actividades / Activity Provider
 - El proveedor de actividades es una organización externa que permite a Adventure Builder agendar una actividad.
- Ajax
 - Asynchronous JavaScript and XML. Empleando clientes Ajax se puede acceder a la data del servidor de manejar asíncrona, de esta manera creando aplicaciones web más interactivas que empleando otras tecnologías tradicionales.
- Proveedor de aerolínea / Airline provider
 - El proveedor de aerolínea es una organización externa que permite a Adventure Builder realizar gestión de viajes.
- C&C
 - Componente y conector. Es sinónimo de una vista de tiempo de ejecución.
- DAO
 - Data Access Object. DAO es un patrón de diseño común usado para abstraer el acceso de data en una aplicación.
- EJB
 - Enterprise Java Beans. EJBs son componentes de lógica de negocio (no interfaz de usuario) en la tecnología Java EE. Existen cuatro tipos de EJBs: beans de sesión sin estado, beans de sesión con estado, beans de entidad y beans orientado a mensajes.
- GWT
 - Google Web Toolkit. Es un framework para crear aplicaciones web.

- JSP
 - Java Server Page. Es una tecnología de lado del servidor para separar la interfaz de usuario de la lógica de negocio.
- Proveedor de alojamiento / Lodging provider
 - El proveedor de alojamiento u hospedaje es una organización externa que permite a Adventure Builder realizar reservas de hoteles.
- OPC
 - Order Processing Center. Es el componente central del sistema de Adventure Builder.
- Servlet
 - Es un componente del lado de servidor que usualmente es usado como controlador en el patrón de diseño Model View Controller (MVC).
- WAF
 - Web Application Framework.

1.1. Referencias materiales

Nombre corto	Referencias
[Bass 2003]	Bass, Clements, Kazman, <i>Software Architecture in Practice</i> , second edition. Addison Wesley Longman, 2003.
[Clements 2001]	Clements, Kazman, Klein, <i>Evaluating Software Architectures: Methods and Case Studies</i> . Addison Wesley Longman, 2001.
[Hohpe 2003]	Gregor Hohpe; Bobby Woolf, <i>Enterprise Integration Patterns</i> . Addison-Wesley, October 2003.
[Singh 2004]	Singh et al, <i>Designing Web Services with the J2EE 1.4 Platform</i> . Pearson Education, May 2004.

Referencias Bibliográficas

BACHMANN, Félix

2000 The Architecture Based Design Method: A Step Toward Methodical Architecture Design, Carnegie Mellon University

BACHMANN, Félix

2001 Documenting Software Architectures: Organization of Documentation Package, Carnegie Mellon University

BACHMANN, Félix

2003 Deriving Architectural Tactics: A Step Toward Methodical Architectural Design, Carnegie Mellon University

BACHMANN, Félix

2006 Attribute-Driven Design ADD, Version 2.0: Carnegie Mellon University

BARBACCI

2003 Quality Attribute Workshop (QAWs), Tercera Edición: Carnegie Mellon University

CLEMENTS, Paul

2001 Evaluating Software Architectures: Methods and Case Studies, Carnegie Mellon University

CLEMENTS, Paul

2003 Documenting Software Architecture: Views and Beyond, Carnegie Mellon University

CLEMENTS, Paul

2010 Documenting Software Architecture: Views and Beyond, Segunda Edición, Carnegie Mellon University

CLEMENTS, Paul

2012 Software Architecture in Practice, Tercera Edición, Carnegie Mellon University

KAZMAN

2003 A Life-Cycle View of Architectural Analysis and Design Methods, Carnegie Mellon University

KRUCHTEN, Philippe

1995 The 4+1 View Model of Architecture: IEEE Software

NORD

2004 Integrating the Quality Attribute Workshop (QAW) and the Attribute-Driven Design (ADD) Method: Carnegie Mellon University

RATIONAL

1998 Rational Unified Process. Best practices for software development teams: The Software Development Company