RESUMEN

En el contexto actual de la sociedad del conocimiento y con los retos a los que la educación superior debe enfrentarse, las universidades son llamadas a estar abiertas a nuevos procesos tecnológicos que permitan coexistir y comunicar sistemas informáticos construidos con diversas tecnologías, desarrollados en épocas diferentes, para varias dependencias, cumpliendo variados objetivos; es el ambiente propicio en el cual se incorpora el concepto de SOA.

Esta tesis trata un caso de implantación de SOA sencillo, en una organización como el Departamento de Desarrollo Informático de la Universidad de Cuenca, en donde no existe una metodología formal de desarrollo de software. Por lo que para implantar SOA, se hace una propuesta que se ajuste a sus necesidades, que consiste en dos partes, la primera es que luego de analizar las entidades de negocio que intervienen en los sistemas ya desarrollados, se creen procesos que puedan tomar la información y se pueda proveerla en forma de servicios para ser consumidos por otros sistemas desarrollados en la Universidad.

La segunda parte consiste en plantear un camino que recorrerían los nuevos sistemas, mediante una metodología que se adapta a las necesidades del DDI, la que se denominará RASDUC, y que además de formalizar la metodología de desarrollo de software empleada en el departamento, permitiría que las funcionalidades relevantes de los sistemas se expongan como servicios, de tal forma que una actividad importante para la Universidad se considere como un servicio accesible por todos.

PALABRAS CLAVES: SOA, Método de Integración de Sistemas, Servicios, Arquitectura de Software, Departamento de Desarrollo Informático, Metodología de Desarrollo de Software.

ABSTRACT

In the current context of the society of knowledge and the callenges that higher education has to face, universities are asked to be open to new technological processes which allow them to coexist and communicate with informatics systems built with diverse technologies, developed in different epochs, for different departments, achieving different objectives; this is the appropriate environment to incorporate the SOA concept.

This thesis develops the implementation of a simple SOA in an organization such as the Informatics Development Department, of the University of Cuenca, which there is no formal methodology for software development. In order to use SOA a proposal is made that adjusts to its needs. This proposal consists of two parts, the first one is that after the analysis of the business divisions that intervene in the already developed systems, processes which can take the information and supply them as services that can be consumed by other systems developed in the University are created.

The second part consists of proposing a path which passes through the new systems through a methodology which adapts to the needs of the IDD, which will be denominated RASDUC, it will, as well as formalize the software development methodology used in the department, allow the relevant function of the systems to be shown as services – favoring scalability, flexibility and low cost of the systems in such a way that an important activity for the university it will be considered as a service accessible to all.

Índice

UNIVERSIDAD DE CUENCA
FACULTAD DE INGENIERIA

MAESTRÍA EN GERENCIA DE SISTEMAS DE INFORMACION

"MÉTODO DE INTEGRACIÓN Y DESARROLLO DE SOFTWARE HACIA UNA ARQUITECTURA ORIENTADA A SERVICIOS PARA EL DEPARTAMENTO INFORMÁTICO DE LA UNIVERSIDAD DE CUENCA "

PROYECTO DE GRADUACIÓN PREVIO A LA OBTENCIÓN DEL GRADO DE MAGISTER EN GERENCIA DE SISTEMAS DE INFORMACION.

AUTORA: ING. CLAUDIA MARINA ESPINOZA LEÓN

DIRECTOR: ING. CLAUDIO CRESPO MERCHÁN MSC.

OCTUBRE DE 2010

CUENCA-ECUADOR

El agradecimiento es para al Señor creador de todas las maravillas del universo, por inspirar mis sueños, construirlos y transformarlos en una hermosa realidad.

Gracias a mi amado esposo y a mis preciosas hijas por su paciencia, su apoyo y cariño recibido en todo este tiempo.

Gracias al personal del Departamento de Desarrollo Informático de la Universidad de Cuenca por la colaboración recibida, así como a mi Director de Tesis un sincero agradecimiento por su puntual revisión y ánimo proporcionado en circunstancias decisivas, finalmente muchas gracias a las personas que colaboraron como parte del tribunal, sus valiosas sugerencias ayudaron a enriquecer el presente trabajo.

RESPONSABILIDAD

EL CONTENIDO DE ESTA TESIS ES DE ABSOLUTA RESPONSABILIDAD DEL AUTOR.

_____

INTRODUCCIÓN

ANTECEDENTES

En algún punto del crecimiento de una organización u empresa, en relación a las soluciones informáticas que proveen de sistemas para el soporte de los procesos de negocios, y cuando el tamaño creciente de la empresa y su complejidad en cantidad y profundidad de procesos lo justifica; hay que dar el salto al concepto de arquitectura empresarial. (Urrutia, 2006) con el objetivo de conseguir integrar y reutilizar la infraestructura disponible, para que funcione de manera flexible y aportando más valor, de forma que los procesos de negocio y servicios de la universidad esté alineado con el catálogo de aplicaciones y sistemas del área de TI.

Cada vez es más frecuente que las aplicaciones requeridas deban responder a necesidades que hace pocos años eran muy raras o no existían: ser independientes de los sistemas operativos y gestores de bases de datos; ser accedidas desde lugares geográficamente distantes; interactuar con otros sistemas ya en funcionamiento; atender grandes volúmenes de interacciones por unidad de tiempo, originadas por usuarios que utilizan la aplicación simultáneamente entre otras necesidades. Así, las aplicaciones desarrolladas han transitado de un enfoque centralizado y monolítico hacia entornos distribuidos y heterogéneos; que ha traído un aumento de la complejidad de las aplicaciones desde el punto de vista estructural. (Romero, 2005)

Si el caso es que las funciones comunes que se necesitan son parte de aplicaciones nuevas se empieza a desarrollar desde cero, entonces la solución se muestra fácil, con el inconveniente de requerir un considerable tiempo de desarrollo. Sin embargo, cuando son parte de sistemas existentes y en explotación, al arquitecto de software le quedan dos caminos: modificar los sistemas viejos en aras de reutilizar el código existente; o por otra parte reimplementar la programación existente, con las consecuencias de funcionalidad replicada y/o dificultad de migración de los sistemas internos

puesto que pueden existir múltiples conexiones desde sistemas que dependen de estos para su funcionamiento.

En las entidades de educación existen diversos sistemas que deben ser accedidos en red, a través de diversas plataformas y cuya construcción depende de diversas técnicas que se combinan para producir estructuras complejas que en la mayoría de casos no pudieron ser creados para interactuar entre ellas, lo que ha originado problemas ocasionados por los sistemas que fueron desarrollados para una necesidad específica pero que con el tiempo no pudieron satisfacer las necesidades de interconectividad con su entorno.

Al no haber una estrategia de integración de aplicaciones, se generan múltiples puntos de falla, que pueden detener la operación de todos los sistemas muy fácilmente; un modelo así, por lo general, no escala muy bien. El inconveniente final es una pobre respuesta al cambio. Entonces la solución de duplicar el código encargado de estas aplicaciones se vuelve impensable para aplicaciones grandes, que aumentarían su tamaño, disminuyendo la productividad en el desarrollo, perdiéndose la consistencia y al ser desplegada la aplicación, se haría inmanejable su mantenimiento. (Romero, 2005)

La intervención necesaria para comunicar sistemas informáticos se la hace de una de las formas: la primera es la de volver a introducir a mano los datos entregados por un sistema en otro distinto. La segunda es programando un sistema especialmente diseñado para permitir la transferencia de información entre dos aplicaciones incompatibles, originando interfaces desarrolladas sin políticas claras, de alto impacto en otras aplicaciones cuando existen cambios. Ambas soluciones son costosas e ineficientes debido en el primer caso a que se trata de un esfuerzo de duplicación de datos donde el error humano es frecuente, y en el segundo debido a que para conseguir un buen nivel de interoperabilidad e integración la solución desarrollada resulta complicada y porque su mantenimiento posterior es una constante fuente de problemas: cualquier cambio en alguna aplicación puede hacer fallar la interfaz de transferencia e impedir la comunicación entre ellas, necesitando efectuar cambios e incluso volver a programar en todos los sistemas involucrados.

Muchos de los sistemas desarrollados en la Universidad de Cuenca, carecen de interoperabilidad y para lograr que la información se mueva a través de sistemas distintos, dentro de las fronteras de la organización o a través de ellas es necesaria la intervención humana, por citar un ejemplo: el sistema de recolección de faltas de los docentes debería utilizar un módulo accesible desde diferentes facultades universitarias y estar integrado dentro del sistema de personal, sin embargo el reporte de faltas prácticamente se lo realiza de forma manual puesto que se notifican las faltas por escrito y manualmente se ingresa en el sistema informático para la generación del rol de pagos; trayendo como consecuencia la imposibilidad de presentar información de forma sencilla y útil a los empleados, estudiantes y directivos de una manera ágil, coherente y sistemática.

NECESIDAD A SER SATISFECHA

En muchos centros de cómputo existe una situación frecuentemente presentada: costos incrementados, usuarios insatisfechos, tiempos no cumplidos, alta rotación de personal, baja productividad, clima organizacional problemático; se tiene que lidiar con la existencia de una infraestructura tecnológica diversa, presupuesto insuficiente para el área de TI y una demanda constante de agilidad para soportar los procesos de la organización que están en cambio continúo. Además de que puede encontrarse en la organización que existen procesos poco definidos o definidos ambiguamente, difíciles de planificar y controlar, se encuentra programas defectuosos, que ocupan mucho tiempo en corregir errores, es decir proyectos que no cumplen tiempos, costos y calidad respecto de su estimación inicial.

La falta de planificación y de una metodología claramente establecida, limita la integración entre los componentes de TI: sistemas, aplicaciones y datos, hace difícil obtener una respuesta rápida y efectiva ante los cambios que afectan de forma natural a la Universidad, se generan costos debido a la inflexibilidad y ante las necesidades se reduce la capacidad de respuesta afectando negativamente a la productividad universitaria; **el problema con el que se trata en la presente tesis es la falta de un proceso metodológico que permita integrar sistemas, por lo que se trata de definir una metodología que permita que a futuro los sistemas del Departamento Informático de la**

Universidad de Cuenca puedan ser más fácilmente integrados a través de un método que le permita la integración de sus sistemas, con lo que se conseguirá:

- Incremento en la productividad, pues se evitan estudios, planteamientos no replicables o innecesarios
- Mejora en el tiempo de entrega de soluciones informáticas al disponer de una metodología que se orienta a reutilizar procesos existentes
- Incremento de la satisfacción de los usuarios finales
- Optimización de los tiempos empleados por los desarrolladores al emplear servicios web.
- Ahorro de recursos e incremento del retorno de la inversión

JUSTIFICACION DEL PROYECTO DE TESIS

Con la presente tesis se intenta liberar el potencial de las aplicaciones y recursos de TI para hacerlo disponible de forma general a toda la Universidad con una herramienta como SOA (Service Oriented Architecture), basada en estándares para integrar sistemas y aplicaciones heterogéneas. La intención es plantear un método que lleve a lograr un nivel de integración que facilite los cambios que puedan surgir como respuesta a la evolución de las necesidades de la Universidad de Cuenca, favoreciendo la optimización de procesos que finalmente resultará en mejora de la agilidad Universitaria.

El campo de las Arquitecturas de Software y los sólidos conocimientos en esta área permiten obtener la visión necesaria para diseñar y orientar sistemas informáticos de nueva generación, que estén enfocados hacia la solución de macro problemas que impactan en varios sistemas intra y extra Universitarios dando soluciones orientadas al negocio como es el objetivo de la arquitectura SOA, que permite el desarrollo de una nueva generación de aplicaciones dinámicas que resuelven una gran cantidad de problemas de alto nivel, fundamentales para el crecimiento y la competitividad. Un método de desarrollo orientado a SOA, al ser bien aplicado permite optimizar los recursos de TI de forma más eficiente, ayuda a la organización a optimizar sus procesos internos y sus flujos de información para mejorar la productividad.

La aparición de SOA es una reacción natural frente a las aplicaciones monolíticas y de costo elevado, el enfoque de SOA supone toda una metodología de diseño capaz de alinear la infraestructura TI con los procesos de negocio, sobre la base de servicios compartidos en red. SOA presenta una nueva forma de afrontar los procesos de integración en donde la clave no es desarrollar una aplicación única que implemente absolutamente todos los sistemas necesarios en la universidad, sino crear aplicaciones que sean interoperables y compatibles; incluso aplicaciones existentes se pueden ir transformando en servicios para ser ensamblados.

La metodología propuesta se basa en la arquitectura SOA, y se la hace mediante servicios Web, que al ser diseñados independientes, autónomos, con capacidad de interconexión adecuada, permitirán:

- Utilización en una variedad de escenarios de integración
- Intercomunicación entre sistemas de diversas plataformas
- Descomposición de aplicaciones compactas
- Implementación de funcionalidades de forma modular
- Capacidad de combinación en aplicaciones complejas que respondan a las necesidades de cada momento.

Estos factores ayudan a solventar algunos de los problemas más importantes a los se deben hacer frente para garantizar el crecimiento tecnológico en la Universidad de Cuenca.

OBJETIVOS DE TESIS

Objetivo General

Plantear un método de integración y desarrollo de software hacia una Arquitectura Orientada a Servicios para el Departamento Informático de la Universidad de Cuenca.

Objetivos Específicos

- Hacer un levantamiento de los sistemas y un análisis de arquitecturas a cargo del Departamento de Desarrollo Informático.

- Definición del conjunto de servicios que permita migrar las aplicaciones actuales a una integración SOA.
- Analizar e Integrar una metodología que soporte una Arquitectura Orientada a Servicios para los nuevos sistemas desarrollados dentro del Departamento de Desarrollo Informático.

ALCANCE DEL PROYECTO

El desarrollo del presente trabajo de tesis será efectuado para el DDI (Departamento de Desarrollo Informático de la Universidad de Cuenca) y se limita al estudio de los sistemas que están a cargo de esta dependencia, desarrollados hasta el mes de noviembre de 2008, fecha en la que arranca la presente tesis.

Se incluye dentro del alcance del proyecto el análisis de los sistemas y la definición de los servicios necesarios para conseguir su integración. Para los sistemas a ser desarrollados posteriores a la presente tesis, se planteará una metodología de desarrollo de software que será el resultado de una integración de las mejores prácticas y actividades tomadas de metodologías existentes y utilizadas dentro del DDI.

Los pasos a seguir para el desarrollo de la presente tesis será:

- Se arrancará realizando un análisis del contexto en el que se desenvuelve el DDI y la manera en la que se lleva el proceso de desarrollo de software. Se procederá a inventariar los sistemas a cargo del Departamento de Desarrollo Informático y revisar las arquitecturas que han sido utilizadas para los sistemas.
- Posteriormente se planteará el bloque de servicios que permitirán llegar a un método de integración bajo la arquitectura SOA.
- Finalmente se llegará a plantear una metodología que hará posible que los nuevos sistemas desarrollados en el DDI se orienten hacia una Arquitectura SOA

Capítulo 1.

Marco Teórico

Antes de proceder con el desarrollo del trabajo de tesis, es necesario presentar el marco general que define las tecnologías que intervienen en el presente trabajo y que posteriormente serán utilizadas en el estudio detallado de los sistemas desarrollados para el DDI, en la integración de los sistemas ya construidos y en el planteamiento de la metodología para los nuevos sistemas a desarrollar en el departamento informático; por lo tanto el presente marco teórico se divide en tres secciones:

- Parte 1. Arquitecturas de Sistemas
- Parte 2. SOA
- Parte 3. Metodologías de Desarrollo de Software

1.1. Arquitecturas de Software

Un sistema de software grande y complejo requiere una arquitectura para que los desarrolladores puedan progresar hasta tener una visión común, ya sea para comprender el sistema, organizar el desarrollo, fomentar la reutilización, hacer evolucionar el sistema, en este capítulo se trata de explicar lo que comprende este concepto.

El tema de las arquitecturas de software será utilizado posteriormente en el capítulo tres, debido a que en el Departamento Informático de la Universidad de Cuenca, hay una diversidad de arquitecturas con las que se desarrollan los sistemas, y se precisa hacer una definición de los temas necesarios a conocer para determinar y plantear una arquitectura de un sistema. Posteriormente se describe el tema de los estilos arquitectónicos, que serán necesarios

conocerlos para posteriormente detallar que características tienen cada sistema y clasificarlos en ciertas arquitecturas.

A continuación se resume las etapas recorridas por la Arquitectura de Software.

1.1.1. Recorrido Histórico

Si bien la Arquitectura de Software acostumbra remontar sus antecedentes hasta la década de 1960, su historia no ha sido tan continua como la del campo más amplio en el que se inscribe, la ingeniería de software, las tempranas inspiraciones las dieron el legendario Edsger Dijkstra, David Parnas y Fred Brooks; pero su expansión explosiva se presentó en los manifiestos de Dewayne Perry de AT&T Bell Laboratories de New Jersey y Alexander Wolf de la Universidad de Colorado; sin embargo puede decirse que Perry y Wolf fundaron la disciplina. (Reynoso, 2006)

En un principio, hacia 1968, Edsger Dijkstra, de la Universidad Tecnológica de Eindhoven en Holanda, propuso que se establezca una estructuración correcta de los sistemas de software antes de lanzarse a programar, escribiendo código de cualquier manera; Dijkstra sostenía que las ciencias de la computación eran una rama aplicada de las matemáticas y sugería seguir pasos formales para descomponer problemas mayores como en los sistemas operativos organizados en capas que se comunican sólo con las capas adyacentes y que se superponen “como capas de cebolla”.

P. I. Sharp en la conferencia de la NATO de 1969, un año después de la sesión en que se fundara la ingeniería de software y convencido que en pocos años se hablaría de “la escuela de arquitectura de software de Dijkstra” hace la siguiente apreciación: “La razón de que el sistema operativo OS/360 sea un amontonamiento amorfo de programas es que no tuvo arquitecto. Su diseño fue delegado a series de grupos de ingenieros, cada uno de los cuales inventó su propia arquitectura. Sólo hablamos sobre lo que queremos que haga el programa. En mi opinión, cualquiera que sea responsable de la implementación de una pieza de software debe especificar más que esto. Debe especificar el diseño, la forma y dentro de ese marco de referencia los programadores e ingenieros deben crear algo” (Reynoso, 2006)

En 1972, Parnas publicó un ensayo en el que discutía la forma en que la modularidad en el diseño de sistemas podía mejorar la flexibilidad y el control conceptual del sistema, acortando los tiempos de desarrollo. Introdujo entonces el concepto de ocultamiento de información (information hiding), uno de los principios de diseño fundamentales en diseño de software aún en la actualidad (Parnas, 1972). El pensamiento de Parnas sobre familias de programas, anticipa ideas que luego habrían de desarrollarse a propósito de los estilos de arquitectura: “Una familia de programas es un conjunto de programas a los cuales es provechoso o útil considerar como grupo”. Esta familia de programas puede enumerarse mediante la especificación del árbol de decisión que se atraviesa para llegar a cada miembro de la familia. Las hojas del árbol representan sistemas ejecutables. El concepto también soporta la noción de derivar varios miembros de la familia de un punto de decisión común, aclarando la semejanza y las diferencias entre ellos, **“la elección de la estructura correcta”** es la expresión que sintetiza la Arquitectura de Software.

En 1975, Brooks utilizaba el concepto de arquitectura del sistema para designar “la especificación completa y detallada de la interfaz de usuario” y consideraba que el arquitecto es un agente del usuario, igual que lo es quien diseña su casa identificando y razonando sobre las estructuras de alto nivel, citando la importancia de las decisiones tomadas a ese nivel de diseño. Distingue entre arquitectura e implementación; mientras la arquitectura dice qué hacer, la implementación se ocupa de cómo. (Reynoso, 2006)

El primer estudio en que aparece la expresión “arquitectura de software” en el sentido en que hoy lo conocemos pertenece a Perry y Wolf; ocurrido en 1992, quienes proponen concebir la AS(Arquitectura de Software) por analogía con la arquitectura de edificios (Perry, y otros, 1992). El artículo dice: “El propósito de este artículo es construir el fundamento para la arquitectura de software. Primero desarrollaremos una intuición para la arquitectura de software recurriendo a diversas disciplinas arquitectónicas bien definidas. Sobre la base de esa intuición, presentamos un modelo para la arquitectura de software que consiste en tres componentes: elementos, forma y razón. Los *elementos* son de procesamiento, datos o conexión. La *forma* se define en términos de las propiedades y de relaciones entre los elementos, es decir, restricciones

operadas sobre ellos. La *razón* proporciona una base subyacente para la arquitectura en términos de las restricciones del sistema, que lo más frecuente es que se deriven de los requerimientos del sistema. Discutimos los componentes del modelo en el contexto tanto de la arquitectura como de los estilos arquitectónicos....".

Por lo que se podría resumir a la AS de la siguiente manera:

Arquitectura de Software = { Elementos, Formas,
Configuraciones/Restricciones}

Al definir a la arquitectura de software hay que considerar como un sistema formado por elementos estructurales que comprenden componentes de software, las propiedades de esos componentes y sus relaciones, las configuraciones y las restricciones.

ELEMENTOS (Componentes).- Representan elementos computacionales primarios de un sistema. Los componentes pueden exponer varias interfaces, las cuales definen puntos de interacción entre un componente y su entorno. Corresponden a las cajas de las descripciones de caja-y-línea de las arquitecturas. Ejemplos: clientes, servidores, filtros, objetos, pizarras y bases de datos. Hay que aclarar que la idea de "componente" no es la de la correspondiente tecnología de desarrollo (COM, CORBA Component Model, EJB) más bien es una entidad, a la que los arquitectos prefieren llamar "componente" antes que "objeto".

CONECTORES.- Representan interacciones entre elementos. Los conectores tienen una especie de interfaz que define los roles entre los componentes participantes en la interacción. Son ejemplos: tuberías (pipes), llamadas a procedimientos, broadcast de eventos, protocolos cliente-servidor, o conexiones entre una aplicación y un servidor de base de datos.

CONFIGURACIONES.- Se constituyen como grafos de componentes y conectores. Los sistemas pueden ser jerarquías entre componentes y conectores para simplificar lo que en realidad son complejos subsistemas; también representan propiedades no funcionales, por ejemplo de rendimiento y

latencia probables, cuestiones de seguridad, escalabilidad, dependencia de bibliotecas o servicios específicos, configuraciones mínimas de hardware y tolerancia a fallas.

RESTRICCIONES.- Representan condiciones de diseño que deben acatarse incluso en el caso que el sistema evolucione en el tiempo. Restricciones típicas serían restricciones en los valores posibles de propiedades o en las configuraciones topológicas admisibles. Por ejemplo, el número de clientes que se puede conectar simultáneamente a un servicio.

Dando cumplimiento a lo que anticiparon Perry y Wolf, la década de 1990 fue la de la consolidación de la Arquitectura de Software en una escala sin precedentes. Las contribuciones más importantes surgieron en torno al instituto de ingeniería de la información de la Universidad Carnegie Mellon (CMU SEI). En la literatura se cita constantemente la historia de la especialidad a los planteamientos que hacen a inicios de los años 90, Mary Shaw y David Garlan de la CMU, (Garlan, y otros, 1994) en donde plantean que debido al aumento de tamaño y complejidad de los productos de software, el problema principal no radica en los algoritmos o estructuras de datos sino en la organización de los componentes que conforman el sistema, introduciendo la necesidad de la Arquitectura de Software como una disciplina científica, cuyo objeto de estudio no es más que la determinación de un conjunto de paradigmas que establecen una organización del sistema a alto nivel, la interrelación de los distintos componentes que lo conforman y los principios que orientan su diseño y evolución.

La arquitectura de sistemas comprende información sobre como los componentes de un sistema interactúan entre sí, enfocándose en el lado público de estos componentes, producido por la interface, el lado privado, con sus detalles de implementación no es arquitectura.

1.1.2. Definiciones de Arquitectura

Para sintetizar los conceptos expuestos en las líneas anteriores, se cita algunas de las tantas definiciones tomadas del portal del Instituto de Ingeniería

de Software de la Universidad Carnegie Mellon www.sei.cmu.edu/architecture/definitions.html

Definición Clásica:

> Una arquitectura es el conjunto de decisiones significativas sobre la organización de un sistema de software, la selección de elementos estructurales y sus interfaces por las cuales el sistema estará compuesto, junto a las colaboraciones entre esos elementos, la composición de esos elementos estructurales y su comportamiento dentro de subsistemas progresivamente más grandes y el estilo arquitectónico que guía a la organización, esos elementos y sus interfaces, sus colaboraciones y su composición. (Booch, y otros, 1999)

Definición Moderna:

> La arquitectura de software de un programa o sistema de computación, es la estructura de estructuras del sistema, que comprende elementos de software, las propiedades externamente visibles de esos elementos y las relaciones entre estos. (Bass, y otros, 2003)

Las diferencias que existen entre la abstracción cualitativa de la arquitectura y las cuantificaciones que han sido la norma en ingeniería de software se definen en las siguientes normas:

Según IEEE 1471-2000

> **La Arquitectura de Software** es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución.

Según IEEE 610.12.1990

> La aplicación de la **ingeniería al software** es el empleo de una estrategia sistemática, disciplinada y cuantificable al desarrollo, aplicación y mantenimiento del software.

1.1.3. Marcos y Vistas Arquitectónicas

Para plantear el modelo arquitectónico de las aplicaciones existentes en el departamento de desarrollo informático de la Universidad de Cuenca y expuesto más adelante en el punto 3.3.3 y en el capítulo 5, se necesita aclarar los conceptos relacionados a los Marcos y Vistas arquitectónicas, mediante los cuales se expresa una arquitectura de software.

Existen unos cuantos organismos de estándares (ISO, CEN, IEEE, OMG) que han codificado diferentes aspectos de la AS, con el objetivo de homogeneizar la terminología, los modelos y los procedimientos. Los emergentes del trabajo de sus comités son especificaciones y recomendaciones de variada naturaleza, como RM-ODP, RUP, RDS, MDA, MOF, MEMO, XMI o IEEE 1471-2000 (Reynoso, 2006)

Es muy complejo capturar la arquitectura de software en un solo modelo o diagrama. Para manejar esta complejidad se representan diferentes aspectos y características de la arquitectura en múltiples vistas. Una vista es una representación de un modelo, una descripción completa de un sistema desde una particular perspectiva (Kruchten, 1995). Es un subconjunto resultante de practicar una selección o abstracción sobre una realidad, desde un punto de vista determinado.

Tanto los marcos arquitectónicos como las metodologías de modelado de los organismos acostumbran ordenar las diferentes perspectivas de una arquitectura en términos de vistas. Cada paradigma de desarrollo de software exige diferente número y tipo de vistas o modelos para describir una arquitectura. La expresión de una arquitectura se constituye de vistas, que constituyen la representación de un conjunto de elementos del sistema y sus relaciones. No obstante, existen al menos tres vistas absolutamente fundamentales en cualquier arquitectura:

- La visión **estática:** describe qué componentes tiene la arquitectura.
- La visión **funcional**: describe qué hace cada componente.
- La visión **dinámica:** describe cómo se comportan los componentes a lo largo del tiempo y cómo interactúan entre sí.

Las vistas o modelos de una arquitectura pueden expresarse mediante uno o varios lenguajes, el más obvio es el lenguaje natural, pero existen otros lenguajes como diagramas de estado, diagramas de flujo de datos, etc. Existe cierto consenso en adoptar UML (Unified Modeling Language, lenguaje unificado de modelado) como lenguaje único para todos los modelos o vistas.

La mayoría de los frameworks y estrategias reconoce entre tres y seis vistas, que son las que se incluyen en el cuadro expuesto en (Reynoso, 2006)

Zachman (Niveles)	TOGAF (Arquitecturas)	4+1 (Vistas)	[UML] (Vistas)	POSA (Vistas)	Microsoft (Vistas)
Scope	Negocios	Lógica	Diseño	Lógica	Lógica
Empresa	Datos	Proceso	Proceso	Proceso	Conceptual
Sistema lógico	Aplicación	Física	Implementación	Física	Física
Tecnología	Tecnología	Desarrollo	Despliegue	Desarrollo	
Representación		Casos de uso	Casos de uso		
Funcionamiento					

Tabla 1. Vistas en los principales marcos de Referencia (Reynoso, 2006)

Las vistas se introdujeron como una herramienta conceptual para manejar la complejidad de lo que ya por aquel entonces se llamaban artefactos, tales como especificaciones de requerimientos o modelos de diseño. (Reynoso, 2006).

La AS clásica se funda en una vista singular e implícita, de carácter estructural. Muchos arquitectos de la corriente principal evitan hablar de vistas, porque cuando proliferan se hace necesario o bien elaborar lenguajes formales específicos para tratar cada una de ellas, o bien multiplicar las extensiones del lenguaje unificado. Sin duda las vistas son una simplificación conveniente; pero su abundancia y sus complicadas relaciones recíprocas generan también nuevos órdenes de complejidad. (Reynoso, 2006)

No hay un límite necesario para el número de vistas posibles, ni un procedimiento formal para establecer lo que una vista debe o no abstraer. El estándar IEEE 1471 no delimita el número posible de vistas, ya que se estima que no puede haber acuerdo en ello, pero señala lineamientos para su constitución y considera que un punto de vista es a una vista como una clase es a un objeto. Hay una “lista corta” de vistas que se usa en los textos generales de AS y una “lista larga” que gira en torno de UML, el cual especifica nueve clases de diagramas correspondientes a ocho vistas, como se indica en el siguiente cuadro tomado de (Reynoso, 2006).

Área	Vista	Diagramas	Conceptos principales
Estructural	Vista estática	Diagrama de clases	Clase, asociación, generalización, dependencia, realización, interfaz
	Vista de casos de uso	Diagramas de casos de uso	Caso de uso, actor, asociación, extensión, inclusión, generalización de casos de uso
	Vista de implementación	Diagrama de componentes	Componente, interfaz, dependencia, realización
	Vista de despliegue	Diagrama de despliegue	Nodo, componente, dependencia, localización

Área	Vista	Diagramas	Conceptos principales
Dinámica	Vista de máquinas de estados Vista de actividad Vista de interacción	Diagrama de estados Diagrama de actividad Diagrama de secuencia Diagrama de colaboración	Estado, evento, transición, acción Estado, actividad, transición de terminación, división, unión Interacción, objeto, mensaje, activación Colaboración, interacción, rol de colaboración, mensaje
Gestión del modelo	Vista de gestión del modelo	Diagrama de clases	Paquete, subsistema, modelo

Tabla 2. Vistas y Diagramas UML

Toda arquitectura de software describe diversos aspectos, cada uno de estos se detalla de una manera más comprensible si se utilizan distintos modelos o vistas. Cada aspecto constituye una descripción parcial de una misma arquitectura, existiendo cierto solapamiento entre ellos, pues todas las vistas deben ser coherentes entre sí ya que describen la misma cosa. “Como un edificio, un sistema de software es una única entidad, pero al arquitecto del software y a los desarrolladores les resulta útil presentar el sistema desde diferentes perspectivas para comprender mejor el diseño. Estas perspectivas son vistas del modelo del sistema. Todas juntas representan la arquitectura” (Jacobson, 99).

Partiendo de que un modelo es una representación simplificada de la realidad, orientada a comprender de mejor manera la misma; el modelo seleccionado debe describir al menos las piezas mayores y decisiones claves globales, que es lo planteado en el capítulo 5 con la metodología que se propone y se denominará RASDUC (metodología **R**up **A**gil orientada a **S**oa para el departamento de **D**esarrollo Informático de la **U**niversidad de **C**uenca).

1.1.4. Estilos arquitectónicos

En esta tesis hacemos un estudio global de los estilos arquitectónicos para posteriormente en el punto 3.1.3 utilizar este tema para realizar el análisis de los estilos arquitectónicos utilizados en el DDI y proceder la clasificación de los sistemas (punto 3.1.3.2) en alguno de los estilos arquitectónicos que a continuación se detallan.

Un estilo arquitectónico se define en términos de un patrón de la estructura de una organización, determina el vocabulario de componentes y conectores que

pueden ser utilizados junto a restricciones de cómo deben combinarse. (Garlan, y otros, 1994).

Un estilo es un concepto descriptivo que define una forma de articulación u organización arquitectónica. El conjunto de los estilos cataloga las formas básicas posibles de estructuras de software, mientras que las formas complejas se articulan mediante composición de los estilos fundamentales. (Perry, y otros, 1992).

La descripción de un estilo arquitectónico, se puede formular en lenguaje natural o en diagramas, pero lo mejor es hacerlo en un lenguaje de descripción arquitectónica o en lenguajes formales de especificación. David Garlan and Mary Shaw citan estilos arquitectónicos como Pipes y Filtros, Abstracción de datos y orientación a objetos, invocación implícita basada en eventos, sistemas de capas, sistemas basados en repositorios, proceso distribuido y su forma más común cliente/servidor, organización por subrutinas, arquitecturas específicas a un dominio de software, sistemas de transición de estados y sistemas de control de procesos. (Garlan, y otros, 1994)

Clasificación de Estilos Arquitectónicos	
Estilos de Flujo de Datos	Tubería y filtros Secuencial por lotes
Estilos Centrados en Datos	Bases de Datos Arquitecturas de Pizarra o Repositorio Sistemas de HiperTexto(Modelo REST)
Estilos de Llamada y Retorno	Sistema Modular Model-View-Controller (MVC) Arquitecturas en Capas Arquitecturas Orientadas a Objetos Arquitecturas Basadas en Componentes
Estilos de Código Móvil	Arquitectura de Máquinas Virtuales
Estilos Peer-to-Peer	Arquitecturas Basadas en Eventos Arquitecturas Orientadas a Servicios Arquitecturas Basadas en Recursos
Estilos heterogéneos	Sistemas de control de procesos Arquitecturas Basadas en Atributos

Tabla 3 Clasificación general de estilos arquitectónicos

En la tabla anterior podemos observar la clasificación general de estilos arquitectónicos *(tomado de la presentación de Vallecillo y Monge, "Desarrollo de aplicaciones basado en Componentes y Frameworks").* De esta clasificación, a continuación se describen los principales estilos arquitectónicos y la arquitectura orientada a servicios que es el objetivo principal de la tesis, la cual será descrita en la siguiente sección 1.2

Arquitectura Filtro/tubería

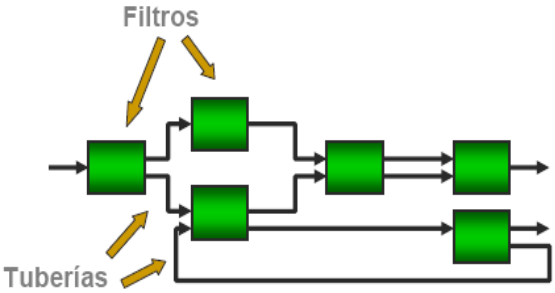


Figura 1. Arquitectura Filtro/Tubería

- Cada componente o filtro tiene un conjunto de entradas y un conjunto de salidas. Cada filtro lee flujos de datos en sus entradas y produce flujos de datos en sus salidas. Un componente lee entradas y las transforma en salidas.
- Los conectores o tubos transmiten la salida de un filtro como entrada de otro, no efectúan ningún otro procesamiento visible.
- Las restricciones de este tipo de arquitectura son: los filtros deben ser independientes, no deben compartir estado con otros filtros, no conocen la identidad de sus continuadores o predecesores y los filtros realizan la labor independientemente del flujo de entrada.

Arquitectura de Pizarra

Existen dos tipos de componentes

- Una estructura central de datos (representa el estado del proceso)
- Componentes independientes (operan en función del depósito de datos)
- Las interacciones entre el repositorio y los demás componentes es variable:
    - La entrada de los datos es seleccionada por los componentes

- El estado de los datos del repositorio selecciona el proceso a

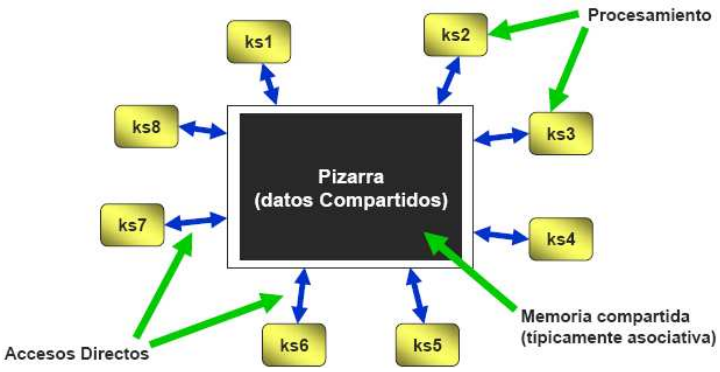


Figura 2. Arquitectura de Pizarra

ejecutar (pizarra)

Arquitectura Basada en Eventos

- Un componente anuncia un evento. Otros registran interés en ese tipo de evento. Cuando se produce, el sistema lo comunica a los suscriptores.
- Los componentes son los módulos cuyas interfaces ofrecen un conjunto de procedimientos y de eventos. En lugar de invocaciones a procedimientos explícitas o directas, un componente anuncia uno o más eventos y otros componentes registran el interés en un evento asociando un procedimiento a dicho evento.
- Los conectores incluyen llamadas a procedimientos tradicionales, así como la ligadura de eventos con llamadas a procedimientos. La ocurrencia de un evento causa la invocación "implícita"de procedimientos en otros módulos.

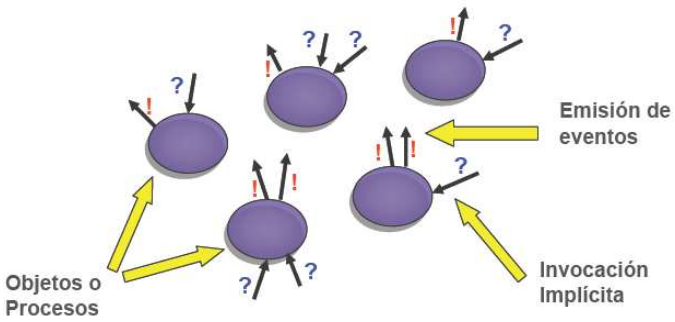


Figura 3. Arquitectura Basada en Eventos

- Entre las restricciones está que quien anuncia el evento no conoce a qué componentes afecta éste y no se puede determinar el orden de procesamiento

Sistemas Basados en Capas

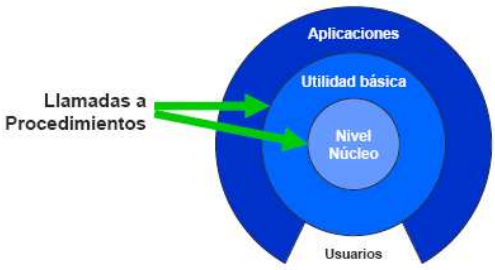
- Organizado jerárquicamente en capas, donde cada capa provee servicios a la capa superior y es servido por la capa inferior
- Los componentes son cada una de las capas
- Los conectores son los protocolos de interacción entre las capas
- Restricciones, la interacción está limitada a las capas adyacentes

Figura 4 Arquitectura en Capas

1.2. Arquitectura Orientada a Servicios. SOA (Service Oriented Architecture)

1.2.1. Definición de SOA

El concepto de lo que hoy se conoce como Arquitectura SOA tiene su origen en las últimas décadas, SOA no se deriva de una propuesta académica, de hecho no hay reportes técnicos de SOA en el SEI (Instituto de Ingeniería de Software en la Universidad Carnegie Mellon), la arquitectura orientada a servicios fue descrita por la primera vez por Gartner en 1996, en los documentos: SSA Research Note SPA-401-068, "'Service Oriented' Architectures, Part 1" y SSA Research Note SPA-401-069, "'Service Oriented' Architectures, Part 2". La Arquitectura Orientada a Servicios supone una estrategia general de organización de los elementos de IT, de forma que una colección heterogénea de sistemas distribuidos y aplicaciones complejas se pueda transformar en una red de recursos integrados, simplificada y sumamente flexible. (Microsoft Corporation, 2006)



La Arquitectura Orientada a Servicios, según (Krafzig, y otros, 2004) es un estilo de Arquitectura de Software basado en la definición de servicios reutilizables con interfaces públicas bien definidas, donde proveedores y

consumidores de servicios interactúan desacopladamente para realizar los procesos del negocio, y donde los servicios se componen en secuencias definidas para realizar los procesos de negocio (orquestación, coreografía). Como meta principal se plantea la reusabilidad e interoperabilidad de las aplicaciones, mediante la definición de servicios que puedan ser reutilizados por la organización y fuera de ésta.

SOA consiste en una forma de modularizar los sistemas y aplicaciones en componentes, que pueden combinarse y recombinarse con interfaces bien definidas para responder a las necesidades de la organización. Supone toda una metodología de diseño capaz de alinear la infraestructura TI con los procesos de negocio sobre la base de servicios compartidos en red. (COMPUTING-ES, 2004)

SOA permite compartir, reutilizar aplicaciones, implementar procesos, mediante la creación de una capa de abstracción que permita separar los procesos de negocio de los servicios necesarios para realizarlos. Los servicios son proporcionados, por las aplicaciones y una vez que están disponibles pueden ser utilizados a la vez que facilitan su modificación posterior. SOA representa un paso en la evolución de la arquitectura software de las organizaciones, en que su modelo de negocio está orientado a procesos cooperativos. Si bien su concepto no es nuevo, puesto que ya se manejaba una abstracción de la arquitectura distribuida de componentes sugerida por CORBA, la creciente popularidad de SOA responde al uso de servicios web como componentes tecnológicos estándar. (Folgado, 2004)

1.2.2. De XML a SOA

SOA es el producto de una evolución, deben su origen al nacimiento del XML, el cual a su vez es un sistema de marcado que permite integrar tanto contenido como estructura de los mismos. Con el surgimiento de XML se vio lo interesante que podría empaquetar las llamadas a RPC. En el año 2000 el W3C recibió una posible especificación que unificaba/reemplazaba las comunicaciones propietarias de los RPC con XML. Las empresas recibieron esta propuesta con las manos abiertas, debido a que se permitía una comunicación “puramente web”, se llamaron “Web Services”; se centraron en la

especificación de la interfaz pública, la parte más importante, con el lenguaje WSDL (Web Service Description Language), posteriormente las empresas vieron que los servicios web podían formar parte de una plataforma arquitectónica distinta y aparece el concepto de lo que hoy se conoce como SOA. (Cortiz,2009)

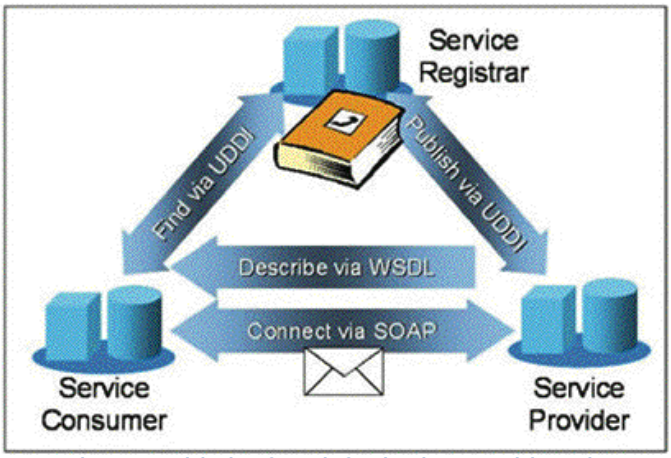


Figura 5. Modelo de referencia funcional para servicios web

"SOA es una arquitectura de aplicación en la cual todas las funciones se definen como servicios independientes con interfaces invocables bien definidas, que pueden ser llamadas en secuencias definidas para formar procesos de negocios" (IBM).

SOA puede ser considerado como un "Paradigma de Arquitectura Corporativa". Así como en su momento, el Paradigma de Orientación a Objetos cambió la forma en que pensamos en términos del análisis, diseño y construcción de aplicaciones, SOA propone un modelo de pensamiento con el cual atacar la definición de arquitecturas, y ya no las de aplicaciones individuales, sino las arquitecturas de organizaciones completas. (Duré, Octavio, 2010)

La principal tecnología detrás de la arquitectura SOA son los Servicios Web, que en pocas palabras son un conjunto de protocolos, procesos e interfaces estandarizadas que permiten separar la interface de los servicios de la implementación; logrando que los servicios sean utilizados por múltiples aplicaciones permitiendo así reutilizar los activos existentes y aprovechar los sistemas actuales. (Arceo, 2006) Los servicios web hacen uso de un conjunto de tecnologías (XML, SOAP, WSDL, UDDI) estándares que en la actualidad

son los utilizados, en el futuro pueden ser reemplazados por otras tecnologías, mientras que el concepto de SOA es considerado como un modelo conceptual.

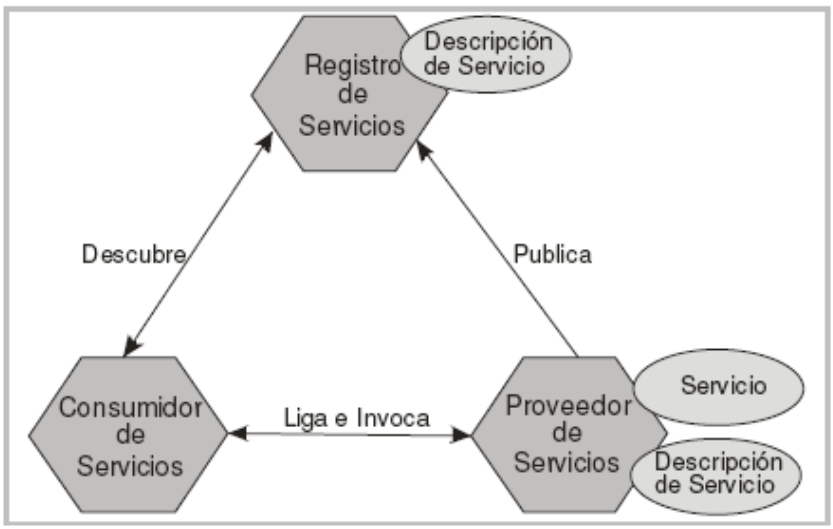


Figura 6. Colaboraciones en SOA (de Alvez, 2006)

Web Services no es sinónimo de SOA. Por el contrario, es posible utilizar Web Services y seguir en un esquema de integración punto a punto, así como es posible implementar un esquema SOA sin utilizar Web Services. Sin embargo, tanto Web Services como los estándares asociados (XML, SOAP, WSDL, UDDI, WS-I) conforman una base de tecnologías y estándares que facilitan la implementación de SOA. (Dure Octavio, 2010)

1.2.3. Servicios Web

1.2.3.1. Utilización de servicios

Al diseñar sistema distribuidos se puede llegar a un enfoque donde los diversos procesos a automatizar han de ser vistos como servicios autónomos que interactúan entre sí a través de mensajes. Cada servicio además de poseer lógica y datos propios, expone claramente la interfaz basada en mensajes para accederlo; constituyendo una aplicación autónoma e independiente que permite la comunicación con otras aplicaciones. Cada aplicación conoce, cuáles servicios son necesarios y cómo deben ser utilizados para ofrecer los resultados esperados a los usuarios finales. La comunicación hacia y desde el servicio, es realizada utilizando mensajes; estos mensajes deben contener o referenciar toda la información necesaria para entenderlo. La idea es que haya el mínimo posible de llamadas entre el cliente y el servicio. (Romero, 2005)

Cuando se usan múltiples servicios para implementar un sistema, es muy fácil que la comunicación entre estos se salga de control. Por ejemplo, si se tiene un servicio que llama a otros seis servicios, alguno de los cuales llama a otros servicios, por lo que muy fácilmente el sistema se vuelve inmanejable. De esta manera, un sistema grande puede terminar con múltiples dependencias. Detectar un problema de rendimiento o funcionalidad se puede volver muy complicado. Una solución lógica a este problema es extraer los aspectos de procedimiento de varios servicios dentro de uno dedicado, llamado servicio de negocio. Un servicio de negocio controla las acciones paso a paso en la ejecución de algún trabajo, moviendo el sistema de un estado a otro. En cada paso, éste llamará una operación de negocio provista por un servicio. Así, un servicio de negocio centraliza la definición del proceso, en vez de tener piezas del proceso entre todos los servicios. Esto disminuye las dependencias entre servicios y las aplicaciones clientes; ayudando a facilitar la administración del sistema.

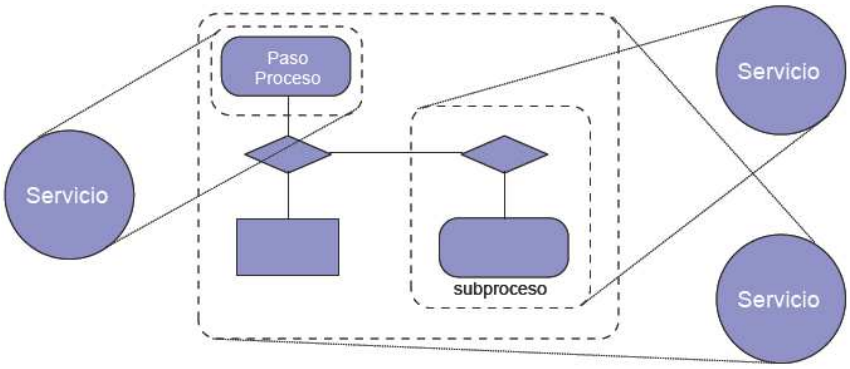


Figura 7 Servicio como un proceso de negocio (fuente Delgado, 2007)

Un servicio es la evolución en complejidad de un componente distribuido, la construcción de un servicio es una tarea mucho más complicada que la de un simple componente distribuido. La respuesta del servicio es afectada directamente por aspectos externos como problemas en la red y configuración por citar algunos. Estos deben ser tenidos en cuenta en el diseño, desarrollándose los mecanismos de contingencia que eviten la parálisis de las aplicaciones y servicios que dependen de él. (Romero, 2005)

1.2.3.2. Estilos de utilización de servicios web

Existen tres estilos de utilización principales para los servicios web:

- Tipo RPC Remote Procedure Calls (Llamadas a Procedimientos Remotos)

Los Servicios Web basados en RPC presentan una interfaz de llamada a procedimientos y funciones distribuidas, la unidad básica de este tipo de servicios es la operación WSDL (WSDL es un descriptor del Servicio Web, es decir, el homologo del IDL[1] para COM). Las primeras herramientas para Servicios Web estaban centradas en esta visión, algunos lo llaman la primera generación de Servicios Web. Esta es la razón por la que este estilo está muy extendido. Sin embargo, ha sido algunas veces criticado por no ser débilmente acoplado, ya que suele ser implementado por medio del mapeo de servicios directamente a funciones específicas del lenguaje o llamadas a métodos. Muchos especialistas creen que este estilo debe desaparecer. (Navarro, 2006)

- Tipo SOA. Arquitectura Orientada a Servicios (Service-oriented Architecture)

Los Servicios Web pueden también ser implementados siguiendo los conceptos de la arquitectura SOA, donde la unidad básica de comunicación es el mensaje, más que la operación. Esto es típicamente referenciado como servicios orientados a mensajes. Los Servicios Web basados en SOA son soportados por la mayor parte de desarrolladores de software y analistas. Al contrario que los Servicios Web basados en RPC, este estilo es débilmente acoplado, lo cual es preferible ya que se centra en el "contrato" proporcionado por el documento WSDL, más que en los detalles de implementación subyacentes. (Navarro, 2006)

- Tipo REST (Representational State Transfer)

En el año 2000 la tesis de Roy Fielding presentó el modelo REST (Representational State Transfer), el cual establece definitivamente el tema de las tecnologías de Internet y los modelos orientados a servicios. Los Servicios Web basados en REST intentan emular al protocolo HTTP o protocolos similares mediante la restricción de establecer la interfaz a un conjunto conocido de operaciones estándar (por ejemplo GET, PUT, DELETE,…). Por tanto, este estilo se centra más en interactuar con recursos con estado, que

[1] IDL Lenguaje de Descripción de Interfaz, se usa para definir las interfaces para acceder y operar sobre objetos

con mensajes y operaciones. El término fue introducido en la tesis doctoral de Roy Fielding en 2000, quien es uno de los principales autores de la especificación de HTTP. En realidad, REST se refiere estrictamente a una colección de principios para el diseño de arquitecturas en red. Estos principios resumen cómo los recursos son definidos y diseccionados. El término frecuentemente es utilizado en el sentido de describir a cualquier interfaz que transmite datos específicos de un domino sobre HTTP sin una capa adicional, como hace SOAP. (Navarro, 2006)

Los principios que guiarán a plantear la arquitectura SOA (Erl, 2005) y que serán empleados en los anexos para determinar el catálogo de servicios, son:

1.2.3.3. Principios según Thomas Erl

Para realizar un análisis para los servicios web, se plantean los siguientes principios que se deberían seguir, tal y como se describe en el punto 3 del anexo Modelo de Servicios

- *Los Servicios deben ser reusables:* Diseñar y construir pensando en la reutilización, dentro de la misma aplicación, del dominio de aplicaciones de la empresa o dentro del dominio público para su uso masivo.
- *Los Servicios deben proporcionar un contrato formal*: Se debe proporcionar un contrato en el cual conste: el nombre del servicio, su forma de acceso, funcionalidades que ofrece, datos de entrada de cada una de las funcionalidades y datos de salida. Todo consumidor del servicio accederá mediante el contrato, logrando independencia entre el consumidor y la implementación del propio servicio. En el caso de los Servicios Web, esto se logrará mediante la definición de interfaces con WSDL.
- *Los Servicios deben tener bajo acoplamiento***:** Los servicios tienen que ser independientes unos de otros. Para lograr ese bajo acoplamiento, cada vez que se vaya a ejecutar un servicio, se accederá a través del contrato, logrando independencia entre el servicio a ejecutar y el que lo llama. Si conseguimos este bajo acoplamiento, entonces los servicios podrán ser totalmente reutilizables.
- *Los Servicios deben de ser autónomos*: Todo Servicio debe tener su propio entorno de ejecución. De esta manera el servicio es totalmente

independiente y podemos asegurar que podrá ser reutilizable desde el punto de vista de la plataforma de ejecución.

- *Los Servicios deben permitir la composición:* Todo servicio debe ser construido de forma que pueda ser utilizado para construir servicios genéricos de más alto nivel, el cual estará compuesto de servicios de más bajo nivel. En el caso de los Servicios Web, esto se logrará mediante el uso de los protocolos para orquestación (WS-BPEL) y coreografía (WS-CDL).
- *Los Servicios no deben tener estado:* Un servicio no debe guardar ningún tipo de información, ya que una aplicación está formada por un conjunto de servicios, si un servicio almacena algún tipo de información, se pueden producir problemas de inconsistencia de datos. La solución, un servicio sólo tendrá lógica, pues toda la información está almacenada en algún sistema de información sea del tipo que sea.
- *Los Servicios deben poder ser descubiertos:* Todo servicio debe ser descubierto para que pueda ser utilizado, consiguiendo así evitar la creación accidental de servicios que proporcionen las mismas funcionalidades. En el caso de Servicios Web, el descubrimiento se lo hará publicando interfaces de servicios en registros UDDI.

1.2.4. Coreografía y Orquestación de Servicios Web

Del modelado de procesos, de la arquitectura orientada a servicios y del uso de web services surge el concepto de composición de servicios, que permite modelar el proceso de negocio maximizando las potencialidades que SOA brinda a través de la integración de datos y aplicaciones. La composición de servicios implica encontrar un mecanismo que permita a dos o más de ellos cooperar entre sí para resolver requerimientos que van más allá del alcance de sus capacidades individuales. (Alvez, y otros, 2006)

Desde el punto de vista tecnológico, SOA propone varias capas de servicios que exponen funcionalidad de negocio, que permiten la composición de aplicaciones a partir de los mismos. En el nivel más bajo, tendremos una capa de servicios de “Acceso a Información y Datos”, cuya función es exponer en un ESB (Enterprise Service Bus) la funcionalidad y facilidades provistas por los sistemas corporativos. Muchas veces, estos servicios básicos no representan realmente “servicios de negocio” que aporten real valor al resto de la

organización, al menos no por sí solos. Es por esto que sobre esta primera capa, encontraremos la segunda capa de servicios que denominamos de “Servicios de Negocio Compartidos” que componen y enriquecen la funcionalidad expuesta en la capa precedente, con el objeto de agregar valor desde el punto de vista del negocio de una organización. Con base en estos servicios básicos y de negocio, es posible comenzar el desarrollo de aplicaciones compuestas. Estos desarrollos son los que aprovechan los beneficios de un esquema SOA en cuanto a reusabilidad de servicios existentes y adaptabilidad al negocio. Sin embargo, es necesario que en esta capa, las aplicaciones también expongan en el Bus de Servicios la funcionalidad que resuelven, para que sean aprovechadas por otras aplicaciones y servicios. . (Dure, 2010)

Existen dos conceptos que abordan el problema de la integración efectiva de procesos de negocio basados en tecnologías web: coreografía y la orquestación de servicios web. Estos conceptos dirigen sus esfuerzos a describir de manera global las interacciones y cambios de estado, así como el flujo interno del proceso. (Peltz, 2003)

Coreografía

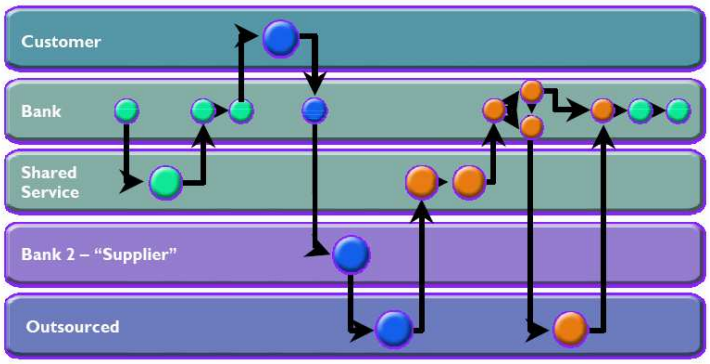


Figura 8. Coreografía de Servicios. Fuente Wikipedia

En algunos casos los requerimientos de información planteados no son logrados a través de un simple servicio web, sino que se hace necesario combinar varios de ellos para generar un servicio que pueda satisfacer los objetivos del usuario. Las interacciones que ocurren entre estos a fin de lograr su cometido es interés de la coreografía, ya que esta describe la secuencia y condiciones en la que los datos son intercambiados por los servicios web. La coreografía de servicios web describe las interacciones pertinentes durante la integración entre los servicios, con el fin de lograr los objetivos propuestos. (Giraldo, 2008)

La coreografía puede entenderse como un proceso público y no ejecutable; es público porque define el comportamiento común y globalmente visible entre los diferentes participantes en una interacción; por otro lado, es no ejecutable porque no está pensado para ser llevado a cabo, sino para actuar como un protocolo de negocio que dicta reglas de interacción que deben ser cumplidas por las entidades participantes (Peltz, 2003).

Orquestación

Un proceso se puede considerar una orquestación de servicios cuando es controlado totalmente por una única entidad, este proceso define completamente las interacciones con los servicios componentes y la lógica requerida para conducir correctamente esas interacciones.

Este proceso puede entenderse como privado y ejecutable ya que solo la entidad que está orquestando el proceso conoce el flujo de control e información que sigue el proceso que se está orquestando. De esta manera se crea un proceso que utiliza diferentes servicios manipulando la información que fluye entre ellos, convirtiendo por ejemplo los datos de salida de algunos servicios en datos de entrada de otro.

Es posible realizar una coreografía mediante un conjunto de orquestaciones de cada uno de los componentes que la integran.

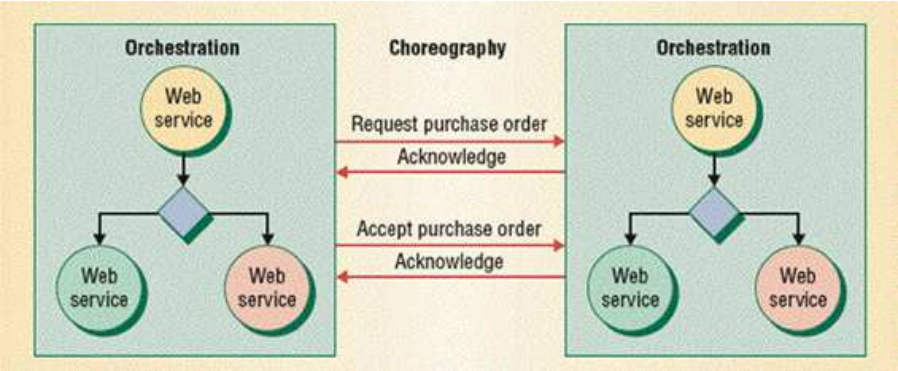


Figura 9. Coreografía vs. Orquestación

En el ámbito industrial hay distintos estándares que ayudan a la descripción de la integración de procesos de negocio; estos abordan tanto la orquestación

como la coreografía; entre ellos se encuentran: XLANG (Microsoft, 2001), WSFL (IBM, 2001), WSCI (W3C, 2001), WSCDL y BPEL4WS

1.2.5. Tecnologías relacionadas a los Web Services

Los Web Services surgen con mayor fuerza hacia el año 2000. El consorcio W3C define los Servicios Web como sistemas software diseñados para soportar una interacción interoperable maquina a máquina sobre una red. Los Servicios Web suelen ser APIs Web que pueden ser accedidas dentro de una red (principalmente Internet) y son ejecutados en el sistema que los aloja. Esta definición de Servicios Web contempla muchos tipos diferentes de sistemas, pero el caso común se refiere a clientes y servidores que se comunican mediante mensajes XML que siguen el estándar SOAP. (Navarro, 2006).

Los Web Services permiten que aplicaciones web interactúen dinámicamente con otras aplicaciones web, utilizando estándares abiertos como XML, UDDI y SOAP. Esto es posible porque están basados en un protocolo simple y liviano para realizar las invocaciones, como SOAP, un protocolo estándar creado por W3C, basado en XML. (Rodríguez, y otros, 2006)

Los Web Services son mucho más que simplemente SOAP, éste último solo define el protocolo de invocación remota a los métodos, incorporan WSDL como un lenguaje basado en XML que permite describir los contratos de cada servicio. Este lenguaje fue desarrollado conjuntamente por Microsoft e IBM y sometido para su aprobación como un estándar. (Rodríguez, y otros, 2006).

Se puede considerar que SOA está compuesto por las tecnologías, cada una de la cual será descrita a continuación:

SOA = XML+SOAP+WSDL+UDDI+Bus

XML

XML (Extensible Markup Language), el cual es un lenguaje de marcas extensible que permite una mejor adaptación y flexibilidad entre sistemas heterogéneos; su característica principal es la adopción de etiquetas en un archivo de texto que permite la interpretación por sistemas en diferentes

plataformas, su principal debilidad es el tamaño que puede alcanzar cualquier archivo creado con este estándar. (Lugo Gonzalez, 2009). Los servicios web incluyen un protocolo que define el formato de un documento XML con la información de descubrimiento y un protocolo para que un usuario pueda recibir este documento, permitiendo descubrir servicios de una URL conocida.

SOAP. Simple Object Access Protocol

Es un protocolo simple para el intercambio de información estructurada en un entorno distribuido y descentralizado. Utiliza XML para definir un framework extensible de mensajería proveyendo un formato de mensaje para que pueda ser intercambiado sobre una variedad de protocolos subyacentes. El framework fue diseñado para ser independiente de cualquier modelo de programación o cualquier semántica específica de alguna implementación (Alvez y otros, 2006). SOAP está compuesto por cuatro componentes fundamentales: un *envoltorio* que define un framework para describir los mensajes y como estos serán procesados, un conjunto de *reglas* para codificar o serializar instancias de tipos de datos definidos por las aplicaciones, una convención de cómo representar *invocaciones* remotas y sus respuestas y una convención para vincular el *intercambio* de mensajes con el protocolo de transporte. El protocolo de transporte que normalmente utiliza SOAP y el utilizado por los web services es http. (Rodríguez, y otros, 2006)

- SOAP 1.0 - Específico de MS+Developmentor. XML + HTTP
- SOAP 1.1 - MS+IBM+Lotus. Bindings de transporte para no-HTTP
- SOAP 1.2 - W3C.org

WSDL

WSDL (Web Services Description Language), el lenguaje de descripción de servicios web nació en septiembre de 2000 de la mano de Microsoft, IBM y Ariba. Se basa en los lenguajes de definición NASSL44(Network Accesible Services Specification) de IBM y SCL45(SOAP Contract LAnguage) de Microsoft. En Marzo de 2001 estas compañías con el

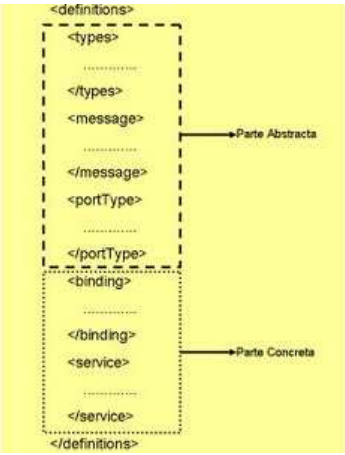


Figura 10 Componentes de un documento WSDL

apoyo de algunas otras enviaron la versión WSDL 1.146 al W3C donde fue publicado como una nota. La especificación original en esencia no ha cambiado. (Alvez, y otros, 2006). Es un estándar utilizado para describir el servicio web, gracias a los archivos descritos en este lenguaje se pueden descubrir los servicios web y utilizarlos de acuerdo a sus características, es un contrato entre el proveedor del servicio y el cliente mediante el cual el proveedor del servicio indica:

- Qué funciones se pueden invocar
- Qué tipos de datos utilizan esas funciones
- Qué protocolo de transporte se utilizará para el envío y recepción de los mensajes
- Cómo acceder a los servicios
- Mediante que URL se utilizan los servicios.

Consta de una parte abstracta y de otra concreta con los componentes que constan en la figura10.

UDDI . Universal Description, Discovery and Integration

Es una tecnología que consiste en un directorio de servicios web en donde se pueden publicar los servicios ofrecidos, dar características del tipo de servicio y realizar búsquedas. (Alvez, y otros, 2006) Para los casos en los que el usuario no conoce la URL, también incluye mecanismos UDDI, estándares para que los proveedores de los servicios publiquen sus servicios y los consumidores los encuentren. El propósito funcional de un registro UDDI es la representación de datos y metadatos a cerca de servicios web. Tanto para ser usado en una red pública como dentro de la infraestructura interna de una organización, un registro UDDI ofrece un mecanismo basado en estándares para clasificar, catalogar y manejar servicios web de forma de que puedan ser descubiertos y consumidos por otras aplicaciones. (Alvez, y otros, 2006). A la hora de publicar los servicios web, existen muchas posibilidades, la más común y mas discutida es la de utilizar UDDI que no es más que una especie de directorio en donde podemos encontrar un servicio web de acuerdo a ciertas características funcionales; una gran debilidad a la hora de descubrir un servicio web es localizarlo solamente por sus características funcionales sin considerar las no

funcionales como los tiempos estimados de procesamiento, entre otros. (Lugo Gonzalez, 2009)

Repositorios de Servicios: Suelen ser componentes basados en registros UDDI, que hoy en día van mucho más allá. Normalmente permiten almacenar servicios para ser disponibles en la propia arquitectura SOA siguiendo un modelo de publicación, descubrimiento y subscripción. Su forma de exponer los servicios se basa en los WSDL, lo que permite tener definido cualquier servicio, principalmente a través de un ESB o Bus de Servicios (al que normalmente va conectado el Repositorio de Servicios). Gracias a un repositorio de servicios vamos a disponer de todos los servicios organizados y fácilmente accesibles para ser reutilizados. (Espacio SOA, 2007)

1.3. Metodologías de desarrollo de software

Debido a que uno de los principales aportes de esta tesis es definir el planteamiento de una metodología tenemos que hacer un estudio de las diferentes metodologías de desarrollo de software empleadas al desarrollar sistemas, con el propósito de conocer de cerca cada una de ellas para que podamos tomar lo mejor de cada una y aplicarlas para generar la metodología que en el capítulo 5 se describen las siguientes metodologías que mejor se adaptan a las necesidades del Departamento de Desarrollo Informático.

1.3.1. Metodologías Agiles y Tradicionales

Al revisar el marco conceptual de la ingeniería de software y referente al proceso de desarrollo de software, se encuentran metodologías que promueven iteraciones a lo largo de todo el ciclo de vida de un proyecto de software, para el presente estudio se han clasificado en dos grupos: métodos tradicionales y métodos ágiles.

En nuestro medio tradicionalmente se han empleado los modelos en cascada, espiral, prototipos, método en V, desarrollo por etapas, Rational Unified Process (RUP) , Essential Unified Process for Software Development (EssUP), Open Unified Process (OpenUP).

En los últimos años se han desarrollado metodologías de tipo ágil, como Scrum, Extreme Programming (XP), Agile Unified Process (AUP), Microsoft Solutions Framework (MSF).

A continuación detallaremos dentro de estos grupos las metodologías más empleadas en nuestro medio, dentro de las cuales posteriormente son tomadas RUP y AUP.

1.3.1.1 Desarrollo de software tradicional

Son métodos extraídos del modelo de desarrollo de software en cascada, y hacen énfasis en la planeación, está planeación se la realiza esperando que el resultado de cada proceso sea determinante y predecible. Pero los métodos que son de tipo tradicional son vistos como burocráticos por las organizaciones que necesitan software eficiente en corto tiempo y cuyos requerimientos no se mantienen estables en el tiempo.

A continuación se mencionan tres de los métodos tradicionales que se han venido utilizando en algunos de los sistemas planteados dentro del DDI.

CASCADA.- Es el ciclo de vida en donde el desarrollo del software se divide en fases que son ejecutadas de modo secuencial. Cuando trabajamos con el modelo en cascada no pasamos a la siguiente fase del desarrollo hasta que la actual se ha completado al 100%. El modelo en cascada es un modelo muy estricto y nada compatible con proyectos de complejidad media-alta, esto se debe a que casi nunca partimos de un conjunto de requerimientos claros y cerrados.

OPEN UP.- Es una metodología donada por un grupo de empresas de software y publicada con licencia libre en el año 2007. Este proceso es conciso e incluye solo contenido fundamental, esta metodología no contiene lineamientos para todos los elementos de un proyecto pero si para los básicos. Los principios del Open up son compartir conocimiento, equilibrar las prioridades, centrarse en la arquitectura y el mejoramiento continuo a través de la retroalimentación.

RUP.- El proceso unificado racional es un proceso de desarrollo de software que se ha convertido en una de las metodologías estándar más utilizadas. El

RUP no tiene pasos firmemente establecidos sino que se adapta a las necesidades de cada organización.

1.3.1.2. Desarrollo Ágil de software

Se entiende como desarrollo ágil de software a un paradigma de desarrollo basado en procesos ágiles, comprende metodologías livianas que intentan alivianar tortuosos caminos burocráticos empleados en metodologías tradicionales, enfocándose en la gente y en los resultados. El desarrollo ágil minimiza los riesgos, desarrollando software en cortos lapsos de tiempo, poniendo énfasis en las comunicaciones personales en lugar de documentar absolutamente todo. Sin embargo los métodos ágiles son criticados como indisciplinados por la falta de documentación técnica.

Brevemente se describen los siguientes ejemplos de metodologías de este tipo:

SCRUM (1986) Metodología iterativa y creciente, enfocada hacia la gestión de procesos, contiene las mejores prácticas para trabajar en equipos en altamente productivos. Indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, con requisitos cambiantes, poco definidos, donde la innovación, la competitividad y la productividad son fundamentales. En Scrum el proceso parte de la lista de objetivos/requisitos priorizada del producto, que actúa como plan del proyecto. En esta lista el cliente prioriza los objetivos balanceando el valor que le aportan respecto a su coste y quedan repartidos en iteraciones y entregas. Un proyecto se ejecuta en bloques temporales cortos y fijos (mes y hasta de dos semanas), cada iteración tiene que proporcionar un resultado completo, un incremento de producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite.

PROGRAMACION EXTREMA XP (1996) Formulado por Kent Beck es el más destacado de los procesos ágiles, poniendo énfasis en la adaptabilidad más que en la previsibilidad considerando que los cambios de requisitos en la marcha son un aspecto inevitable en el desarrollo de un proyecto por lo tanto no se define todos los requisitos al inicio de un proyecto. Se trata de adoptar las mejores metodologías de desarrollo de acuerdo a lo que se pretende y

aplicarlo dinámicamente durante el ciclo de vida del software. En la XP, la importancia de los test (para encontrar errores) y la experiencia del cliente (feedback) adquieren una mayor importancia, haciendo que se trabaje en ciclos de menor tiempo

PROCESO UNIFICADO AGIL, AUP Es una versión simplificada de RUP, describe de una manera simple y fácil el proceso para el desarrollo de software usando técnicas ágiles y conceptos válidos en RUP. Esta metodología aplica técnicas comunes a otros enfoques ágiles, tales como Desarrollo Dirigido por Pruebas (test driven development - TDD), el diseño conducido por pruebas, desarrollo dirigido por modelos ágiles, gestión ágil de los cambios, y técnicas de refactorización de la base de datos para mejorar la productividad.

1.3.2. Metodologías orientadas a servicios

Si bien se ha avanzado en la visión conceptual y tecnológica de SOA y su soporte a los procesos de negocio, no ha ocurrido un similar avance en las metodologías y guías para su desarrollo. Sin embargo se citan a continuación los principales intentos para acoplar SOA dentro de enfoques y metodologías orientadas a servicios.

MSOAM.- Mainstream SOA Methodology. MSOAM es un enfoque práctico de SOA, esta metodología fue popularizada en los libros de Thomas Erl, proporciona un conjunto de procesos genéricos y prácticas que requieren mayor personalización cuando se incorporan a entornos empresariales, se la puede considerar como punto de partida; está diseñada para enfrentar decisiones clave relacionadas con valorar ventajas a través de estrategias de top-down contra las preferencias de enfoques de abajo hacia arriba. Es la metodología SOA menos propietaria en la industria, en la que intencionalmente se la mantiene genérica para que pueda ser personalizada en las empresas específicas y según los requisitos. En ese sentido, sus objetivos son muy similares a los de RUP. Como parte de una metodología general, estos procesos generan consideraciones y preocupaciones comunes sin dictar una secuencia rígida (Working with SOA and RUP, 2008).

SOMA

IBM anunció la arquitectura orientada a servicios y modelación (SOMA) como la primera metodología relacionada con SOA en el año de 2004. SOMA se refiere a un dominio general de modelado de servicios necesario para diseñar y crear SOA; cubre un ámbito más amplio e implementa SOAD (descrito más abajo) a través de la identificación, la especificación y la realización de servicios, los componentes que dan cuenta de los servicios (componentes de servicio) y los flujos que pueden usarse para componer los servicios. Ayuda a la identificación de actividades de alto nivel creando ciertos artefactos del modelamiento orientado a servicios (Arsanjani, 2004).

SOAD

Service-Oriented Analysis and Design (SOAD), también conocida como modelamiento orientado a servicios, es un proceso definido por IBM siguiendo la Arquitectura Orientada a Servicios SOA. Es una metodología que ha sido creada para cubrir las necesidades propias de SOA en el análisis y diseño orientados a Servicios, agrega innovaciones para repositorios de servicios, orquestación de servicio y el bus de servicios empresarial, ayuda a desplegar aplicaciones como servicios Web basados en tecnologías SOAP, WSDL y UDDI (Olaf Zimmermann, 2004).

Capítulo 2. DOMINIO DEL NEGOCIO

2.1. Antecedentes del Departamento de Desarrollo Informático

Para adentrarnos en los temas propuestos para la presente tesis, inicialmente se hace una contextualización del Departamento de Desarrollo Informático. Al conocer la manera en que se originó, al comprender los objetivos de su creación y las tareas que se realizan dentro de este departamento, se logrará una completa comprensión de los alcances y limitaciones en el desarrollo de sistemas, y de las propuestas realizadas en la presente tesis.

2.1.1. Reseña del Departamento

La Universidad de Cuenca mantuvo por algunos años dependencias como la Red Académica Integral, el Sistema de Información Regional, el Centro de Cómputo Administrativo, que conformaban el área de Informática de la Universidad de Cuenca y no estaban relacionados formal ni informalmente. Las labores que realizaban estas dependencias eran:

- La Red Académica Integral estaba a cargo de la implementación y mantenimiento de la red de datos universitaria, tenía a su cargo el portal Web de la Universidad y se encargaba del soporte a los usuarios a nivel de red.
- El Sistema de Información Regional estaba a cargo de las actividades relacionadas con sistemas geo referenciados, los cuales se realizaban en base a convenios con instituciones externas a la Universidad.
- El Centro de Cómputo Administrativo estaba a cargo del desarrollo de sistemas informáticos para el área administrativa y académica de la Universidad y del mantenimiento de estos sistemas.

Las dependencias citadas mantenían tareas separadas con la necesidad de coordinación cuando se requería información o labores compartidas.

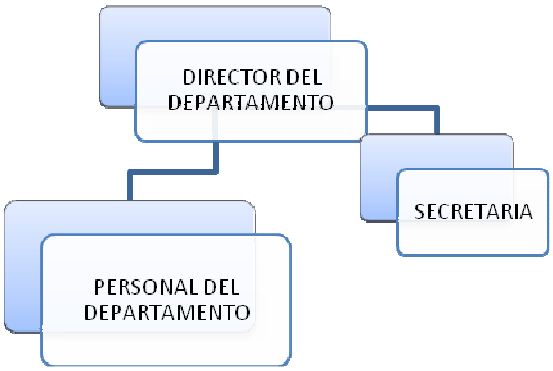
En el mes de marzo del 2001 el Consejo Universitario crea el Departamento de Desarrollo Informático de la Universidad de Cuenca mediante resolución del H. Consejo Universitario, a raíz de una estructuración del equipo de trabajo de la entonces nueva administración universitaria liderada por el Dr. Jaime Astudillo

Romero e Ing. Fabián Carrasco Castro; en sesión de fecha 20 de Marzo de 2001, el Consejo Universitario resuelve crear la Dirección del Departamento de Desarrollo Informático a cargo del Ingeniero Patricio Guerrero; quien estuvo en funciones hasta el mes de septiembre de 2001. Posteriormente el Ing. Otto Parra Gonzalez asume la dirección del departamento, desde septiembre de 2001 hasta septiembre de 2007. Desde el mes de octubre del año 2007 Ing. Rodrigo Padilla es designado como Director del Departamento, hasta la fecha (año 2010), que continúa.

El Departamento de Desarrollo Informático fue concebido bajo la siguiente dependencia administrativa:

Depende y responde directamente al Rectorado como se observa en la figura 11, atiende las necesidades de servicios informáticos de todas las facultades y dependencias administrativas de la Universidad de Cuenca.

Figura 11 Dependencia del DDI



Inicialmente se planteó la organización definida en la figura 12, propuesta inicialmente como un Director de Informática a cargo de varios ingenieros de sistemas.

Figura 12 Organización inicial del DDI

El Director del Departamento es el encargado de planificar y hacer un seguimiento de las actividades y proyectos en el área informática en base al presupuesto aprobado para su realización. El Personal del Departamento está formado por Ingenieros de Sistemas, los cuales ejecutan las actividades asignadas por el Director del Departamento de Desarrollo Informático.

Para facilitar la organización interna, inicialmente el Departamento se dividió en las áreas de Redes y Desarrollo de software. La tarea del soporte a los usuarios era compartida por todos los ingenieros que laboraban en el departamento.

Sin embargo, aunque el DDI se creó para dar soporte especialmente a las áreas de desarrollo de software y soporte a la red académica universitaria; ambas áreas no pudieron despegar con un impulso similar; existió gran demanda de interconectividad con la utilización de internet y correo electrónico, con lo que el área de redes tuvo mucha demanda, reflejándose en el impulso y crecimiento mayor del área de comunicaciones.

En Octubre de 2007 se reorganiza el departamento internamente, se crea una nueva área encargada del soporte a los usuarios, se designa un coordinador de cada área y se definen perfiles de cargos; que aunque no se oficializa en la estructura orgánica externa, ni en la definición formal de cargos, internamente se lo utilizó para organizar mejor el trabajo de los equipos a la vez que cada coordinador podía hacer el seguimiento de proyectos e informar al Director sobre la situación de los mismos.

Desde septiembre de 2008 el Departamento de Desarrollo informático queda organizado de la siguiente manera:

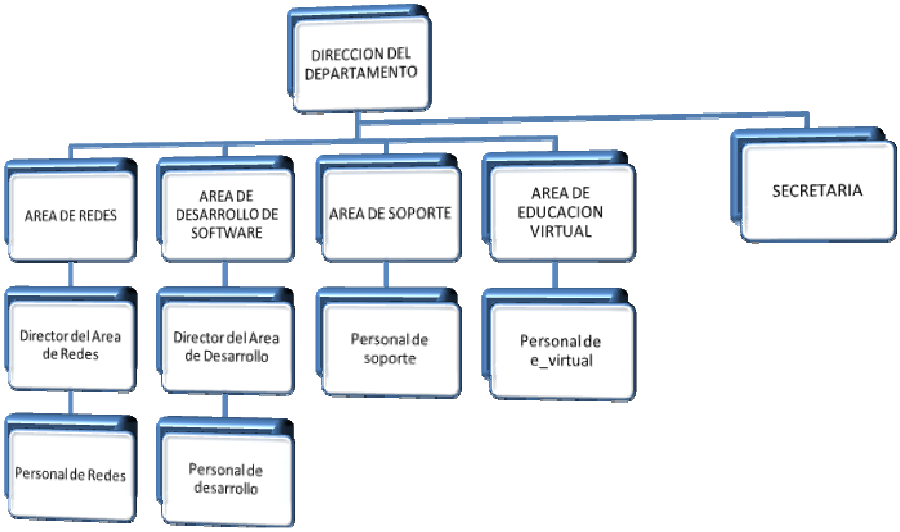


Figura 13 Estructura orgánica del DDI

En la figura 14 podemos observar las áreas de trabajo en las que se organiza el DDI, a continuación el detalle de cada una.

DIRECCION DEL DEPARTAMENTO.- Encargada de la planificación de las actividades a realizarse, de la coordinación periódica con el personal que labora en cada área, de las gestiones ante las autoridades y que son requeridas para el cumplimiento de las actividades, de la solución de problemas de orden técnico que se presenten en cualquiera de las áreas del Departamento, entre otras

AREA DE REDES Y SERVICIOS DE INTERNET.- Reemplaza a la Red Académica Integral. Está a cargo de la implementación de redes locales, del mantenimiento de las redes existentes, del desarrollo y mantenimiento del portal de la Universidad, de la implementación de servicios de red, del mantenimiento del Sistema Nacional de Información Bibliográfica SNIB

AREA DE DESARROLLO DE SOFTWARE.- Está a cargo del mantenimiento de los sistemas informáticos existentes en la Universidad, del desarrollo del nuevo sistema de gestión universitaria que integrará las diferentes dependencias universitarias.

AREA DE SOPORTE A USUARIOS.- Está bajo la responsabilidad de esta área el brindar soporte técnico a los usuarios, tanto en el manejo de equipos (hardware) como en el uso de las herramientas de software (programas, utilitarios, sistema operativo, etc.)

AREA DE E_VIRTUAL.- A partir de la implementación del sistema de créditos de la Universidad de Cuenca, en el mes de Noviembre de 2008 se ha constituido esta nueva área para el departamento, encargada de dar soporte la educación virtual, mantenimiento de la plataforma y procesos de capacitación relacionados a este tema.

2.1.2 Visión del Departamento

En la fecha del estudio que se realiza con la presente tesis, no se dispone de un documento en el que conste explícitamente información estratégica que describa la misión del departamento, únicamente existe un documento que en el año 2005 el Director del Departamento Informático de ese entonces hace como una definición de la visión, misión y objetivos del departamento:

VISION

“Ser un departamento que propicie el desarrollo de Informática en la Universidad de Cuenca, para lo cual debe liderar el cambio tecnológico y la modernización del sistema de información de manera que se agreguen valores de calidad y productividad a la gestión académica y administrativa. Además debe propiciar, a través de los servicios informáticos que ponga a disposición, la innovación y el desarrollo de la cultura informática que permita que la Universidad de Cuenca se muestre como una universidad moderna y eficiente a través del uso de las tecnologías de información y comunicación (TICs)”

MISION

La misión del Departamento de Desarrollo Informático es la de proporcionar y administrar, con calidad, los servicios que cumplan con las necesidades informáticas, con la finalidad de apoyar a los miembros de la comunidad universitaria de manera eficiente, efectiva y oportuna en sus funciones y en los procesos administrativos y académicos de la Universidad de Cuenca.

2.1.3. Situación del área de desarrollo de software

El área de desarrollo de software, dentro de los primeros años de existencia del Departamento de Desarrollo Informático de la Universidad de Cuenca, se dedicó a desarrollar sistemas de tipo administrativo, dando soluciones puntuales a las necesidades que día a día iban planteando las diferentes dependencias universitarias. Inicialmente se encargaba de proporcionar datos, como nóminas o padrones electorales, proporcionando información en forma de reportes personalizados, que eran solicitados continuamente, ocasionando que los desarrolladores tengan poco tiempo para dedicarse a tareas específicas de su área.

Debido a las necesidades de software en las Facultades, el crecimiento del área de desarrollo de software dentro de la misma Universidad de Cuenca ha sido dispar, cada Facultad se preocupó de hacer sus propios sistemas conforme sus necesidades lo requerían; consecuencia de la misma estructura y reglamentos diferentes para cada Facultad. En la etapa consolidación inicial del DDI, no existía una autoridad directa del departamento de informática hacia las

Facultades y Áreas Universitarias en lo que ha software se refería, no se disponía de personal suficiente para satisfacer la demanda de sistemas y muchos proyectos de software que se desarrollaron para el departamento, fueron resultado de proyectos de tesis, que solucionaban pequeños problemas puntuales, el sistema resultante operaba temporalmente mientras estaban presentes los desarrolladores y al retirarse los graduados el sistema quedaba sin sustento; esta problemática ocasionó que surja desconfianza hacia el desarrollo por parte del DDI, decayendo en algunos casos la imagen del departamento. Un ejemplo de esta problemática constituye el sistema académico único que estuvo a cargo del DDI y fue planteado como el resultado de un proyecto de tesis, que no pudo ser implantado de forma única en todas las Facultades, debido a que los reglamentos y procesos de calificación diferían sustancialmente en cada una de las Facultades de la Universidad.

La realidad es que los sistemas existentes dentro de la Universidad de Cuenca se han ido desarrollando aisladamente para solucionar problemas puntuales sin una visión integral, cuya construcción depende de diversas técnicas, creando estructuras complejas que no fueron diseñadas para interactuar entre ellas, con los problemas derivados de sistemas que responden a una necesidad específica sin satisfacer las necesidades de interconectividad con su entorno. Tal es el caso del sistema para el departamento de Recursos Humanos, desarrollado como un proyecto de tesis, que desde el inicio no tuvo una visión de relación con otros sistemas y no se dejó interfaces abiertas hacia sistemas de otros departamentos como Admisión y Becas; por lo que fue necesaria una reprogramación completa hacia un nuevo sistema de Recursos Humanos. En ese entonces se comenzó a analizar la existencia de un sistema unificado en el que todos se deberían enlazar y que a futuro sería el núcleo común para crear un sistema integrado.

En la actualidad existen otros proyectos en desarrollo y están en una etapa de implementación inicial, como: el sistema académico orientado al sistema de créditos, el sistema de colaboración interno, el sistema de evaluación docente, el sistema de voto electrónico, el sistema de investigación, en algunos casos la implementación será contratada a terceros bajo un esquema de análisis y diseño elaborado por el Departamento de Desarrollo Informático. Con las

experiencias aprendidas en proyectos de desarrollo de software anteriores, la visión es el orientar el desarrollo hacia un gran sistema unificado, que disponga un núcleo común.

2.1.4. Madurez del área

Si bien en los párrafos anteriores se describe la situación en la que se ha movido el área de desarrollo de software del DDI en los últimos años y se conoce que el área de desarrollo de software dentro del Departamento de Desarrollo Informático, adolece de los problemas comunes a muchas organizaciones que se encargan de crear y mantener software: no existen planes rigurosos, sus actividades se enfocan en resolver las crisis que se presentan, carecen de bases objetivas para evaluar la calidad de los productos o para resolver los problemas que surgen.

Para tener un acercamiento y evaluar la situación del área de desarrollo de software se se utilizó un cuestionario definido en el Anexo III. CUESTIONARIO DE ACERCAMIENTO AL AREA DE DESARROLLO DE SOFTWARE EN EL DDI, el cual ha sido creado tomando como base el marco CMMI con Representación Continua, en base a la disciplina Ingeniería de Software, sumada al modelo de madurez para Administración de Proyectos PMI; si bien se pudiera evaluar con mejores instrumentos como el SCAMPI, que se basa en un método de evaluación propio de CMMI, basado en evidencias y con un método que no únicamente se basa en entrevista, hay que considerar que esta técnica del cuestionario sirvió únicamente para un acercamiento y conversaciones iniciales con el personal del departamento y conocimiento de la realidad en la que se llevan los proyectos de software en el DDI.

Las preguntas definidas en el cuestionario han sido organizadas según las áreas de conocimiento para administración de proyectos: Alcance, Tiempo, Costos, Comunicación y Riesgos, cuidando de incluir por lo menos una de cada nivel de CMMI, cada grupo ha sido estimado para conseguir un porcentaje del 100% o un nivel de perfección al conseguir todos los puntos estimados como más necesarios hacia los niveles más altos de CMMI y considerando que se tiene 5 niveles, para cada nivel se asignó un valor del 20%, si dentro de un

nivel están definidas varias preguntas, a cada una se les a otorgado un porcentaje proporcional para finalmente conseguir la suma del 20%.

La encuesta se aplicó al personal desarrollador de software, y se tomaron las respuestas contestadas con SI, las cuales aplicadas en proporción a los pesos asignados a cada pregunta reflejaron el nivel, se obtuvieron los siguientes resultados:

Ámbito	Numero Preguntas	Porcentaje obtenido
Alcance	7	73 %
Tiempo	6	70 %
Costo	6	58 %
Comunicación	6	45 %
Riesgos	5	22 %
	Madurez Promedio	***53%***

Tabla 4 Madurez en la Gestión de Proyectos

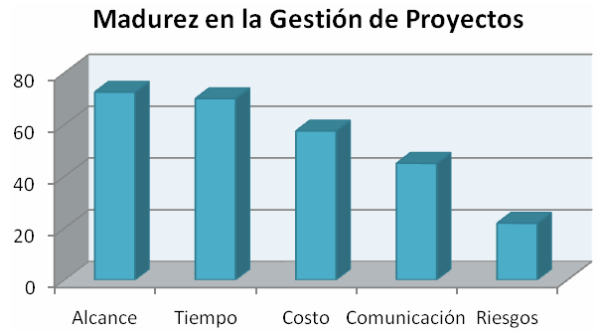


Figura 14 Madurez en la gestión de proyectos

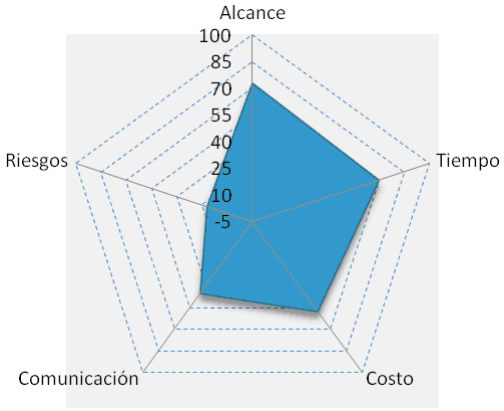


Figura 15 Madurez esquematizada por áreas

En los gráficos se puede observar que el personal del área de desarrollo de software encuestado, estima que el área de definición del alcance los proyectos y la definición de los tiempos empleados es la que mejor se especifica dentro de los proyectos planteados para el desarrollo de software, el área de costos está considerada parcialmente; mientras que el área de comunicación entre los miembros del equipo si bien existe, adolece de procesos explícitamente definidos por escrito. El área que tiene varias deficiencias es la del manejo de riesgos, la cual casi no ha sido considerada en la definición y desarrollo de proyectos.

2.2. Análisis de contexto

2.2.1. Organigrama de la Universidad de Cuenca

El esquema está disponible en la página Web de la Universidad y presenta una variación de dependencia del DDI hacia el Vicerrectorado, pero no refleja exactamente la situación de la Universidad.

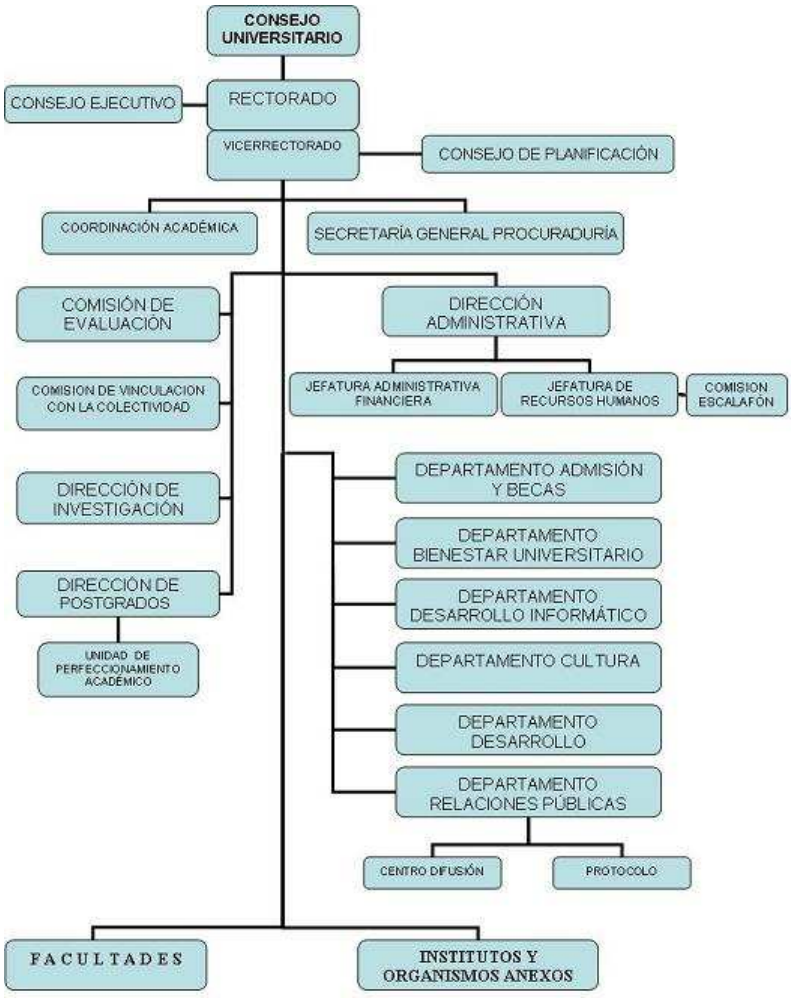


Figura 16 Organigrama de la Universidad de Cuenca

2.2.2. Análisis de Actores

En una organización como la Universidad de Cuenca, que no dispone de una estructura estática, más bien está en constante cambio y existen muchos actores; para facilitar el análisis del entorno se utilizará la clasificación propuesta en (Gea, y otros, 2003) se clasifica a los actores universitarios de la siguiente manera, que posteriormente será utilizada para la codificación y permisos asignados al definir servicios web:

- **Grupo**. Es la unidad mínima de organización, consistente en una agregación estructurada de actores. Los grupos poseen identidad y comportamiento.
- **Organización**. Todas la estructuras de grupos se disponen en torno a organizaciones, que representan ecosistemas con características compartidas
- **Rol**. Los grupos se organizan y estructuran en base a roles. Un rol identifica un comportamiento estereotipado dentro del entorno, el cual puede desempeñar un actor.
- **Actor**. Un actor es un agente activo con iniciativa y capaz de interactuar con el resto de miembros del grupo. La asignación de roles a actores en los grupos pueden variar por diferentes causas, por lo tanto se puede denominar participante al actor que en un instante dado desempeña un rol dentro de un grupo.

Grupos

Grupo	Descripción
Consejos	Máximo organismo de gobierno, conformado por un grupo electo de la comunidad universitaria
Direcciones	Máxima autoridad ejecutiva de la comunidad universitaria, a diferentes niveles
Dependencias	Unidades de organización para asuntos administrativos y de gestión necesarios para la actividad académica y de investigación universitaria
Unidades	Unidades de organización y coordinación dentro de la Universidad, de naturaleza disciplinaria o interdisciplinaria
Facultades	Máximas unidades académicas de la Universidad
Escuelas	Son unidades académicas y administrativas de las Facultades, de naturaleza disciplinaria, que organizan e imparten los estudios y carreras
Departamentos	Son unidades académicas y administrativas de integración interdisciplinaria
Institutos	Son unidades académicas y administrativas adscritos o no a las Facultades sin conferir grados o títulos
Comisiones	Comisión especializada en conseguir un objetivo de la vida universitaria
Centros	Unidades de apoyo a las actividades académicas de la Universidad o de las Facultades para su promoción, coordinación, desarrollo y fortalecimiento
Asociaciones	Grupos que garantizan la libertad de asociación, opinión y pensamiento para estudiantes, docentes, empleados y obreros de la Universidad
Miembros	Personas que conforman la comunidad universitaria, conformada por estudiantes, docentes, directivos, empleados, obreros y personas que se relacionan con la Universidad.

Tabla 5 Grupos que intervienen en la Universidad de Cuenca

Organización

Organización	Descripción
Órgano y Autoridad de Gobierno	Son órganos y autoridades de gobierno que dirigen el rumbo de la Universidad
Órgano de Apoyo	Son órganos colegiados de apoyo a la gestión Universitaria
Comunidad Universitaria	Organismo consultivo y electivo, conformado por profesores, estudiantes, empleados y obreros.
Órgano Ejecutivo	Son órganos encargados de llevar a la práctica resoluciones adoptadas por órganos de gobierno Universitario
Órgano Asesor	Son órganos de carácter técnico encargados de propuestas y planificación para la Universidad
Órgano Académico	Son organismos que posibilitan el cumplimiento de las actividades de docencia, investigación, extensión y otras de carácter disciplinario o interdisciplinario
Órgano Administrativo /Gestión	Son organismos que posibilitan el cumplimiento de funciones administrativas o de gestión
Externo	Son organismos, entidades o personas que se relacionan con la universidad y son independientes a esta.

Tabla 6 Organizaciones en la Universidad de Cuenca

Roles

Rol	Descripción
Gobierno	Rol asignado al gobierno que emana de la libre voluntad manifestada por docentes, estudiantes, empleados y obreros.
Ejecutivo	Rol asignado a órganos y autoridades de carácter ejecutivo
Administrativo	Rol asignado a personas y dependencias que cumplen actividades de tipo administrativo
Asesor	Rol asignado a actividades de planificación o de asesoramiento
Técnico	Rol asignado a actividades técnicas
Académico	Rol asignado a tareas académicas
Investigación	Rol asignado a tareas investigativas
Directivo	Rol asignado a tareas de dirección y gestión administrativa
Extensión	Rol asignado a tareas de extensión y vinculación con la colectividad
Consultivo/Electivo	Función asignada a Tareas de consulta o elecciones

Tabla 7 Roles en la Universidad de Cuenca

Clasificación de ACTORES en la Universidad de Cuenca

Actor/Participante	Grupo	Organización	Rol
AREA DE GOBIERNO Y GESTION			
Consejo Universitario	Consejos	Órgano y Autoridad de Gobierno	Gobierno
Rector	Direcciones	Órgano y Autoridad de Gobierno	Gobierno / Ejecutivo
Rectorado	Dependencias	Órgano Administrativo	Ejecutivo
Vicerrector	Direcciones	Órgano y Autoridad de Gobierno	Gobierno / Ejecutivo
Consejo de Planificación	Consejos	Órgano Asesor	Técnico / Asesor
Consejo Ejecutivo	Consejos	Órgano Ejecutivo	Ejecutivo
Secretaría General Procuraduría	Unidades	Órgano Asesor	Técnico
Dirección Administrativa-Financiera	Direcciones	Órgano Ejecutivo	Administrativo
Jefatura de Recursos Humanos	Direcciones	Órgano Ejecutivo	Administrativo
Asamblea de Facultad	Asambleas	Órgano y Autoridad de Gobierno	Consultivo/Electivo
Consejo Directivo de Facultad	Consejos	Órgano Académico y Órgano Administrativo	Gobierno
Consejo Académico	Consejos	Órgano Académico y Órgano Administrativo	Gobierno
Decano	Miembros	Órgano y Autoridad de Gobierno	Gobierno/Directivo
SubDecano	Miembros	Órgano y Autoridad de Gobierno	Directivo
Director de Escuela	Miembros	Comunidad Universitaria	Directivo
AREA ADMINISTRATIVA			
Vicerrectorado	Dependencias	Órgano Administrativo	Ejecutivo
Coordinación Académica del Rectorado	Centros	Órgano de Apoyo	Ejecutivo
Unidad de Perfeccionam. Académico	Unidades	Órgano de Apoyo	Académico
Comisión de Escalafón	Comisiones	Órgano de Apoyo	Administrativo
Comisión de Evaluación Interna y Acreditación	Comisiones	Órgano de Apoyo	Asesor
Vinculación con la Colectividad	Comisiones	Órgano de Apoyo	Extensión

Departamento de Desarrollo Informático	Departamentos	Órgano Administrativo	Administrativo
Departamento Admisión y Becas	Departamentos	Órgano Administrativo	Administrativo
Departamento de Bienestar Universitario	Departamentos	Órgano Administrativo	Administrativo
Departamento de Cultura	Departamentos	Órgano Administrativo	Administrativo
Departamento de Relaciones Públicas	Departamentos	Órgano Administrativo	Administrativo
Auditoría	Departamentos	Órgano de Apoyo	Administrativo
Archivo	Dependencias	Órgano de Apoyo	Administrativo
Presupuesto	Dependencias	Órgano de Apoyo	Administrativo
Coordinadores de Informática de Dependencias	Miembros	Comunidad Universitaria	Administrativo
Aula de Derechos Humanos	Centros	Órgano de Apoyo	Extensión
APUC / AETUC	Asociaciones	Comunidad Universitaria	Extensión
Empleados	Miembros	Comunidad Universitaria	Administrativo
Trabajadores	Miembros	Comunidad Universitaria	Administrativo
Obreros	Miembros	Comunidad Universitaria	Administrativo
AREA ACADEMICA			
Facultad de Jurisprudencia	Facultades	Órgano Académico	Académico
Facultad de Filosofía	Facultades	Órgano Académico	Académico
Facultad de Ciencias Médicas	Facultades	Órgano Académico	Académico
Facultad de Ingeniería	Facultades	Órgano Académico	Académico
Facultad de Arquitectura	Facultades	Órgano Académico	Académico
Facultad de Ciencias Agropecuarias	Facultades	Órgano Académico	Académico
Facultad de Ciencias Químicas	Facultades	Órgano Académico	Académico
Facultad de Odontología	Facultades	Órgano Académico	Académico
Facultad de Artes	Facultades	Órgano Académico	Académico
Facultad de Ciencias de la Hospitalidad	Facultades	Órgano Académico	Académico
Escuela de Informática	Escuelas	Órgano Académico	Académico
Departamento de Idiomas	Departamentos	Órgano Académico	Académico
Instituto de Educación Física	Institutos	Órgano Académico	Académico
Directores de Escuelas	Miembros	Comunidad Universitaria	Académico
Coordinación de asignaturas obligatorias	Miembros	Órgano de Gestión	Directivo/Académ.
Dirección de Postgrados	Direcciones	Órgano Ejecutivo	Directivo/Investig.
Postgrados	Organismo	Comunidad Universitaria	Investigación
Docentes	Miembros	Comunidad Universitaria	Académico
Estudiantes	Miembros	Comunidad Universitaria	Académico
Centro Documental Juan Bautista Vásquez	Dependencias	Órgano Administrativo	Académico
AREA DE INVESTIGACION			
DIUC. Dirección de Investigación	Direcciones	Órgano de Gestión	Investigación
Consejo de investigación	Consejos	Órgano Ejecutivo	Investigación
Coordinación de Programas y Proyectos de Investigación	Dependencias	Órgano Ejecutivo	Administrativo
DIP Departamentos de Investigación y Postgrado de Facultades	Dependencias	Órgano de Gestión	Investigación
Programas y Proyectos de Investigación	Programas	Órgano Ejecutivo	Investigación/Extensión
PROMAS	Programas	Órgano Ejecutivo	Investigación/Extensión

CESPLA	Programas	Órgano Ejecutivo	Investigación/Extensión
CECEMIN	Programas	Órgano Ejecutivo	Investigación/Extensión
PYDLOS	Programas	Órgano Ejecutivo	Investigación/Extensión
ACORDES	Programas	Órgano Ejecutivo	Investigación/Extensión
CEDIUC	Programas	Órgano Ejecutivo	Investigación/Extensión
Investigador	Miembros	Comunidad Universitaria	Académico
AREA EXTERNA			
VLIR	Institución	Externo	Técnico
SENACYT	Institución	Externo	Técnico
CEDIA	Institución	Externo	Técnico
Usuario/Beneficiario	Miembros	Externo	Extensión
Proveedor de Tecnología	Miembros	Externo	Técnico
Fondo Provida	Institución	Externo	Administrativo

Tabla 8 Clasificación de Actores de la Universidad de Cuenca

2.2.3. Diagramas de Contexto

A continuación se representa la situación del Departamento de Desarrollo Informático dentro de la Universidad de Cuenca en forma de diagramas de contexto, concepto tomado del modelado funcional de procesos IDEF0 (*ICAM Definition Method Zero,* método formalizado de descripción de procesos), con el objetivo de representar en una forma de esquema de sobrevista general, tanto las áreas fundamentales que interactúan con dicho departamento, como las organizaciones y los grupos en los que se clasificaron todos los actores que intervienen en el quehacer universitario.

Esquema de contexto por Áreas

En la siguiente figura se observa que el Departamento de Desarrollo Informático de la Universidad de Cuenca interactúa con 6 áreas en las que se

Figura 17 Esquema de contexto por áreas

puede organizar la gestión de la Universidad de Cuenca, algunas de ellas constan explícitamente dentro del estatuto universitario, sin embargo áreas

como la administrativa y el área externa hacen referencia a aquellas que intervienen directamente en el quehacer diario del Departamento Informático.

A continuación se expresa gráficamente la relación del DDI con las organizaciones y grupos que se relacionan dentro del convivir universitario, que son los mismos citados en la clasificación de actores en la Universidad de Cuenca propuesto en la tabla 8.

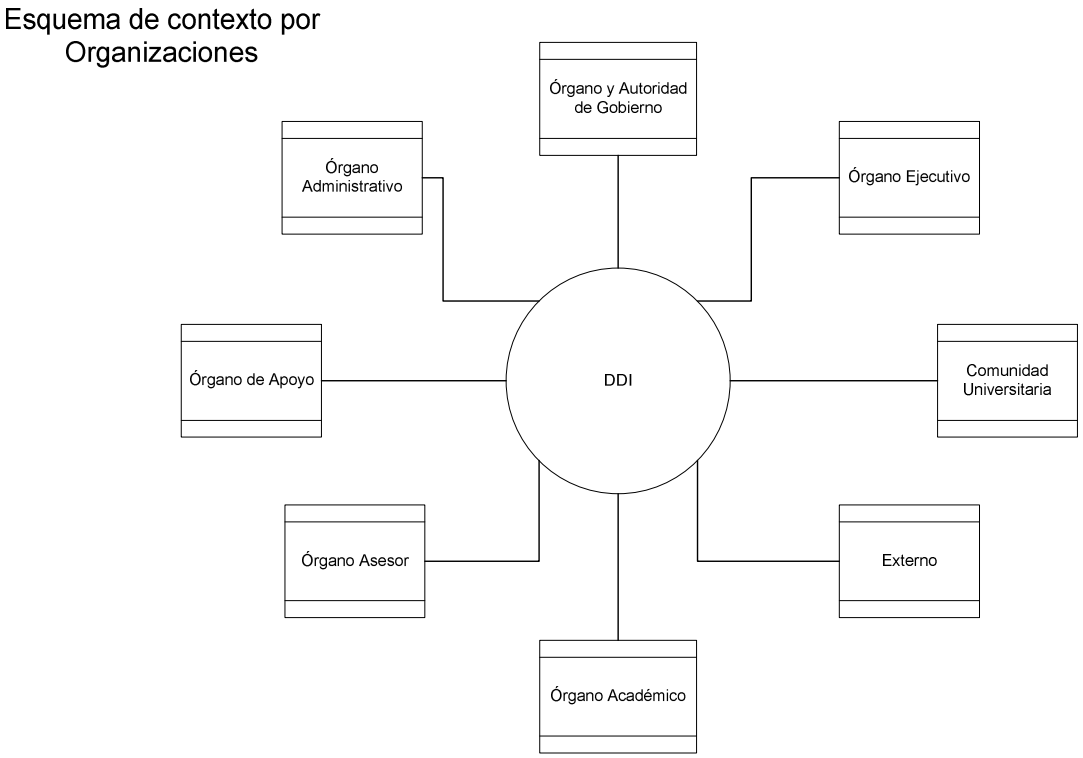


Figura 18 Contexto por Organizaciones

Esquema de contexto por Grupos

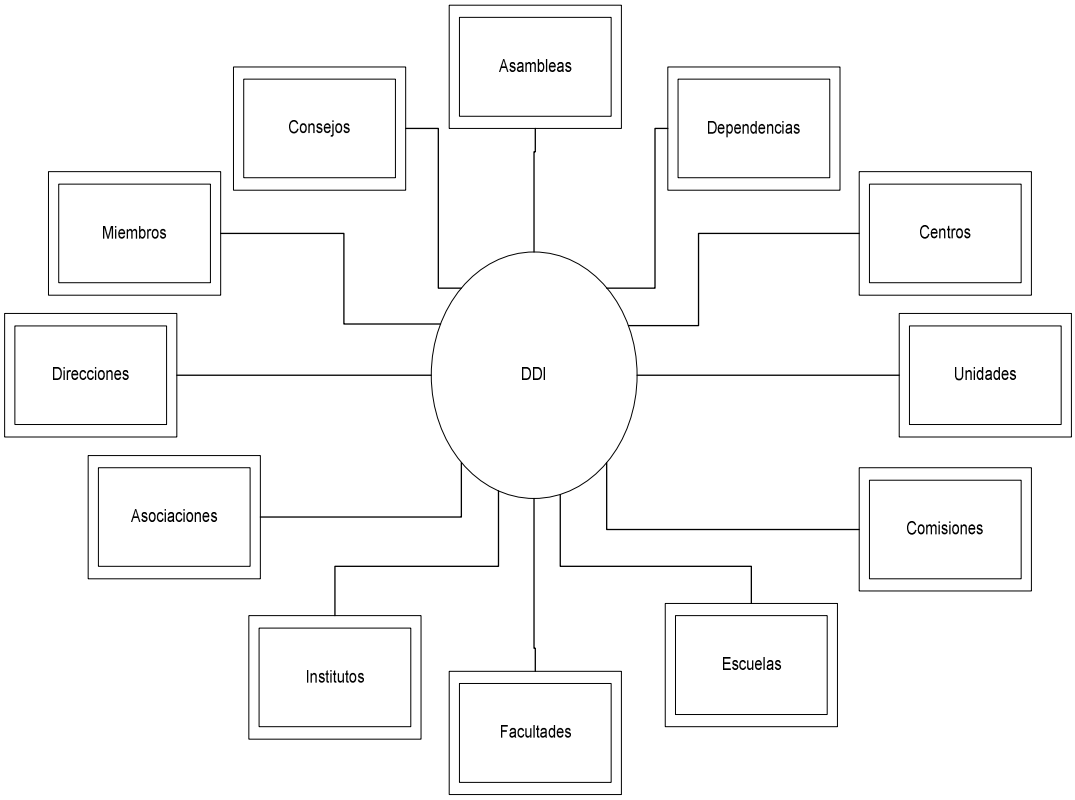


Figura 19 Contexto por Grupos

2.3. Revisión de Sistemas Existentes

Ahora que se tiene claro el panorama con el que interactúa el Departamento Informático y adentrándonos a la parte de desarrollo de software, objetivo central de la presente tesis, a continuación se hace una recopilación de los sistemas que han sido desarrollados en la Universidad de Cuenca y que están bajo el control del Departamento de Desarrollo Informático:

2.3.1. Sistemas desarrollados bajo responsabilidad del DDI

Sistema	Descripción	Departamento	Autor(es)	Plataforma	Utilizado	PUESTA EN MARCHA
Sistema de Contabilidad y Presupuesto	Manejo de Contabilidad y Presupuestos	Financiero Contabilidad	Ing. Hernán Vintimilla	RPG	SI	Diciembre 1974
Rol de Pagos	Control de Recursos Humanos y Pago de Roles	RRHH DDI	Ing. Hernán Vintimilla	RPG	NO	Diciembre 1974
Sistema de Notas	Registro Académico y Matriculas	Quimica, Filosofía,	Patricio Juca Vladimir Morales	FOX	NO	
Sistema Académico	Registro Académico	Ingeniería, Filosofía	Yakiro Feican	Visual Basic	NO	2002
Sistema de Activos Fijos	Control de Activos Fijos	Contabilidad	Doris Suquilanda	Visual Basic	NO	2002 2002
Sistema de Activos Fijos	Control de Activos Fijos	Contabilidad	Andrés Auquilla Maurcio Garzón Milton Sisalima	.NET 2003	SI	2005
Sistema de Gasto interno	Control de Presupuesto de cada departamento	Contabilidad	Patricio Juca Vladimir Morales	Visual Basic	NO	
Sistema de Gasto interno / SCGI	Control de Gastos de cada departamento	Todas las dependencias	Ximena Carrilo Bolivar Piedra	.NET 2000	SI	2002
Sistema de Archivos Y Financiero	Rastreo de Documentación	Archivo y Depto. Financiero	Vladimir Morales	Lotus	SI	
Tesorería	Control de Especies, Vales	Tesorería	Hernán Vintimilla	RPG	SI	2006
Tesorería	Reporte de Inversiones	Tesorería	Tesis Laura Ordoñez y Sandra Calle	.NET 2000	PARCIALMENTE	Junio/2004
Oficina sin papel	Control de documentos	Toda la Universidad	Fernanda Ruiz	LOTUS NOTES	SI	Agosto/2004

Sistema Académico	Control Académico	Filosofía Ingeniería Agropecuarias Artes	José Zumba, Victor Auquilla,Walter Orozco, Cristian Barrera, Victor Rodriguez	.NET 2003	SI	Julio/2006
Sistema de Recursos Humanos	Control de Personal	Recursos Humano	Ma. Eugenia. Fajardo, Gabriela C., Miguel Cruz	.NET 2003	NO	Marzo/2006
Sistema de Gestión de Personal	Control de Personal y Roles de Pago	Recursos Humano	Marcelo Olivo María Eugenia Fajardo	.NET 2005	SI	Enero/2008
Módulo de Vacaciones	Control de Vacaciones de Personal	Recursos Humanos	Juan Pablo Cedeño	.NET 2003	SI	2006
Sistema de Mantenimiento	Control de ordenes de trabajo y control de stocks	Departamento de Mantenimiento	Mariana Paredes Maryuri Arevalo	ASP – Visual Basic	SI	Agosto/2004
Sistema de Gestión Socio Económica	Matrícula de Estudiantes y Manejo de Fichas Socio-Económicas	Admisión y Becas	Carol Guzmán Pablo Garrido Julio Salinas Marlene Robles	.NET 2005	SI	Septiembre/ 2008
Sistema para CONESUP	Registro de títulos en el CONESUP	Oficina del CONESUP	Bolivar Piedra	ASP	SI	Diciembre/ 2004
Sistema de registro de actas	Registro de Actas	Secretaría General	Caterine Coronel	.NET 2005	SI	Noviembre/ 2008
Sistema de Voto Electrónico	Manejo de elecciones	Universidad	Adriana Peñafiel	.NET 2003	NO	2006
Sistema de evaluación interna	Automatizar el proceso de evaluación	Departamento de Evaluación Interna	Tania Padilla Juliana Carpio	.NET 2005	SI	2008
Sistema de Bibliotecas	Control e ingresos de libros	Centro Documental Juan Bautista Vázquez	Cristina Carpio Lorena Siguenza	.NET 2003	SI	Junio/2002
SIUC – Sistema Integrado de Investigación de Universidad de Cuenca	Control de proyectos de investigación Convocatorias a concursos y carga de propuestas Llenado de formularios de presupuestos de proyectos	DIUC – Departamento de Investigaciones	Juan Pablo Cedeño Karina Quinde (Análisis y Diseño) DATAACROM (Desarrollo)	.NET 2008	SI	En implementación /Sep09

Tabla 9 Sistemas a cargo del DDI [2]

[2] *Por falta de información documentada, se ha recurrido a información obtenida de forma verbal en entrevistas directas a los responsables de los sistemas*

2.3.2. Clasificación de sistemas por tipo de tecnología

De la clasificación anterior y tomando solamente los sistemas que están siendo utilizados en la elaboración de la presente tesis: año 2008-2009, se ha procedido a clasificarlos según los tipos de tecnología descritos en el *Anexo II. TIPOS DE TECNOLOGIAS EN LA IMPLEMENTACION DE SISTEMAS,* obteniendo el siguiente cuadro de clasificación:

Sistema	Departamento	Plataforma	Tecnología
Sistema de Contabilidad y Presupuesto	Financiero Contabilidad	RPG	MONOLITICA
Sistema de Activos Fijos	Contabilidad	.NET 2003	MONOLITICA
Sistema de Gasto interno	Contabilidad y Todas las dependencias	.NET 2000	Tres Capas
Sistema de Archivos Y Financiero	Archivo Departamento Financiero	Lotus	PROCESAMIENTO CENTRAL
Tesorería	Tesorería	RPG	MONOLITICA
Oficina sin papel	Vicerrectorado Toda la Universidad	LOTUS NOTES	Lógica Distribuida
Sistema Académico Actual	Filosofía, Ingeniería Agropecuarias, Artes	.NET 2003	Tres Capas
Sistema de Gestión de Personal	Recursos Humano	.NET 2005	Tres Capas
Módulo de Vacaciones	Recursos Humanos	.NET 2003	Gestión Remota de Datos
Sistema de Mantenimiento	Departamento de Mantenimiento	ASP – Visual Basic	Tres Capas
Sistema de Gestión Socio Económica	Admisión y Becas	.NET 2005	N Capas
Sistema para CONESUP	Oficina del CONESUP	ASP	Tres Capas
Sistema de registro de actas	Secretaría General	.NET 2005	Tres Capas
Sistema de evaluación interna	Departamento de Evaluación Interna	.NET 2005	Tres Capas
Sistema de Bibliotecas	Centro Documental Juan Bautista Vázquez	.NET 2003	Bases de Datos Distribuidas
SIUC – Sistema Integrado de Investigación de Universidad de Cuenca	DIUC – Departamento de Investigaciones	.NET 2008	Tres Capas
Sistema Académico en Desarrollo	Todas las Facultades, Postgrados y Departamentos	.NET 2005	Tres Capas

Tabla 10 Clasificación de sistemas por tecnología

En la tabla anterior se puede observar que la mayoría de sistemas han sido desarrollados bajo la concepción de capas y han sido construidos con la plataforma .NET

2.4. Análisis de necesidades

Para cada uno de los sistemas existentes y citados anteriormente, se ha procedido a realizar un análisis detallado de cada uno de ellos para encontrar los posibles servicios e información compartida que se podría utilizar por otros actores dentro de la Universidad, siguiendo la estrategia Botom-Up que plantea ir al detalle de los sistemas mismos para terminar conceptualizando de forma general el conjunto de servicios globales a la mayoría de sistemas.

En siguiente tabla se puede visualizar la revisión de cada uno de los sistemas administrados por el DDI en los que se describe que tipo de información es la que se puede considerar compartida por otros entes universitarios y se hace una visualización de los servicios que se podrían obtener de esa aplicación.

Sistema	Información Compartida	Actores usuarios de la información	Posibles Servicios
Sistema de Contabilidad y Presupuesto	Presupuesto Anual Estado de cuentas contables x dependencia	• Nivel Gerencial • Dependencias	Revisión Cuanto puede gastar y que puede gastar
	Balances mensuales de la Universidad	• Gerencia • Gobierno	Revisión por entidades internas y externas
Sistema de Activos Fijos	Listado de bienes por custodio	• Custodios	Revisión de bienes a cargo de una persona
	Reporte de Activos	• Inventarios	Revisión de activos
Sistema de Gasto interno	Consumos por período o por ítem	• Jefes de Dependencia	Revisión de bienes consumidos
	Consumos x dependencias	• Nivel Gerencial	Revisión de consumos
Tesorería	Venta de especies valoradas Ingresos por rubros	• Director Financiero	Revisión de Especies Valoradas
Sistema de Gestión Socio Económica	Número de matrículas de estudiantes Proceso de cálculo de matrículas	• Sistema Académico • Estudiantes • Directivos	Servicio costos de matrícula Ficha Socioeconómica Reportes Financieros Cuotas o dividendos Estados de Pagos Becas
Sistema de Gestión de Personal	Datos de Profesores Distributivos Datos de empleados	• Recursos Humanos • Departamento Financiero • Biblioteca • Fondo Jubilación • Investigación	Nomina de Personas Verificación de datos uno o varios docentes, empleados y trabajadores Datos laborales Dependencias
Sistema Académico (anterior)	Datos de estudiantes de Pregrado Datos académicos Pensums Horarios	• Admisión y Becas • Órganos Académicos	Plan de carrera Revisión o creación de personas Listado de Titulados x Carrera x Fecha

			Servicio Reporte de uno o varios docentes
Sistema Académico (en desarrollo)	AdminUC, tablas comunes Personas Localidades Dependencias Períodos Lectivos Estudiantes Servicios Datos Estadísticos • Matrículas (Números) • Pases de Año • Abandonos Planes de Carrera Espacios físicos Tesis	• Admisión y Becas • Directivos • Recursos Humanos • FEUE • Secretarías • CONESUP • ConsejoPlanificación • Unidad Estadística • Unidad de Planificación • Recursos Humanos • Dpto Financiero • Biblioteca	Datos del Estudiante Datos Académicos Datos de Dependencias Datos estadísticos Distributivos Actualización de espacios físicos Padrones Electorales Reportes de calificaciones Datos de Graduados
Sistema de Bibliotecas	Revisión Deudores	• Dependencias	Servicio reporte individual deudores
Sistema de Mantenimiento		*La información que maneja es solamente de utilidad dentro de la dependencia*	
SIUC – Sistema Integrado de Investigación		*La información que maneja es solamente de utilidad dentro de la dependencia*	
Sistema para CONESUP		*La información que maneja es solamente de utilidad dentro de la dependencia*	
Sistema de registro de actas		*La información que manejan es solamente de utilidad dentro de la dependencia*	
Sistema de evaluación interna		*Será reemplazado por decisiones gubernamentales*	

Tabla 11 Visualización de posibles servicios por sistemas

Hasta este punto se han eliminando aquellos sistemas que prestan utilidad solamente dentro de ciertos departamentos y que prácticamente no tienen información útil para otros actores universitarios. A continuación en la tabla se los prioriza y se cita la cantidad de servicios que se originarían de cada uno de estos, considerándolos como los estratégicos para considerarlos dentro del estudio y marco de integración descrito en el siguiente capítulo 3.

Sistema	Lenguaje	Tecnología	# Servicios
Sistema Contabilidad y Presupuesto	RPG	MONOLITICA	3
Sistema de Activos Fijos	.NET 2003	MONOLITICA	1
Sistema de Tesorería	RPG	MONOLITICA	1
Sistema de Bibliotecas	.NET 2003	C/S Bases Datos Distrib	1
Sistema de Gasto interno	.NET 2003	Tres Capas	2
Sistema Académico Actual	.NET 2003	Tres Capas	5
Sistema de Gestión de Personal	.NET 2005	Tres Capas	3
Sistema Gestión Socio Económica	.NET 2005	N Capas	4
Sistema Académico en Desarrollo	.NET 2005	N Capas	10

Tabla 12 Sistemas Estratégicos para SOA

Capítulo 3. INTEGRACION DE SISTEMAS

Para lograr el objetivo de que los sistemas informáticos desarrollados en la Universidad de Cuenca puedan intercomunicarse entre sí, se hace la diferenciación entre sistemas ya desarrollados y los sistemas que se desarrollarán a futuro, que están a cargo del Departamento de Desarrollo Informático.

En el presente capítulo nos enfocaremos únicamente en los sistemas que ya están desarrollados en el DDI y guiándonos de la estrategia Top-Down[3], se procederá de tal manera que partiendo de los objetivos universitarios y del modelo del dominio de la Universidad de Cuenca, se pueda plantear un esquema de integración de los sistemas desarrollados en el DDI, utilizando servicios web.

3.1. Evaluación de la situación actual

3.1.1. Objetivos y situación del negocio

Los procesos de negocio se pueden ver como un conjunto estructurado de tareas que contribuyen colectivamente a lograr los objetivos de la organización, los procesos de negocio de una organización son parte de su cultura, se registran y difunden en manuales de procedimientos, diagramas de flujo y hasta en forma verbal (Alvez, y otros, 2006)

Se ha tenido dificultad al tratar de recopilar documentación que contenga los procesos de negocio que diariamente tienen lugar dentro de la Universidad de Cuenca, debido a que todas las actividades que se realizan no se encuentran documentadas, existe un conocimiento tácito que no se encuentra explicitado en documentos físicos. A pesar de que la elaboración del plan de desarrollo institucional es una exigencia contemplada en la Ley de Educación Superior, la Universidad de Cuenca no dispone de un plan de desarrollo en los términos que plantea la Ley de Educación Superior.

Existe un documento de carácter estratégico que es el estatuto de la Universidad de Cuenca, en donde se describe los fines de la universidad Anexo

[3] Top-Down: Estrategia de procesamiento de información. Se parte de lo general para entrar a detalles

I. FINES DE LA UNIVERSIDAD DE CUENCA, de los cuales se desprende la naturaleza de las actividades enfocadas en la *docencia, investigación, extensión y otras*, así como su carácter disciplinario o interdisciplinario (Estatuto de la Universidad de Cuenca, 2003), el mismo que define a rasgos generales el quehacer universitario.

En un texto auspiciado por la Facultad de Ingeniería de la Universidad de Cuenca y elaborado por el Lcdo. Lucas Achig Subía (Achíg Subía, 2007) se hace una propuesta sobre el perfil del plan estratégico de la Universidad de Cuenca para el período 2006-2010 y se plantea como misión para la Universidad de Cuenca: "La formación de seres humanos profesionalmente competentes, éticamente íntegros, comprometidos con los sectores sociales, especialmente los más vulnerables, con capacidad y motivación para educarse en forma continua, con sensibilidad para comprender, respetar y proteger los derechos humanos y el ambiente natural, a través del ejercicio articulado, crítico, innovador y prospectivo de la ***docencia, investigación, producción y vinculación con la sociedad".*** En el texto citado se manifiesta "En la Universidad de Cuenca se tiene un panorama de planeamiento de más incertidumbres que certezas, con más interrogantes que respuestas, con más réplicas que propuestas" y precisamente uno de los principales problemas con los que se ha encontrado el área de software del departamento de desarrollo informático es que para la mayoría de dependencias universitarias no se encuentran definidos los procesos sobre los cuales hay que trabajar, y aunque no es de competencia directa del DDI el definir los procesos, al necesitar desarrollar un sistema se ha tenido que empezar haciendo de facilitadores para la definición de los procesos, incluyéndolo dentro del alcance de los proyectos de software.

Debido a que no existe un manual de procedimientos aprobado y en vigencia, antes del desarrollo de los sistemas se cumple una fase previa a cargo de los mismos desarrolladores que permite llegar a un acuerdo común y lograr entendimiento sobre las actividades visualizadas por los programadores de las labores efectuadas por los usuarios finales, las mismas que se analizan y perfeccionan para posteriormente proceder a levantar los requerimientos detallados.

Debido a experiencias previas, en muchos proyectos desarrollados se ha sufrido retrasos y reescritura de programas debido a que los encargados del negocio universitario no están de acuerdo con los resultados obtenidos en los sistemas, fracasos ocurridos debido a que no pudo ser captada la realidad compleja del departamento o área que utilizaría el software desde el mismo punto de partida del proyecto. Tal es la situación del área financiera, que aunque es un pilar clave para la marcha universitaria, con la situación de nuevos requerimientos financieros solicitados por el presente gobierno del Econ. Rafael Correa y las nuevas reformas del gobierno con los sistemas ESIGEF: Sistema de Gestión Financiera y ESIPREM: Sistema para pagos y remuneraciones, para los cuales no se puede prever el tiempo de reemplazo de sistemas, ni se aclara totalmente los cambios solicitados y la manera en la que llevarán los registros financieros de las entidades de educación superior ecuatoriana. Por las nuevas disposiciones ocasionadas por la nueva constitución del país, se detuvo el proceso de adquisición de un sistema financiero que ya fue analizado y estuvo listo para la compra hasta asumir las reglas impone el gobierno para la gestión contable, con lo que los procesos financieros dentro de la universidad y los reportes para el gobierno ecuatoriano se los defina completamente y sean considerados estables.

3.1.2. Modelo del Dominio

De conformidad con los fines y naturaleza de las actividades universitarias, que son de docencia, investigación, extensión y otras; así como de su carácter disciplinario e interdisciplinario, la Universidad de Cuenca organiza su quehacer académico utilizando sistemas informáticos que dan apoyo a sus actividades diarias.

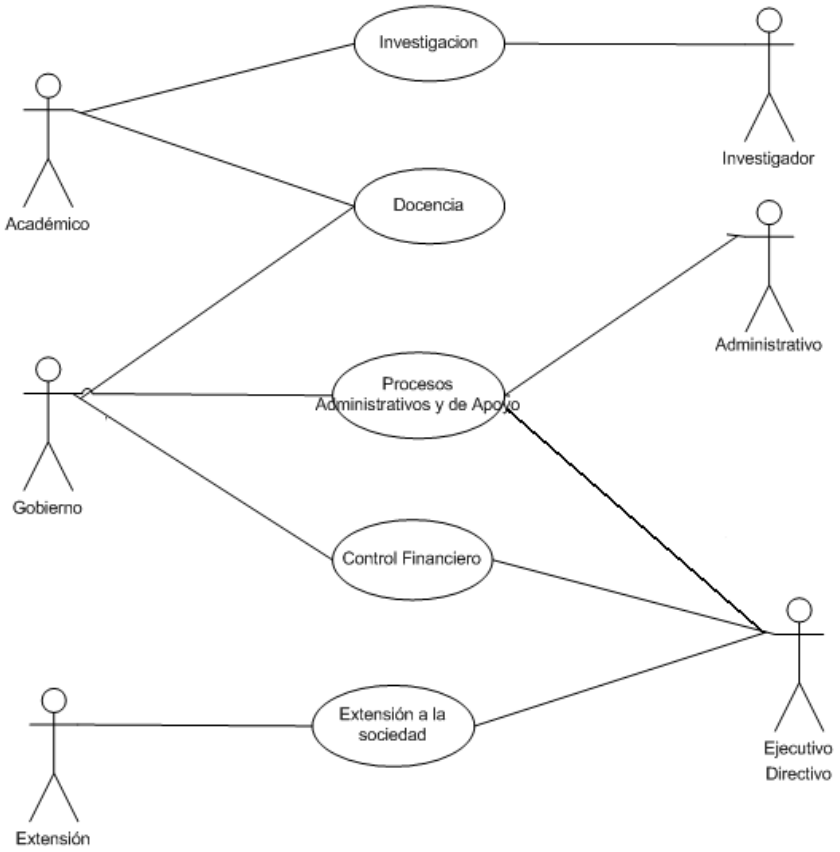


Figura 20. Caso de Uso del Negocio

El Departamento de Desarrollo Informático la Universidad de Cuenca no está a cargo de todos los sistemas que la universidad necesita y otros sistemas importantes para el quehacer universitario por diversas razones aun no han sido desarrollados, como los sistemas para el área de investigación y extensión hacia la colectividad; en base a los sistemas existentes y en base a los fines y objetivos que la Universidad tiene; para proceder con el estudio detallado de los sistemas existentes, se han agrupado estos en las áreas que hacen referencia al Control Académico, Control Administrativo, Control Financiero.

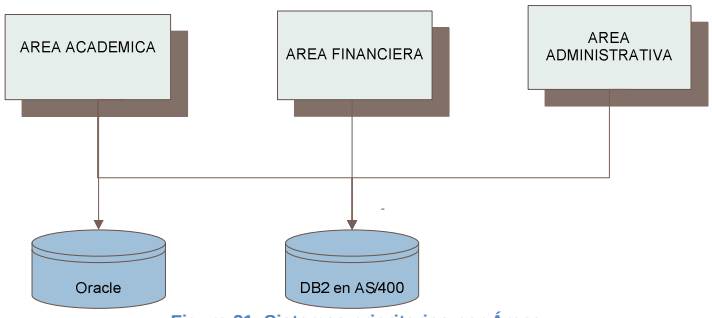


Figura 21. Sistemas prioritarios por Áreas

Tomando como base el listado de sistemas a cargo del departamento de desarrollo informático, y visualizando la necesidad de compartir información entre varias dependencias, los sistemas considerados prioritarios para el estudio, son:

Área	Sistema
Área Financiera	Sistema de Contabilidad y Presupuesto
	Sistema de Tesorería
	Sistema de Activos Fijos
	Sistema de Gasto interno
	Sistema de Gestión Socio Económica
Área Administrativa	Sistema de Gestión de Personal
Área Académica	Sistema Académico Operando al momento
	Sistema Académico en Desarrollo
	Sistema de Bibliotecas

Tabla 13 Sistemas prioritarios para SOA

Cada uno de estos sistemas ha sido construido en diversas situaciones, momentos, personas y con variadas herramientas utilizadas en Departamento de Desarrollo Informático, para revisar el detalle de cada uno de estos revisar el *Anexo IV. Detalle de Sistemas del DDI*

3.1.3. Arquitecturas empleadas en el DDI

Tomando como fundamento a la Arquitectura de Software como una disciplina científica, para la determinación de un conjunto de paradigmas que llevan a una organización del sistema a un alto nivel, que lleva a la relación de los componentes que lo conforman y los principios que orientan su diseño y evolución; conscientes de la importancia de esta y omitiendo información sobre elementos que no pertenecen a una interacción, abstrayendo detalles de elementos que no afectan a como estos serán utilizados y reconociendo que elementos interactúan con otros mediante interfaces que contienen detalles sobre partes públicas y privadas, se clasificará a los sistemas desarrollados en el DDI en ciertos estilos arquitectónicos.

Existen varios estilos de arquitecturas basados en grafos de tuberías (pipes) y filtros, las arquitecturas de pizarras basadas en un espacio compartido de datos o los sistemas en capas tal y como ha sido descrito en el capítulo uno, en donde se describe el marco teórico de estilos arquitectónicos. Debido a que los sistemas que operan en el Departamento de Desarrollo Informático

comprenden más de una estructura, hay que encontrar la definición de una familia de sistemas en términos de un patrón de organización estructural o un "estilo", este modelo semántico ayudará a determinar propiedades del todo a partir de las partes, además de definir el vocabulario de tipos de componentes, conectores y las restricciones de combinación.

3.1.3.1. Análisis de los estilos arquitectónicos utilizados en el DDI

En la primera parte del marco teórico de la presente tesis, concretamente en la sección 1.1.4.Estilos arquitectónicos, se describieron los diferentes estilos arquitectónicos empleados en la construcción de sistemas, en base a estos, en este punto haremos una descripción de cuáles son aplicables al departamento informático de la Universidad de Cuenca.

Para asignar los sistemas a cargo del DDI a un estilo arquitectónico, y conociendo que estos sistemas no contienen la arquitectura explícitamente documentada, se hará un pequeño análisis en relación a lo que comprende la arquitectura.

- En los estilos de flujo de datos tal como en una arquitectura tubería y filtro, los elementos son estructuras con entradas y salidas, no comparten estados con otros elementos, se pueden correr procesos concurrentes, especialmente usado cuando no existe interactividad, los datos se pasan en forma completa a los filtros, a este tipo de arquitectura no se ajusta ninguno de los sistemas priorizados en el DDI
- En una arquitectura de pizarra, los componentes operan independientemente, en función del depósito de datos como es el caso de los sistemas de Tesorería, Contabilidad y Presupuestos en donde el depósito de datos o la pizarra compartida está constituida por la base de datos en DB2, y los componentes están constituidos por las opciones de los menús, pues al seleccionar una opción directamente se procesan las instrucciones contra el repositorio común de datos.
- En la arquitectura basada en eventos, hay que encontrar componentes que centran su atención en eventos, cuando este se produce el sistema lo comunica a los procesos relacionados. Los componentes tienen interfaces que ofrecen conjunto de procedimientos y de eventos. En

lugar de invocaciones a procedimientos, se anuncia uno o más eventos y otros componentes registran el interés en un evento asociando un procedimiento a dicho evento. Un ejemplo de este tipo de estilo arquitectónico es un sistema que opera en el web, reaccionando a solicitudes mediante el protocolo http como es el caso del sistema de control de gasto interno. En donde se tienen procesos que anuncian eventos y reaccionan a estos, no se conocen que componentes están afectados y no se puede determinar el orden de procesamiento.

- En el departamento de desarrollo informático existen otras aplicaciones que están en el caso de poseer arquitecturas basadas en capas, donde cada capa provee servicios a la capa superior y es servido por la capa inferior, considerando que los componentes son cada una de las capas y los conectores son los protocolos de interacción entre las capas, la restricción la constituye la interacción que está limitada a las capas adyacentes, es el caso de los sistemas de activos fijos, académico anterior, sistema académico en desarrollo, gestión de personal.
- Sistemas heterogéneos, formando estilos híbridos como se puede apreciar en el nuevo sistema de biblioteca, en el que el sistema se conforma a su vez de un sistema existente, más un sistema que está en adaptación, una interface web, más servicios que darán solución a necesidades puntuales de la biblioteca, también es el caso del sistema de gestión socio económica, que se forma de una combinación de programación en capas, programación con componentes y servicios.

3.1.3.2 Clasificación de sistemas en estilos arquitectónicos

Del análisis anterior de los sistemas realizado en el punto anterior, podemos observar que las aplicaciones a cargo del DDI seleccionados se adaptan a los siguientes estilos:

Sistema	Estilo Arquitectónico	Componentes	Repositorios de Datos
Contabilidad y Presupuesto	Arquitectura de Pizarra	• Archivos de nombres, Archivos de Mayor, Subcuentas/Auxiliares, Ingresos, Gastos • Control movimientos contables • Informes contables, Control de Bancos • Almacén Universitario • Proforma presupuestaria de Gastos e Ingresos • Organización del archivo de nombres	• DB2 en AS/400

		• Clasificación del archivo para proforma • Distributivo de sueldos a proforma • Actualización y Consulta para proforma • Informes Presupuestarios	
Tesorería	Arquitectura de Pizarra	• Control de bancos • Edición de Documentos • Control de Especies	• DB2 en AS/400
Activos Fijos	Arquitectura de Capas	• Control de inventario de activos fijos • Levantamientos físicos • Actas, Ordenes, Consultas • Usuarios, Dependencias	• DB2 en AS/400
Gasto interno	Arquitectura de Capas	• Gestión de Dependencias • Gestión de Contabilidad • Gestión de la Bodega General • Gestión de Proveeduría	• DB2 en AS/400
Gestión Socio Económica	Arquitectura Heterogénea	• Gestión de configuración • Gestión de Cuentas • Gestión de Matrícula • Gestión Socio Económica	• DB2 en AS/400 • Oracle • Servicios Web
Gestión de Personal	Arquitectura de 2 Capas	• Administración del sistema • Gestión de personas • Gestión de servidores • Nómina	• DB2 en AS/400 • Oracle
Académico Anterior	Arquitectura de Capas	• Subsistema Autentificación • Subsistema Carreras • Subsistema Control Materias Dictadas • Subsistema Entidades • Subsistema Horarios • Subsistema Labores Universitarias • Subsistema Materias Dictadas • Subsistema Paralelos • Subsistema Pénsums • Subsistema Periodos Lectivos • Subsistema Reglamentos • Subsistema Reglas • Subsistema Acceso Datos •	• DB2 en AS/400
Académico en Desarrollo	Arquitectura de 2 Capas	• Archivos Generales • Carreras • Planes de Carreras • Mallas Curriculares • Inscripciones • Oferta de Asignaturas • Sílabos • Matrículas • Calificaciones • Certificaciones	• Oracle
Biblioteca	Arquitectura Heterogénea	• Portal WEB del CDJBV • Sistema ABCD • Repositorios de información bibliográfica • Sistema D-Space. • Servicios especializados	• MySQL • ISIS

Tabla 14. Estilos Arquitectónicos en sistemas del DDI

En la tabla podemos distinguir cada uno de los sistemas estudiados junto a la arquitectura bajo cuya concepción fue construido, se observa los subsistemas que lo componen y los repositorios de datos de cada aplicación; de la tabla se puede observar que las arquitecturas predominantes en los sistemas son los que tienen organización por capas.

3.2. Análisis de brecha entre la situación actual y la deseada

En este punto y considerando que uno de los principales aportes de la tesis es el de proponer una arquitectura que permita la integración de los sistemas ya existentes, se procederá a describir la situación actual, posteriormente se detallará la situación deseada, para finalmente realizar un análisis de la brecha existente.

3.2.1. Situación actual

En la Universidad de Cuenca, las Facultades y Departamentos colaboran y comparten información por correo electrónico, sitios Web y sistemas informáticos que no están integrados, carecen de una plataforma que les permita visualizar como un único sistema informático el acceso a cualesquier sistema que necesiten. Los sistemas desarrollados a cargo del departamento informático no disponen de una documentación completa dentro de sus diseños que defina de una manera unificada la arquitectura de los sistemas desarrollados, los encargados de efectuar cada proyecto han tratado de esquematizar lo que sobreentienden por arquitectura, es en los últimos sistemas desarrollados a cargo del departamento a partir del año 2009 en donde se tiene algunos documentos plantilla que sirven de modelo para presentar y desarrollar nuevos proyectos, aunque no constituyen algo definitivo a seguir.

El problema con el que se han enfrentado los sistemas desarrollados a cargo del DDI, es que han observado un contraste entre lo planificado e implementado, los sistemas han partido a cargo de modelos esquematizados con la metodología de desarrollo de software RUP, pero el momento de hacer la implementación en la plataforma de desarrollo .NET, han tenido que hacer cambios que no han quedado documentados, por citar un problema en RUP se establece un esquema en clases para hacer las capas, pero al construirlas en

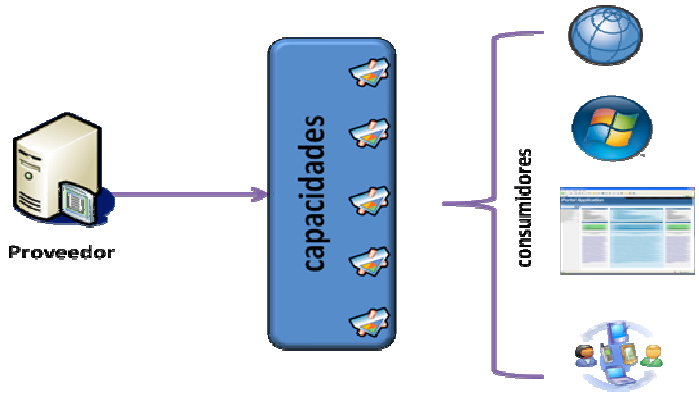
.NET por dificultades en programación se necesita cambiar las clases y en la práctica llega a tenerse dos clases, en la capa de interface se embebe la capa de negocios para facilitar el desarrollo y sacarle potencialidad a la herramienta de implementación. Por ejemplo en un formview se puede insertar, modificar, eliminar cierta tabla omitiéndose la capa de negocios pues es en el mismo formview el encargado de implementarlo, se incluyen las 3 capas utilizando los objetos que ofrece .NET, se optimiza al heredar los objetos que tiene .NET evitando crear propios objetos para realizar las acciones de creación, modificación, eliminación. Para aprovechar las ventajas que proporciona la herramienta de programación que ha seleccionado el DDI, no se tiene módulos encapsulados que podrían ser componentes comunes para varios sistemas, toda esta situación ha llevado a tener sistemas altamente acoplados, considerando que cada nuevo sistema será nuevamente reprogramado.

Debido a los grandes tiempos que necesita el personal del departamento informático, para lograr cubrir la curva de aprendizaje que requieren otros lenguajes como Java, que no se han venido empleando en el DDI, no se tomo la decisión de emplear plataformas tecnológicas alternativas al software propietario de Microsoft.NET, y debido a la falta de experiencia del personal en estas herramientas se optó por utilizar vistas a la base de datos para solucionar necesidades de acceso a datos compartidos entre sistemas.

3.2.2. Situación deseada

La arquitectura orientada a servicios (SOA) desarrollada a finales de los 90, es un marco de trabajo conceptual que permite a las organizaciones unir los objetivos de negocio con la infraestructura de TI integrando los datos y la lógica de negocio de sus sistemas separados. El concepto de SOA como tal consiste en lograr estructurar los procesos de negocios y los activos de tecnología de manera modular e intercambiable, creando módulos flexibles y seguros con una arquitectura basada en estándares definidos. La arquitectura orientada a servicios pretende conseguir unir las tecnologías de información con los requerimientos de negocio, superando la dificultad de mantener tecnologías y bases de datos diferentes, coexistiendo incluso con servicios tan propios de .net con intercambio de objetos de esta plataforma como datasets, que no pueden enlazarse a otras plataformas como java por ejemplo.

El paradigma propuesto por SOA define un sistema de información en donde todas las funciones u objetivos del negocio se definen como servicios, estos servicios son independientes y su ubicación resulta irrelevante, dado que sus interfaces están definidas y publicadas. La dificultad para los servicios que son proporcionados por las aplicaciones disponibles es identificarlos y hacerlos accesibles, ya que las aplicaciones originalmente no se pensaron para



interactuar con otras o la tecnología en la que se crearon no solventaba esta necesidad. Por lo tanto se plantea definir varios servicios orientados a crear tareas compartidas por varios sistemas, como en es el caso del sistema de recursos humanos, en el que para recuperar datos de personas se propone pasar de la situación actual en la que se crean vistas a la base de datos, se otorga los permisos necesarios y se proporciona la ruta para acceder a los datos directamente y leer o alterar las tablas y campos requeridos a transformar esas tareas propias de una organización académica, a que sean transformadas en servicios o funciones propias que hace el negocio de la Universidad de Cuenca, visualizándolos como los servicios que presta la universidad, para enlazarlos, integrarlos y generar los servicios a alto nivel requeridos, como se presenta en la figura:

Figura 22. Capacidades ofrecidas mediante servicios web (de Hernández, 2008)

A partir de la visión de que en un futuro se pueda acceder mediante internet, con un login único a cualquier sistema desarrollado dentro de la Universidad, la visión es plantear un modo de vincular los sistemas desarrollados mediante una capa de abstracción que pueda operar sobre cualquier sistema y que permita "hablar" y compartir recursos ya programados, de manera que se optimice el tiempo y recursos de programación, logrando integración de aplicaciones mediante servicios web, lo que permitirá que otras piezas de funcionalidad hagan uso de otros servicios de manera natural, sin importar su ubicación física. Así cada sistema evoluciona con la adición de nuevos servicios, cada uno evolucionando de una manera independiente.

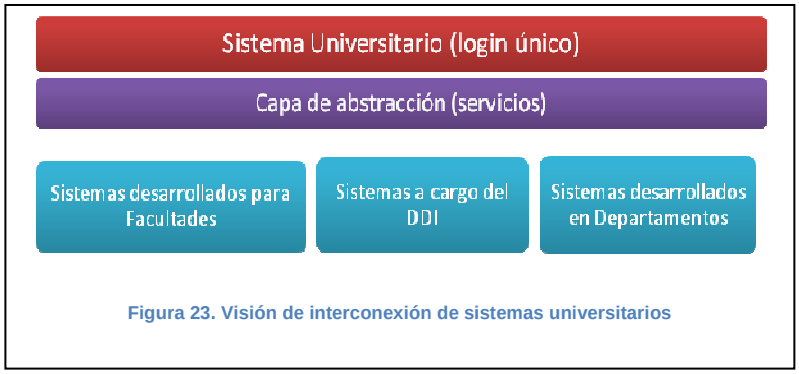


Figura 23. Visión de interconexión de sistemas universitarios

3.2.3. Análisis de brecha

Al determinar la madurez del área de desarrollo de software del Departamento Informático de la Universidad de Cuenca al compararlo con el estándar CMMI, se determina que se encuentra en el nivel denominado *Inicial*, donde hay procesos improvisados y caóticos, sin calendarios ni estimados precisos de costos, donde la funcionalidad y calidad del producto es imprevisible; en ciertos puntos está tratando de escalar hacia el nivel denominado *Repetible*, en el cual los procesos relacionados a la gestión de proyectos y gestión de requerimientos son manejados de alguna manera para su repetición en proyectos distintos, aunque lo ideal es llegar al nivel denominado *Definido* con procesos para actividades administrativas y técnicas estandarizados y documentados, totalmente definidos y disponibles para todos los miembros de una organización; la realidad es que muchos de los procesos y más aún de los reglamentos no se encuentran estandarizados peor aún documentados.

Al profundizar en el estudio de las aplicaciones desarrolladas a cargo del departamento de desarrollo informático, se puede observar que el desarrollo de software dentro del DDI, concibe arquitecturas centradas en las aplicaciones, y

se nota claramente la existencia de funcionalidad repetida en los sistemas. Muchos sistemas actualmente están operando sin conocer la concepción misma de servicio funcionando sobre el web con duplicación de datos y problemas de inconsistencia de los mismos, ya que existe información base como el listado e servidores universitarios y otra información compartida entre la mayoría de procesos que se dan en la universidad, que se la encuentra repetida en otras bases de datos.

Ocurre que cada vez que el área de desarrollo de software crea complejas aplicaciones, en cierto momento del desarrollo el sistema necesita funcionalidades ya antes implementadas, aunque lo ideal reutilizar la funcionalidad ya construida, en la práctica es una tarea difícil de realizar pues los sistemas no fueron diseñados para integrarse, existe incompatibilidad en la forma en la que se intercambian datos entre plataformas y/o tecnologías diferentes. Entonces se opta por re-implementar la funcionalidad requerida, como está ocurriendo en el caso de la re-implementación del nuevo sistema académico, que aunque implica más tiempo de desarrollo es la opción más fácil y segura. Al escoger esta opción de reimplementación, trae las consecuencias:

- En varias aplicaciones hay funcionalidad replicada
- Al haber varias conexiones desde sistemas que dependen de estos para su funcionamiento, existe dificultad al migrar a nuevas tecnologías
- Al no existir una estrategia de integración de aplicaciones, se generan múltiples puntos de falla, que pueden fácilmente detener la operación de todos los sistemas muy fácilmente
- Es un modelo muy poco escalable
- Al final se tiene una pobre respuesta al cambio, las aplicaciones siguen siendo concebidas desde el inicio como islas independientes.

La realidad es que la universidad de Cuenca está llevando una arquitectura centrada en las aplicaciones, lo que acarrea el problema de inexistencia de una semántica universal, problemas de integración entre sistemas desarrollados en diferentes plataformas, difícil integración entre diferentes fuentes, va a llevar a que siempre se mantengan “Islas de Datos”, en el siguiente cuadro se puede apreciar una comparación entre el enfoque que se está dando al desarrollo de sistemas vs. el enfoque que se quiere dar con la arquitectura SOA.

	Arquitectura centrada en las aplicaciones	Arquitectura SOA
Diseño e Implementación	Orientado a las funciones	Orientado a la coordinación
	Construido para durar	Construido para cambiar
	Largos ciclos de desarrollo	Construido e instalado incrementalmente
Sistema Resultante	Aplicaciones aisladas	Soluciones Empresariales
	Acoplamiento fuerte	Acoplamiento débil

Tabla 15. Arquitectura Centrada en aplicaciones vs. SOA

La brecha entre los problemas citados anteriormente, que ocurren en las diferentes fases de análisis, diseño e implementación de sistemas y las soluciones ideales que deberían darse a los requerimientos de una institución educativa que congrega a gran cantidad de departamentos, personas con ideales, pensamientos y tecnología diversa será cubierta con ciertos servicios que en una fase inicial podrían ser considerados como los básicos, para posteriormente seguir extendiéndolos a medida que sean necesarios.

En una organización como la Universidad de Cuenca, la aproximación a SOA es posible, con un modelo en el que los servicios ofrecidos por las aplicaciones ayuden a crear una arquitectura que permita la integración de las aplicaciones, promuevan la reutilización de funciones nuevas y ya existentes, eliminando islas de datos aislados y muchas interfaces en la conexión de los sistemas.

3.3. Marco de Integración

Los tópicos presentados en la introducción del marco teórico en el tema 1.2. SOA, se utilizarán como referente para el planteamiento que se presenta a continuación y que resulta finalmente en el marco de integración propuesto al finalizar este tema. Si bien están definido el plan estratégico de la Universidad de Cuenca, se carece de documentos que definan de manera explícita los procedimientos a seguir y se constata que en la práctica que los procesos que se realizan, no concuerdan con ciertos reglamentos definidos, que en muchos casos están desactualizados, por lo tanto no resulta que la metodología top-down sea la mejor opción para descubrir servicios; por lo que la estrategia utilizada es la bottom-up pues es forma de análisis orientada a la necesidad actual cuando los requerimientos provienen de sistemas heredados y los procesos de negocio no están definidos, se trata de identificar los servicios a partir de las funciones que ya están implementadas en las aplicaciones, creando un “wrapper” o envoltorio a las funciones existentes, conservándolas tal y como son para presentarlas como servicios.

Para determinar la manera en la que es posible integrar los sistemas ya desarrollados en el DDI y obtener la arquitectura resultante, utilizamos el paradigma de Programación Orientada a Objetos, y procedemos a determinar los principales objetos del negocio educativo de la Universidad de Cuenca, a continuación se hace una abstracción de los principales procesos que son compartidos por varios objetos y que pueden dar lugar a posibles servicios web, se detalla que haría cada proceso y finalmente se relaciona a los sistemas desde los cuales se puede tomar la fuente de información. Posteriormente se definen los servicios de los cuales estará compuesta la arquitectura planteada, se los clasifica en genéricos y específicos, y se efectúa una coreografía de servicios determinando como se relacionan entre ellos y los repositorios de datos que serán la fuente de la información para cada uno de los servicios.

3.3.1. Objetos de negocio

De los sistemas que ya se encuentran desarrollados se han tomado con un alto nivel de abstracción los objetos más importantes, los cuales se los representa en la siguiente figura, en donde también se puede observar las relaciones de relación y de herencia que se puede entre los principales objetos de negocio:

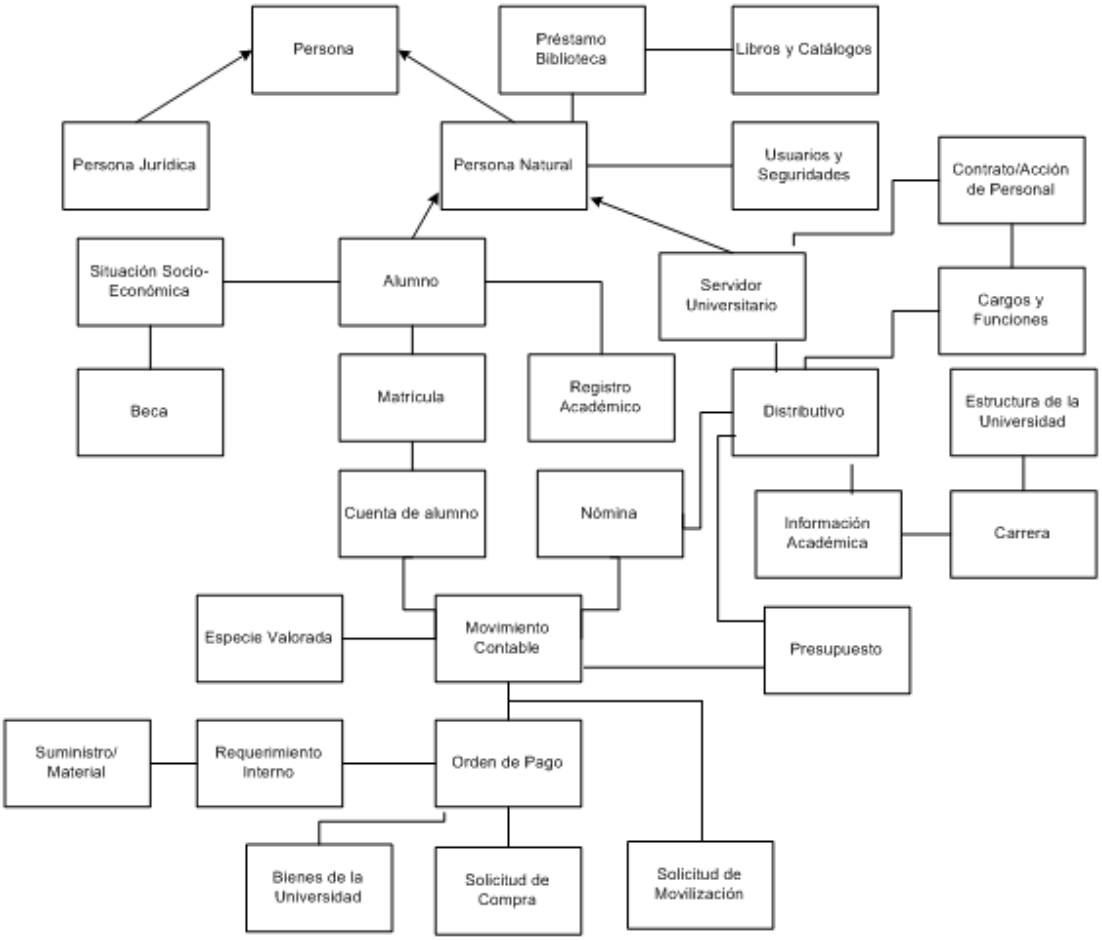


Figura 24. Objetos del negocio en los sistemas a cargo del DDI

3.3.2. Procesos compartidos

Siguiendo con el principio de la orientación a objetos y tomando a cada uno de los objetos del negocio rescatados en la figura 24, utilizando la abstracción y luego de analizar los procesos que posiblemente podrían contener esos objetos, se ha procedido a desechar aquellos que no ofrecerían procesos compartidos por varios entes universitarios. En este punto se trata de detallar los procesos que serán los compartidos por varios sistemas y que pueden convertirse en servicios, que se encuentran inmersos dentro de los principales objetos, los cuales a su vez se encuentran inmersos dentro de las aplicaciones existentes.

Luego del análisis se pudo determinar la existencia de otros objetos que aunque no son parte de los objetos del negocio, son importantes para la implementación de los sistemas, como es el caso del objeto Estructura, que guardaría el esquema de organización de la Universidad y sería compartido por todos los sistemas universitarios.

Procesos compartidos organizados por Objetos

OBJETO	PROCESOS	SISTEMAS INVOLUCRADOS
PERSONA	VERIFICA_PERSONA: Verificación de tipo de persona alumno, docente, empleado, trabajador EDITA_PERSONA: Creación, Actualización, Inactivación de Personas REVISAR_ESTUDIANTES: Listado de estudiantes por período y carrera REVISAR_ADMINISTRATIVOS: Listado de empleados y trabajadores por dependencia REVISAR_DOCENTES: Listado de docentes por periodo, departamento y carrera	**SGP.** Sistema de Gestión de Personal **SGA.** Sistema de Gestión Académica
DISTRIBUTIVO	REVISAR_DISTRIBUTIVOS: Datos de los distributivos de servidores universitarios, asignados por período, dependencia, departamento o servicio académico DISTRIBUTIVO_DOCENTE: Proceso para revisión y edición de los datos de los distributivos asignados a los docentes en un período, se encarga de la parte académica, materias, horarios, paralelos, entre otros. NOMINA: Datos del distributivo de los servidores universitarios, dividido por un factor que indica el tipo de servidor. DATOS_LABORABLES: Detalle de los datos laborales de una persona que trabaja en la Universidad	**SGP.** Sistema de Gestión de Personal **SGA.** Sistema de Gestión Académica
SOCIO ECONOMICO	CALCULO_MAT_PREGRADO: Cálculo de matrícula diferenciado de pregrado calculado en base a la ficha socio económica para calcular la puntuación, se calcula en base al número de matrícula, por período y carrera. CALCULO_MAT_SERVICIO: Cálculo del valor de matrícula de tipo fijo en el caso de un servicio que puede ser: Postgrados, Talleres, Seminarios, Cursos; por período, según el id del servicio y un parámetro que indica si es diferible en cuotas. FICHA_SOCIOECONOMICA: Datos del estudio socioeconómico por alumno en un período ESTADO_CUENTA: Cuotas fijadas para alumnos en una carrera o curso, incluye deudas y pagos realizados en un período BECAS: Reporte de las Becas asignadas a un alumno en un período	**SGSE.** Sistema de Gestión Socio Económica **ADMINUC.** Sistema de Administración global de la Universidad de Cuenca
ESTRUCTURA	REVISAR_ESTRUCTURA: Detalle de la estructura de la Universidad REVISAR_UNADEPENDENCIA: Listado de subdependencias de una dependencia EDITAR_LOCALIDADES: Datos de espacios físicos universitarios	**ADMINUC.** Sistema de Administración General

OBJETO	PROCESOS	SISTEMAS INVOLUCRADOS
REGISTRO_ ACADEMICO	PLAN_CARRERA: Revisión del plan de una carrera que consta de datos informativos, modalidad de estudios, nivel, duración, número de materias, malla curricular, número de créditos, horas de práctica preprofesional; proporcionando toda o parte de la información, de acuerdo a un filtro de selección REVISAR_MATRICULAS: Listado de matriculados por carrera REVISAR_UNA_MATRICULA: Detalles de matrícula de un alumno por período PADRON: Padrones electorales para ser editados y depurados ESTADISTICAS_CALIFICACIONES: Reporte de notas de estudiantes útiles para planificación, promedios de estudiantes, número de aprobados y reprobados	**SGA.** Sistema de Gestión Académica
PRESUPUESTO	REVISAR_PRESUPUESTO_DEP: Proceso que informa presupuesto asignado por dependencia REVISAR_PRESUPUESTO_NOMINAS: Proceso para la revisión del monto presupuestado y el valor consumido en total para los rubros: Totales_Sueldos, Totales_IESS, Totales_Fondos de reserva, Totales_Decimos.	**SCGI**. Sistema de Gasto Interno **SGP.** Sistema de Gestión de Personal
BIENES	REVISAR_ACTIVOS: Proceso para la revisión de bienes a cargo de una persona o una dependencia para revisión a cargo de los custodios REVISAR_BIENES: Proceso que informa el listado de bienes por dependencia	**SACF.** Sistema de Activos Fijos
CONSUMOS	CONSUMOS: Revisión de bienes consumidos por período, por dependencia y por ítem(puede ser: Requerimiento Interno, Solicitud de compra, Orden de Pago, Viático) REVISAR_PARTIDAS: Revisión de las cuentas contables asignadas por el gobierno que constituyen las partidas asignadas a una dependencia REVISAR_FONDOSADMIN: Revisión de los fondos administrados asignados a una dependencia. ESTADO_CUENTA: Revisión por dependencia de las cuentas asignadas a la misma, la cantidad gastada y la cantidad pendiente por consumir.	**SCGI.** Sistema de Gasto interno
BIBLIOTECA	REVISAR_CATALOGO: Listado de libros, revistas, tesis, folletos disponibles en la biblioteca organizados por tipo VERIFICAR_PRESTAMO: Revisión de estado de préstamos	**SBIB.** Sistema de Bibliotecas

Tabla 16. Procesos clave organizados por objetos

3.3.3. Enlace con las Áreas funcionales

El objetivo de esta sección es hacer un análisis de los procesos que son considerados comunes para las áreas funcionales universitarias, manteniendo el enfoque SOA de orientación al negocio, sin la intensión de efectuar un análisis empresarial; más bien se visualizando el posible enlace que podría existir entre los sistemas existentes respecto a las áreas de la Universidad de Cuenca. Se propone a continuación el siguiente modelo presentado en la siguiente figura, que enlaza las áreas funcionales que son parte de las áreas fundamentales para la organización universitaria, con los sistemas bajo estudio en el DDI mediante los objetos universitarios o entidades empresariales que vienen a ser una abstracción de los principales objetos detectados en los sistemas y presentados con detalle en la tabla anterior.

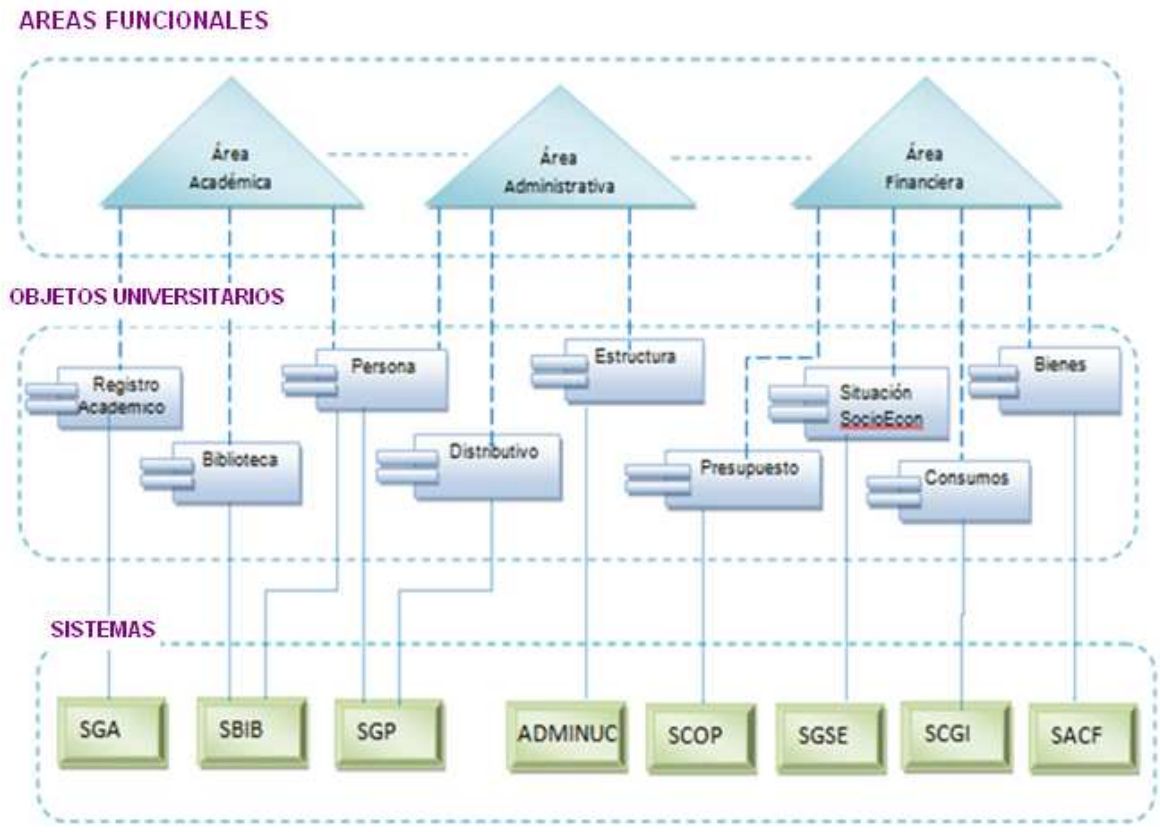


Figura 25. Entidades Empresariales

Los procesos expuestos anteriormente en la tabla 16, al ser organizados de acuerdo a los objetos empresariales y a las áreas funcionales, dan origen a una estructuración del modelo en Servicios, que serán clasificados ulos en Servicios Genéricos y Servicios Específicos.

Capítulo 4. MODELO DE INTEGRACION

4.1. Modelo de Integración basado en Servicios

En el presente capítulo nos enfocaremos a presentar el modelo propuesto para la integración de aquellos sistemas ya implementados, la propuesta es que puedan comunicarse empleando el concepto de servicios; posteriormente en el capítulo cinco, presentaremos una metodología de desarrollo de software, que podría utilizarse como referencia para los nuevos sistemas que se desarrollen a futuro en el Departamento de Desarrollo Informático incorporando el concepto de la Arquitectura SOA para la Universidad de Cuenca.

Utilizando el concepto de lo que es un servicio desarrollado en el marco teórico presentado en el capítulo primero, planteamos el siguiente modelo de integración, que será descrito a lo largo del capítulo.

El modelo de integración propuesto en la figura 26, presenta la manera en que las principales aplicaciones a cargo del DDI podrían integrarse mediante servicios genéricos y servicios específicos, los cuales a su vez se enlazan con los repositorios de datos tanto internos como externos al Departamento de Desarrollo Informático. La integración de sistemas se realiza mediante servicios generales y servicios específicos, cada servicio de tipo general está relacionado con uno o más servicios específicos, los cuales son responsables de tener procesos que permitirán exponer la información requerida por el servicio.

En el nivel superior se puede observar las aplicaciones existentes en el departamento y que son tomadas para el estudio, en el segundo nivel vemos una capa de abstracción de los servicios genéricos, los cuales son los objetivos y funciones principales del negocio universitario, físicamente pueden resultar en servicios web de nivel superior, o pueden ser directorios de trabajo o paquetes, la forma en que finalmente resulten implementados dependerá del entorno de implementación que seleccione el DDI.

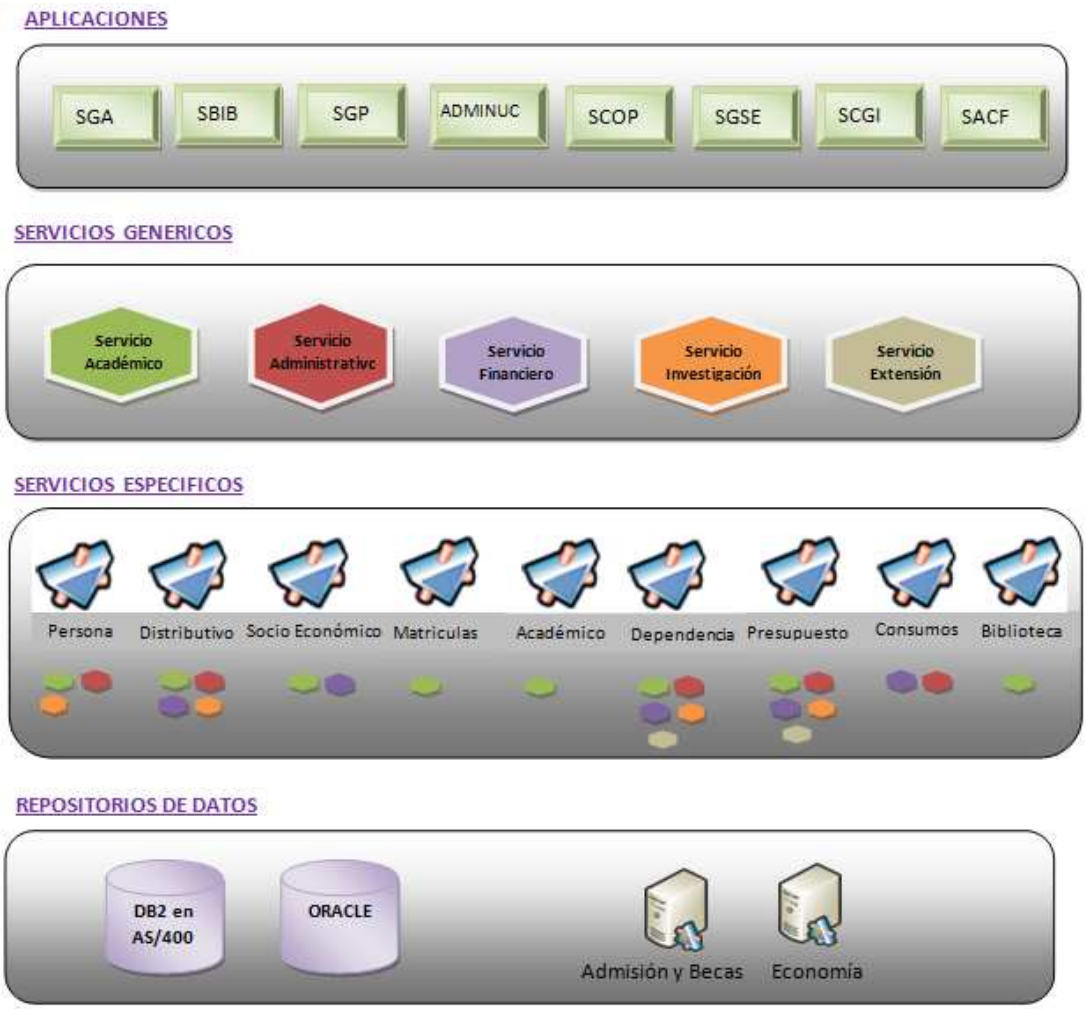


Figura 26. Modelo de Integración

El tercer nivel del esquema corresponde a los servicios web denominados específicos, que son los que corresponderán a servicios implementados como tales. Bajo cada uno de ellos se puede observar resaltado con diferentes colores, el enlace a los servicios genéricos que corresponderían. Es una especie de orquestación que podrá ser utilizada para otorgar acceso y permiso a ciertas aplicaciones que de acuerdo a los servicios genéricos que correspondan.

En el último nivel vemos las dos fuentes principales o repositorios de datos hacia los cuales los servicios tendrán acceso. También se puede observar las dos fuentes de datos externos al departamento que son las fuentes provenientes de las dependencias Admisión y Becas y Economía, las cuales no proporcionan un directo acceso a la base, más bien la comunicación es mediante servicios web, intercambiando datos en formato XML.

4.1.1. Servicios Genéricos

Los servicios genéricos representan los objetivos del negocio y en el caso de la Universidad de Cuenca corresponden a las secciones administrativa, financiera, de investigación y de extensión hacia la comunidad, tal y como se describió en la sección 3.1.1. Debido a que no se va a realizar un análisis empresarial, sino que más bien lo que se desea es encontrar una correspondencia entre los objetivos del negocio que pueden ser posteriormente representados como servicios de nivel general o podrían llegar a ser directorios de servicios participantes en las soluciones de integración en las que intervienen los diferentes sistemas universitarios y son abstractos, es decir no existen en una implementación real no son particulares a una aplicación o proyecto, más bien son generales a todo el modelo empresarial. Los servicios que se proponen como genéricos son los expuestos en la siguiente figura:

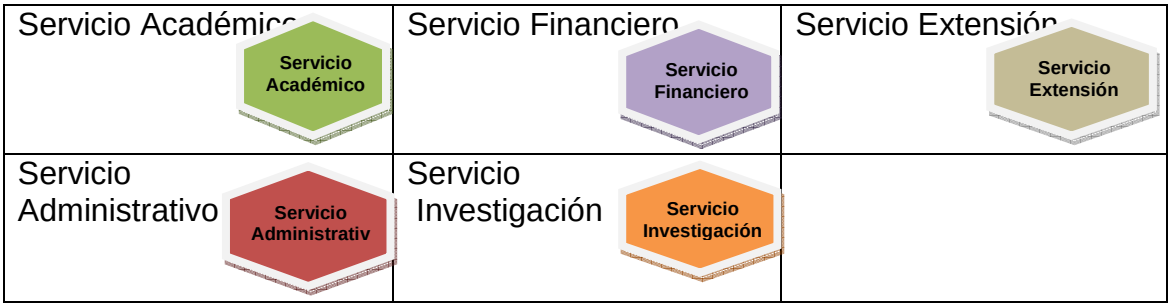


Figura 27. Servicios genéricos

4.1.2. Servicios Específicos

Los servicios específicos son las visiones o implementaciones particulares originados en base a las aplicaciones que se han destacado anteriormente de los servicios genéricos correspondientes y mediante los cuales las aplicaciones desarrolladas en el Departamento de Desarrollo Informático llegarán a integrarse.

Los servicios:

Figura 28. Figura que representa un servicio específico

- **Servicio Estructura**
- **Servicio Persona**
- **Servicio Distributivo**
- **Servicio SocioEconomico**
- **Servicio Matriculas**
- **Servicio Carrera**
- **Servicio Presupuesto**
- **Servicio Bienes y Consumos**
- **Servicio Biblioteca**

Se originan de los objetos empresariales descritos anteriormente y se los organizará de acuerdo a su funcionalidad, luego de un análisis minucioso de los objetos y los procesos que contienen, se ha llegado a la conclusión de que para ser transformados en servicios se puede fusionar algunos objetos revisados inicialmente en el esquema 24, como es el caso de los objetos *entidades* y *consumos* que pueden fusionarse en un solo servicio, y el caso del objeto *registro_académico* que da origen a varios servicios y ha sido dividido en los servicios *matrículas* y *carrera*.

Estos servicios descritos, tendrán los siguientes procesos:

No. Referencia	Servicio	Procesos	Descripción
SR1	ESTRUCTURA	• REVISAR_ESTRUCTURA • REVISAR_UNADEPENDENCIA • EDITAR_LOCALIDADES	Procesos que describen y actualizan la estructura y organización general de la Universidad de Cuenca
SR2	PERSONA	• VERIFICA_PERSONA • EDITA_PERSONA • REVISAR_ESTUDIANTES • REVISAR_ADMINISTRATIVOS • REVISAR_DOCENTES	Contiene los procesos necesarios para verificar o editar los datos de las personas que laboran en la Universidad de Cuenca.
SR3	DISTRIBUTIVO	• REVISAR_DISTRIBUTIVOS • DISTRIBUTIVO_DOCENTE • DISTRIBUTIVO_SUELDOS • DATOS_LABORABLES	Datos laborales que se organizan durante cada período lectivo para los servidores universitarios docentes o administrativos
SR4	SOCIOECONOMICO	• CALCULO_MAT_PREGRADO • CALCULO_MAT_SERVICIO • FICHA_SOCIOECONOMICA • ESTADO_CUENTA, BECAS	Procesos que afectan a los estudiantes que acceden a cualquier servicio universitario, incluye el registro socio económico, becas, cálculo del costo de matrícula, cuotas, pagos
SR5	MATRICULAS	• REVISAR_MATRICULAS • REVISAR_UNAMATRICULA • PADRON	Reportes relacionados a matriculados en las diferentes carreras en la universidad de cuenca
SR6	CARRERA	• PLAN_CARRERA • ESTADISTICAS_CALIFICACIONES	Datos académicos comunes a varias dependencias universitarias
SR7	PRESUPUESTO	• REVISAR_PRESUPUESTO_DEP • REVISAR_PRESUPUESTO_NOMAS	Reportes relacionados a datos presupuestados y consumidos dentro de cuentas universitarias
SR8	BIENESYCONSUMOS	• REVISAR_ACTIVOS • REVISAR_BIENES • CONSUMOS • REVISAR_PARTIDAS • REVISAR_FONDOSADMIN • ESTADO_CUENTA	Información referente a bienes y consumos asignados a las diferentes dependencias universitarias
SR9	Biblioteca	• REVISAR_CATALOGO • VERIFICAR_PRESTAMO	Procesos necesarios para revisión de datos en el centro documental CDJBV

Tabla 17. Procesos de los servicios propuestos

4.1.3. Relación entre servicios

Debido a que los servicios genéricos representan una organización lógica que ayudan al enlace de los sistemas informáticos existentes con los servicios específicos que constituyen en la realidad los que deberán ser implementados sobre las aplicaciones existentes en el DDI, con lo que se conseguirá el objetivo de la Arquitectura SOA en la Universidad de Cuenca; en la siguiente tabla se hace una relación entre los servicios genéricos y específicos propuestos, considerando que algunos servicios específicos forman parte de varios servicios genéricos.

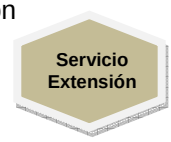
Servicios Genéricos	Servicios Específicos	Servicios Genéricos	Servicios Específicos
Académico 	Estructura Persona Distributivo SocioEconomico Matriculas Carrera Presupuesto Bienes y Consumos Biblioteca	Investigación 	Estructura Persona Distributivo Presupuesto Bienes y Consumos
Administrativo 	Estructura Persona Distributivo Presupuesto Bienes y Consumos	Extensión 	Estructura Persona Carrera
Financiero 	Estructura Distributivo SocioEconomico Matriculas Presupuesto Bienes y Consumos		

Figura 29. Relación de servicios genéricos y específicos

4.2. Esquema de Integración

Las aplicaciones de misión crítica deben ser envueltas en servicios web para crear servicios de negocio de los sistemas existentes y crear nuevas funcionalidades, la idea es ir creando la arquitectura SOA aprovechando las aplicaciones e inversiones ya implementadas, que pueden ser expuestas en forma de servicios web, para posteriormente ir ampliando nuevas funcionalidades gradualmente, promoviendo la reutilización y obteniendo agilidad en los procesos.

A continuación se propone el siguiente esquema en donde se presentan las aplicaciones existentes y se definen los servicios de los cuales estará compuesto el sistema universitario, sus interacciones y desde qué sistemas tomamos información. Los servicios generales de investigación y extensión no serán considerados para el planteamiento, debido a que si bien son parte de los objetivos del negocio, aún no existen sistemas ya implementados que puedan proporcionar los objetos específicos y servicios web necesarios para integrar esos servicios de nivel general.

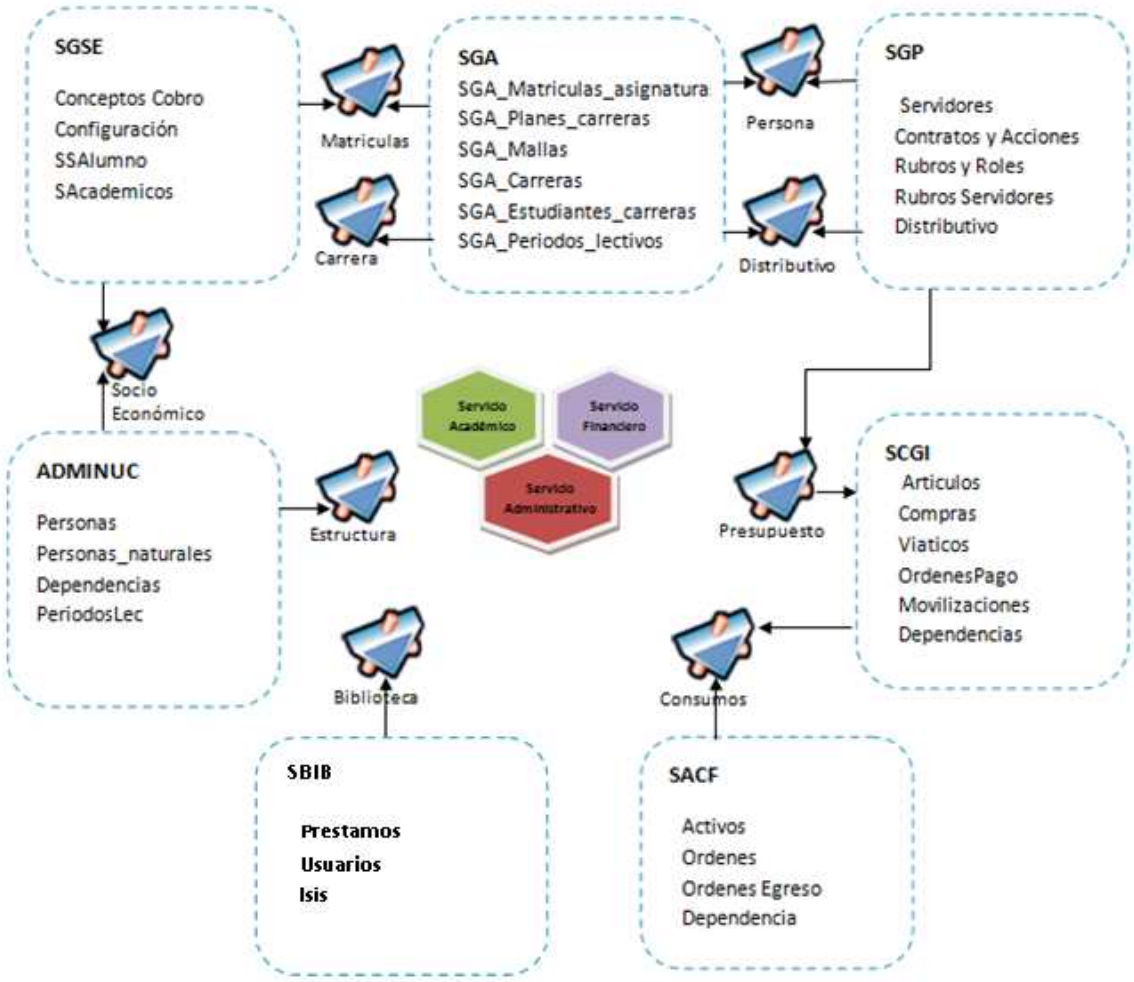


Figura 30. Esquema de Integración mediante servicios web

4.3. Integración con la red universitaria

El siguiente esquema muestra como se encuentra interconectada la red de datos universitaria, sus tipos de enlaces, equipos, etc. Para que el consumidor

y el proveedor de los servicio interactúen, los servicios deben estar disponibles de ser localizados y consumidos en servidores dentro de la red interna.

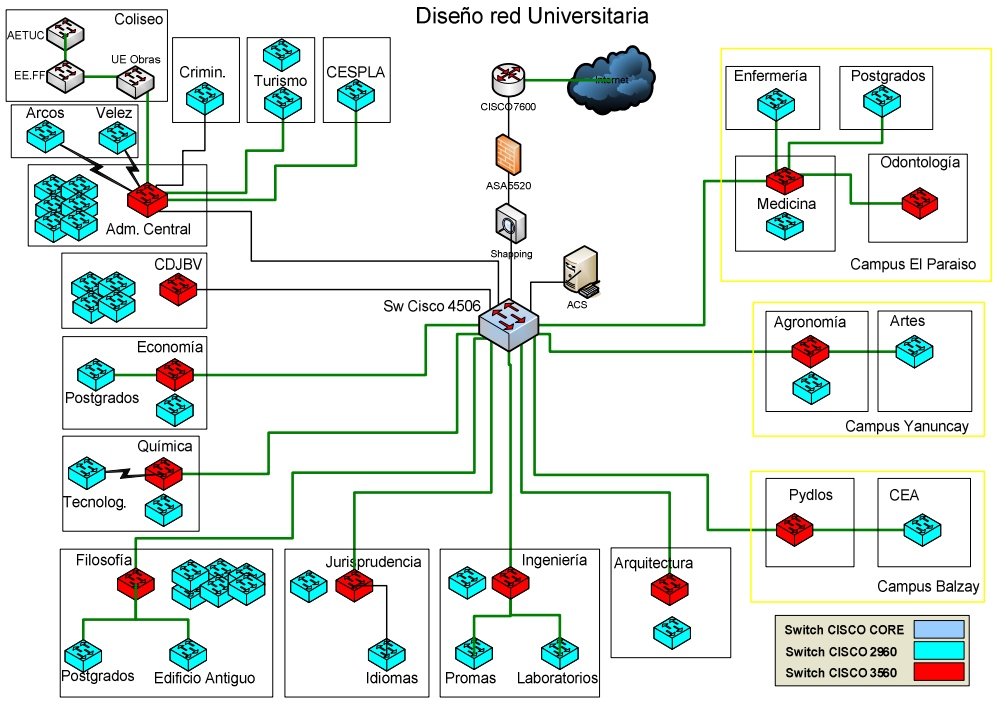


Figura 31. Diseño de la red Universitaria. Tomado de (Mejia, y otros, 2009)

Esta red universitaria tiene una topología en estrella extendida, con un equipo central (CORE), de donde salen enlaces a cada una de las facultades y un servidor proxy para cada una. Para el acceso a internet se tiene un firewall PIX, de donde se deriva la sección para la DMZ en donde residen los servidores Web y de mail. El direccionamiento de la red de datos universitaria, tiene una red pública clase C que entrega el proveedor 184.186.46.0[4], que llega al router de borde, de allí se tiene una red interna clase C (174.8.X.X)[1] que es distribuida para todos los equipos de backbone, es decir, para firewall, proxy, y CORE, del cual se salen todos los enlaces a cada una de las facultades y dependencias. Cada facultad posee un proxy que da una red interna a los hosts de esa facultad (184.186.X.X)[1]. (Mejia, y otros, 2009)

[4] Por motivos de seguridad, el direccionamiento no es el real

En la red universitaria está el servidor que accede a la DMZ y a su vez este accede a los servidores de la red interna, en el gráfico se puede visualizar el núcleo de acceso y se puede observar la DMZ, dentro de la cual se procederá a ubicar los servicios web en el mismo equipo destinado para las aplicaciones web. Otros procesos pueden ser visibles desde el mundo exterior con la seguridad necesaria, en etapas posteriores de implementación de servicios.

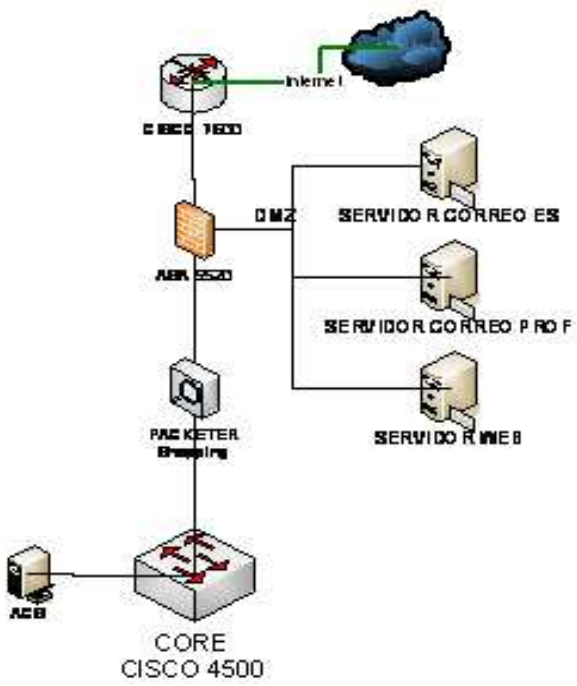


Figura 32. Servidor web dentro del Core Universitario (tesis Mejía, Choglo , 2009)

Los servicios citados para el modelo SOA, pueden ser internos y externos.

Los únicos procesos que pueden considerarse como visibles hacia el mundo exterior para la Universidad de Cuenca, son dentro del servicio PERSONA, el proceso VERIFICA_PERSONA, y dentro de DEPENDENCIA, el proceso REVISAR_ESTRUCTURA, para lo cual deben ubicarse en el servidor ubicado dentro de la DMZ o zona desmilitarizada universitaria con tiene acceso desde Internet con el URL: 12.0.1.X/24[5] que internamente es traducido a 184.186.46.X/24 [2], por ahora los servicios propuestos estarán ubicados dentro de la red de servidores internos con el URL: 174.8.1.0/25

[5] Por motivos de seguridad, el direccionamiento no es el real

4.4. Herramientas de Integración

Los sistemas informáticos desarrollados a cargo del DDI han seguido la orientación tecnológica planteada por las herramientas propietarias de Microsoft, debido a muchas causas como la curva de aprendizaje necesaria para los ingenieros que se encontraban laborando en el departamento, versus la premura de tiempo con la que normalmente se necesitan los sistemas, se verifica que la mayoría de sistemas prioritarios para el DDI, están desarrolladas con la plataforma .NET en las versiones 2003 y 2005, por lo que los servicios web necesarios para integrar los sistemas pueden ser desarrollados desde esta plataforma.

La orientación a futuro y la visión de los Directivos del Departamento Informático es migrar todas las aplicaciones hacia la base de datos ORACLE y utilizar otras tecnologías relacionadas con el lenguaje Java, con el objeto de aprovechar mejor esta base de datos, considerando que será el centro de datos hacia el cual se piensa migrar finalmente todas las aplicaciones anteriores y en base a la cual se desarrollarán los nuevos sistemas informáticos.

Hasta conseguir que todas las aplicaciones se encuentren desarrolladas en una tecnología diferente, debe conseguirse visibilidad e intercambio de información entre los sistemas existentes, la fuente desde la cual se realizará la integración serán los sistemas desarrollados casi en su totalidad en el lenguaje Visual Basic, considerando como repositorio central las bases de datos que reposan en el servidor AS/400.

Se plantea por lo tanto que los sistemas se integren mediante la exposición de servicios web creados en las diferentes versiones de .NET y tomando directamente como fuente de datos los repositorios que utilizan cada una de las aplicaciones, incluso sin alterar el código base de cada una de las aplicaciones. Los servicios que a continuación se describen como necesarios para la integración deben crearse paralelos a la aplicación y con su propio código que realice las operaciones descritas que en su mayoría constituyen reportes o actualizaciones a tablas generales

Las tecnologías más representativas en el mercado local, que permiten construir aplicaciones empresariales son J2EE y .NET; si bien J2EE está basado en la plataforma que gestiona la infraestructura distribuida y apoya a los servicios web interoperables que permiten soluciones empresariales utilizando herramientas open source y es en la actualidad el modelo de componentes más implementado en el mercado mundial, la plataforma .NET de Microsoft es la aplicación utilizada en el Departamento de Desarrollo Informático. Debido a que la mayoría de sistemas desarrollados en el DDI están desarrollados bajo la plataforma .NET, la Arquitectura Orientada a Servicios planteada se basa en los elementos de la pila de tecnologías de Microsoft, desde las herramientas de desarrollo para crear servicios Web como .NET a productos de servidor, como BizTalk Server y Microsoft Office SharePoint Server, donde se produce la ejecución posterior de los servicios Web al conectar y orquestar servicios, y finalmente en las aplicaciones compuestas que consumen servicios Web. Dos de las principales tecnologías de servidor para la orquestación de Procesos, preparados por Microsoft son BizTalk Server y Microsoft Office SharePoint Server

BizTalk Server

BizTalk Server es un producto de servidor que permite la integración de sistemas propuesto como complemento a las tecnologías de desarrollo .NET Framework 3.0, el núcleo de la arquitectura de BizTalk Server se basa en XML, .NET Framework; y es compatible con los estándares abiertos en los que se basan los servicios Web. Una solución que utiliza BizTalk puede consumir servicios Web y exponer los procesos de negocio (orquestaciones de BizTalk) como servicios Web. BizTalk se constituye como la capa de gestión que organiza servicios Web, controlando el flujo e interacciones agregando los servicios individuales dentro de una solución compuesta de nivel superior.

BizTalk Server también permite la integración de aplicaciones y sistemas que no son compatibles con los servicios Web, empleando una gran variedad de adaptadores se puede hacer que las funcionalidades de sistemas y aplicaciones antiguas queden disponibles a los procesos internos de las organizaciones. BizTalk Server se integra con Microsoft Office SharePoint

Server. Juntos, BizTalk Server y SharePoint facilitan la creación de soluciones de procesos de negocio. SharePoint permite a los profesionales de la información recopilar y gestionar datos de negocio (mediante la captura de datos en XML, estructurados y no estructurados). Biztalk Server, en este caso, actúa como el punto central de orquestación para los procesos de gran envergadura, que abarcan tanto a sistemas de información como a personas.

Microsoft Office SharePoint Server

Microsoft Office SharePoint Server 2007 está diseñado para optimizar la forma en que las personas interactúan con los contenidos y los procesos dentro de las organizaciones y a través de ellas. Office SharePoint Server permite aprovechar las ventajas de los workflows para automatizar y mejorar la visibilidad de las actividades de negocio habituales, como la revisión y aprobación de documentos, el seguimiento de incidencias y recogida de firmas. Su integración con aplicaciones conocidas de cliente, correo electrónico y navegadores Web simplifica la experiencia de usuarios. Los usuarios finales pueden definir y modelar con facilidad sus propios procesos aplicando herramientas de Microsoft muy familiares. Office SharePoint Server contribuye a eliminar procesos manuales de gestión de la información, los formularios electrónicos se pueden utilizar para recoger información que luego se puede integrar en los sistemas de línea de negocio en archivos documentales, pueden servir para iniciar procesos de workflow o enviarse a servicios Web.

Además de las citadas, existen algunas herramientas por las cuales puede realizarse la integración entre plataformas J2EE y .NET citadas en (Rodríguez, y otros, 2006)

- IBM Web Service Toolkit (WSTK).- Provee un runtime environment (tanto del lado del cliente como del servidor) demos, ejemplos y algunas herramientas adicionales para diseñar y ejecutar aplicaciones basadas en Web Services. Trabaja como un plug-in para servidores de aplicaciones J2EE, los servidores soportados son IBM WebSphere y Web Sphere MicroEdition, Jakarta Tomcat. Estos servidores ya tienen soporte para Web

Services, pero el WSTK le agrega nuevas funcionalidades y aplicaciones ejemplo. Aparte de incluir todos los estándares como SOAP, WSDL, UDDI, el WSTK provee un conjunto de utilidades extra que ofrecen entre otras cosas servicios de identificación, notificación, medición de uso de Web Services, contratación de Web Services y servicios de acceso a datos similares a JDBC basados en Web Services.

- GLUE.- Provee una plataforma tanto para desarrollar como para consumer Web Services para la plataforma Java. Soporta y es compatible con los estándares XML, SOAP, WSDL, UDDI. Incorpora una API muy completa que disminuye la complejidad del desarrollo y publicación de web services. Este producto es desarrollado por la empresa The Mind Electric, se destaca por su facilidad de uso, su confiabilidad y robustez

- Java XML Pack.- Es un paquete provisto por Sun que incorpora un conjunto de APIs para XML y APIs estándar para servicios web, las APIs provistas soportan WSDL, UDDI, SOAP y XMLP (XML protocol) el cual es una especificación provista por W3C para la interacción entre objetos remotos basada en XML. El paquete se basa en cuatro APIs, JAXM para mensajería en general basada en SOAP, JAXP para procesamiento de datos en formato XML, JAXR para registro de servicios pues soporta UDDI y JAX-RPC que permite hacer invocaciones a web services basándose en SOAP y con soporte WSDL.

En el siguiente capítulo dentro del planteamiento de la metodología de desarrollo de software orientada a implementar SOA para las aplicaciones que se encuentren en desarrollo, y específicamente en el Anexo de Servicios, explicaremos los formatos de intercambio de datos planteados, además de definir los componentes e interfaces que conforman cada servicio para exponer su funcionalidad gobernados por contratos, que definen el conjunto de mensajes soportados, su contenido y las políticas aplicables.

Capítulo 5. METODOLOGIA RASDUC

En el presente capítulo se procede a definir la metodología de desarrollo de software aplicable a los nuevos sistemas que se desarrollen en el Departamento de Desarrollo Informático de la Universidad de Cuenca; la metodología propuesta se ha denominado RASDUC con el significado metodología Rup Agil orientada a SOA para el DDI de la Universidad de Cuenca.

Al inicio del capítulo se hace una revisión de las metodologías de desarrollo de software que se han empleado en el DDI, luego se realiza una comparativa entre las metodologías ágiles y tradicionales para seleccionar el enfoque más adecuado para el Departamento de Desarrollo Informático y finalmente se procede con la exposición de la metodología describiéndola desde el punto de vista de disciplinas y fases de las que se compone.

5.1 Selección de metodologías

5.1.1 Introducción

Las técnicas y metodologías de desarrollo de sistemas se confunden con frecuencia con el concepto de ciclo de vida; la diferencia está en que mientras que el propósito del ciclo de vida es planear, ejecutar y controlar el proyecto de desarrollo de un sistema al definir las fases y las tareas esenciales, sin importar el tipo o magnitud del sistema a construir (Fitzgerald, 1994); una metodología puede seguir uno o varios modelos de ciclo de vida. El ciclo de vida indica qué es lo que hay que obtener a lo largo del desarrollo del proyecto pero no cómo hacerlo; mientras que la metodología indica cómo hay que obtener los distintos productos parciales y finales.

No existe un Modelo de Ciclo de Vida único para todo proyecto, cada proyecto debe seleccionar el ciclo de vida que sea el más adecuado para su caso, tampoco existe un modelo de ciclo de vida ad-hoc para la integración de los procesos de negocio en aplicaciones con arquitectura orientada a servicios. (Lopez, y otros). SOA es una arquitectura en la cual una aplicación se constituye de servicios que se exponen y servicios que se consumen, haciendo énfasis en el bajo acoplamiento entre los componentes de software; puesto que

en los proyectos SOA, los procesos se centran en torno a la lógica de negocio a través de la tecnología, una metodología de desarrollo de software que implemente un enfoque a SOA diferirá de las tradicionales metodologías utilizadas en el Departamento de Desarrollo que no consideraban el planteamiento de servicios, en que se ocupa de ejecutar el análisis del negocio en paralelo al diseño de servicios y desarrollo de los mismos, necesitando un papel más activo en la definición del diseño conceptual de la lógica de la solución.

5.1.2 Metodologías utilizadas en el DDI

El personal del Departamento de Desarrollo Informático ha propuesto elaborar algunos sistemas planteándose diversas metodologías, en el Anexo VI se describen con detalle las metodologías propuestas para los sistemas que se han desarrollado en el DDI. El siguiente cuadro resume las fases y actividades propuestas para la elaboración de los proyectos de desarrollo de software

Sistema	Año	Ciclo de Vida	Metodología	Fases	Entregables	Seguimiento y Control
SGF. Sistema de Gestión Financiera	2006	Cascada	Tradicional[6]	Análisis de requerimientos y Documentación de procedimientos	Manual de procedimientos Especificación de Requerimientos de Software	
				Diseño del Sistema	Prototipos validados con usuarios finales	
				Programación	Elaboración de módulos del sistema Pruebas de Calidad Manuales del sistema	
				Pruebas	Instalación del sistema Plan de pruebas Ajustes	
				Transición en paralelo	Instalación de módulos Capacitación a usuarios Migración de sistemas anteriores	
SGSE. Sistema de Gestión Socio Econom.	2008	Cascada	Tradicional	Requerimientos Análisis Diseño Implementación Pruebas Transición	Al final de cada bimestre se plantea redactar un informe de actividades y hacer una revisión al final de cada flujo de trabajo y cada etapa.	Registro de las versiones y revisiones de cada elemento del proyecto
SGP. Sistema de Gestión de Personal	2007	Cascada	Tradicional	Análisis de requerimientos Documentación de	Documento de Especificación de Requerimientos del Sistema Definir el proceso	

[6] Entendemos por metodología tradicional a la desarrollada en una sola iteración

				procedimientos	óptimo y las necesidades de información	
				Diseño del sistema	Prototipos validados con usuarios finales	
				Programación	Elaboración de módulos del sistema Pruebas de calidad Manuales del sistema	
				Pruebas	Instalación del sistema Plan de pruebas Ajustes	
				Transición	Instalación de módulos Proceso de capacitación	
SGA. Sistema de Gestión Académica	2008	Incremental	RUP	Fase de Inicio	Documento de visión Plan de desarrollo de software Modelado del Negocio Modelo de casos de uso y especificaciones Lista de riesgos Glosario	Gestión de Requisitos Control de Plazos Control de Calidad Gestión de Riesgos Gestión de Configuración
				Fase de Elaboración	Modelo de casos de uso y especificaciones Lista de riesgos y caso de negocio revisados Modelo de datos preliminar	
				Fase de Construcción	Modelos completos (casos de uso, análisis, diseño y datos) Riesgos mitigados Construcción de módulos correspondientes a la primera etapa Plan de proyecto para la fase de transición Manual de usuario.	
				Fase de Transición	Prototipo operacional	

Tabla 18. Metodologías empleadas en el DDI

En el cuadro anterior podemos observar los diferentes ciclos de vida y metodologías utilizadas en los principales proyectos que se han planteado dentro del Departamento de Desarrollo Informático, el planteamiento metodológico ha sido tomado de los proyectos que han sido documentados.

Detallando los sistemas citados:

- Se ha observado que para el sistema SGF no se hizo más que el análisis de requerimientos, solo se llego a hacer una presentación de lo que sería el sistema, pero nunca se concreto la idea, se pensaba comprar un sistema que se acople a las necesidades pero debido a los cambios del gobierno, lo que se ha hecho es parchar el sistema de control de gasto interno, SCGI para que sirva como auxiliar de contabilidad, aunque ese no era el propósito real de ese sistema.
- El sistema SGSE inicialmente fue basado en la metodología planteada por los ingenieros de sistemas del Departamento Informático, pero no fue llevada concretamente en la práctica, puesto que los desarrolladores del sistema en su tesis de grado plantean un ciclo de vida en cascada con modelación en el lenguaje UML que es la que finalmente se realiza.
- En el sistema de gestión de personal, se trata de seguir la metodología propuesta, pero por cuestiones de tiempo y urgencias en asuntos financieros, se realiza directamente la programación, sin concretar el documento planteado inicialmente.
- El sistema que ha seguido una metodología más formal es el sistema académico SGA, que aún está en desarrollo, se pretende seguir con la parte de documentación que por cuestiones de tiempo no ha podido llevarse a cabo como inicialmente se la ha propuesto, no han podido ser elaborados todos los artefactos planteados, por ejemplo en el caso de los riesgos, solamente se los llegaba a determinar de forma verbal, pero no quedaban documentados, Se tiene modelos iniciales que no han sido actualizados, excepto los básicos como los de clases que están actualizados, pero hay otros documentos UML que no están al día.

El Departamento de Desarrollo Informático no dispone de una metodología estándar para ser aplicada en todos los proyectos que se proponen a cargo de esta dependencia, no se tienen documentos estándar ni plantillas en los que los desarrolladores se basan. En los sistemas descritos en la tabla 18 acontece una situación al realizar la propuesta de desarrollo del sistema, y otra situación es la ocurrida en la práctica, pues no se sigue ni la metodología propuesta, ni se realizan los artefactos propuestos por falta de tiempo en el desarrollo del

mismo. Esta situación ha sido descrita en las reuniones de trabajo mantenidas con el personal del Departamento y puede comprobarse al solicitar documentación de las metodologías y artefactos de los sistemas anteriores, los cuales no existen.

5.1.3 Comparativa de Metodologías Agiles y Tradicionales para el DDI

La noción de una metodología propone un proceso disciplinado, con el ánimo de hacer del desarrollo de software una tarea más predecible y eficiente. Tradicionalmente conocemos que las metodologías de desarrollo de software hacen referencia a un *conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevo software.* Aunque el objetivo es claro, en la vida práctica no se conoce que una metodología sea seguida al pie de la letra y que sea exitosa al ciento por ciento. El principal inconveniente es que la mayoría de metodologías están compuestas de fases burocráticas, que llevan a incluir demasiados procesos que en muchos casos son inútiles o que no llegan a revisarse ni en una segunda ocasión.

Para eliminar esta problemática, se han planteado metodologías “Agiles” que tratan de proveer la cantidad de procesos estrictamente necesarios, tratando de conseguir la menor documentación para cierta tarea. Estas metodologías ágiles, difieren de las metodologías tradicionales en que son adaptativas antes que predictivas, útiles en situaciones de constante cambio, mientras que los métodos tradicionales tratan de llegar al detalle en procesos que no cambian en largo tiempo; las metodologías ágiles son orientadas a las personas y las metodologías tradicionales se orientan a procesos. (Fowler, 2000)

A pesar de que el Coordinador del área de desarrollo de software manifiesta que la metodología que utiliza el DDI es RUP debido a que los últimos sistemas desarrollados lo han realizado con el esquema propuesto bajo el proceso unificado, resulta que esta metodología no ha sido utilizada en todos los proyectos de desarrollo de software, principalmente debido al poco tiempo que se dispone para realizar una documentación completa como se establece en el proceso unificado y a la falta de personal que se dedique a esta tarea. Además el proceso unificado requiere que el equipo del proyecto se centre en identificar

los riesgos críticos en una etapa temprana del ciclo de vida, los riesgos de cada iteración, en especial de la fase de elaboración, donde los riesgos principales deben ser considerados primero, contrario a la manera en que el departamento informático está trabajando, sin que los riesgos sean lo primero a tratar.

Los métodos tradicionales para el Departamento de Desarrollo Informático resultan muy estructurados, estrictos, lentos e inconsistentes con el proceso de desarrollo que en la práctica realiza el Departamento, hay que considerar la necesidad de una agilidad en los métodos tomando en cuenta que una de las motivaciones del software orientado a servicios es crear software de manera ágil y adaptada a las necesidades del usuario, y debido al hecho de que las aplicaciones basadas en servicios se construyen por medio de la integración dinámica de servicios que deben ser levemente acoplados.

El análisis realizado al inicio de la presente tesis al tener un acercamiento al área de desarrollo de software del DDI, detectó que se minimiza los riesgos. (Ver sección 2.1.4. Acercamiento al área de desarrollo, para más detalles). Justamente las metodologías de tipo desarrollo ágil tratan de minimizar los riesgos desarrollando software en cortos lapsos de tiempo, por tanto creemos que una metodología que deberá plantearse para el DDI deberá ser una de tipo ágil o con componentes de tipo ágil.

Otra razón para utilizar una metodología ágil en el DDI es el enfoque utilizado en estas, en donde se enfatiza las comunicaciones cara a cara en vez de la formal documentación donde se escribe todo, precisamente es lo que necesita el Departamento de Desarrollo Informático pues dentro de un mismo grupo de ingenieros a cargo de un proyecto están: revisores, editores de documentación y de ayuda, diseñadores de los objetivos a conseguir y el director del proyecto; lo que se trata es de que el proceso de formalización de documentación sea más breve que lo propuesto por RUP, es decir la metodología utilizada debe ser menos orientada al documento, exigiendo una cantidad más pequeña de documentación para una tarea dada.

A pesar de lo mencionado anteriormente, que son factores a favor de incluir metodologías ágiles para el DDI, no se podría decir que para el Departamento de Desarrollo Informático este tipo de metodologías son factibles a ser tomadas

en un ciento por ciento. Una de las razones es que metodologías ágiles como XP están diseñadas para aplicarse a grupos humanos pequeños, en proyectos pequeños, pero los sistemas que está desarrollando actualmente el DDI, son proyectos con un alcance al mediano y largo plazo. *(Ver tabla 19 Sistemas en Desarrollo en el DDI)* Otra razón por la que no se podría asumir totalmente una metodología ágil es que esta enfatiza la adaptabilidad antes que en la previsibilidad, cosa que no ocurre en todos los sistemas universitarios, si el proyecto tiene requisitos estables, entonces se está en la posición tener un plan estable y seguir un proceso planeado, la última razón por la que no se podría asumir totalmente una metodología ágil es que en proyectos críticos para la universidad podría subir demasiado el nivel del riesgo, llegando a reprogramaciones tediosas e incluso de grandes funciones por completo, redundando en tiempos y costos extras innecesarios para la Universidad de Cuenca.

Proyecto	Plazo [7]	Tiempo
Sistema Integrado de Gestión Universitaria	Largo	5 años
Sistema Académico	Mediano	2 años
Sistema de investigaciones	Mediano	1 año y medio
Sistema de evaluación interna	Mediano	1 año y medio

Tabla 19 Sistemas en desarrollo en el DDI

5.2 Metodología planteada

5.2.1 Enfoques utilizados

Como aporte fundamental de esta tesis, se plantea como una posible solución una metodología que integre los conceptos tradicionales, algunos principios de las metodologías ágiles más la orientación a servicios con un enfoque en el negocio, que emplea SOA.

[7] Para los plazos se considera: *corto* si el período que cubre es de un año/ *mediano* si es de más de un año y menos de cinco / *largo* si el período que cubre es de más de cinco años (W. Jiménez C., 1982)

La propuesta utiliza como base el proceso unificado RUP tomando las buenas prácticas que se plantean en las metodologías tradicionales más la variante del proceso unificado que toma los conceptos del desarrollo ágil metodología de proceso unificado ágil AUP (Agil Unified Process), desarrollando un proceso con un énfasis en la planeación.

Luego de un estudio detallado de cada una de estas metodologías, hemos visto que lo mejor para el Departamento de Desarrollo Informático es el integrarlas, con el ideal de fusionar sus mejores características, adicional a esto se tratará de incorporar el concepto de reutilización de las funcionalidades básicas del negocio que pueden estar encapsuladas en forma de servicios web.

En el siguiente esquema observamos los enfoques que se utilizarán para la metodología propuesta.

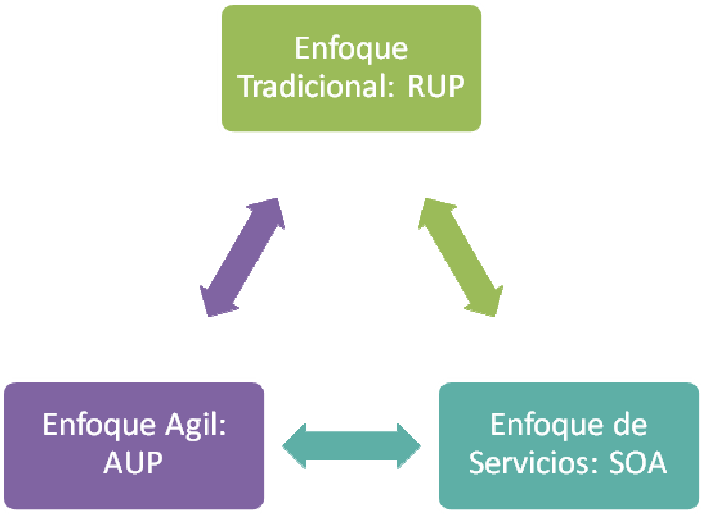


Figura 33. Enfoques de la metodología planteada

La metodología que se planteará a continuación mantiene básicamente los tres enfoques visualizados en la figura anterior, en donde se observa cada enfoque relacionado directamente con los otros, por ejemplo AUP se deriva del enfoque tradicional RUP y permite incorporar SOA dentro de sus actividades; así también los otros enfoques pueden relacionarse entre sí, ya que en ciertos momentos y actividades se puede combinar los principios en los que se basan cada uno de estos; en la figura se ve expresada la relación directa entre los tres enfoques en las flechas que los relacionan y que se pueden observar en la figura con tres colores claramente diferenciados:

- En verde se ve el enfoque tradicional RUP, basado en una metodología de éxito que apoya proyectos orientados a objetos, proporciona disciplinas comprobadas y prácticas. Dentro de RUP hay prácticas que se pueden aprovechar de SOA para mejorar en las áreas de modelos de negocio y análisis orientado al servicio.
- En morado se ve la metodología AUP que teniendo en cuenta la naturaleza cambiante de los requerimientos como lo hacen las metodologías ágiles, utiliza un proceso base adaptado del RUP planteando entregables intermedios para seguimiento y control del proyecto
- En azul se ve el enfoque que incorpora SOA, al considerar la compactación de servicios a la plataforma, construidos sobre una arquitectura. La definición arquitectónica ocurrirá entre los casos de uso y el diseño, lo importante es ejecutar el análisis del negocio en paralelo al diseño de servicios y desarrollo de los mismos.

Aunque existen estudios realizados como los de Solmaz Boroumand,98, quien en su artículo escrito para SOA Magazine, plantea trabajar de forma conjunta SOA con RUP, la realidad del Departamento de Desarrollo Informático de la Universidad de Cuenca no se presta a tener aún servicios web, basados en una seria programación orientada a objetos, realidad que hace que el proceso de incorporación de SOA y servicios web no sea aplicable de la manera en la que se plantea en varios artículos de otras realidades. Haciendo necesaria la fusión de los conceptos antes descritos para la metodología presentada a continuación.

Finalmente y antes de proceder con la descripción de la metodología, en la siguiente tabla se realiza un análisis de la situación que presenta el desarrollo de software en el DDI sobre algunos factores, con el objetivo de que en base a ello se pueda recomendar las actividades que la metodología deberá plantear y de esta forma lograr la mejora.

El campo Situación Actual se califica dentro de los rangos: **Muy bajo, Bajo, Normal, Muy alto, Extra alto**

Factor	Situación Actual	Recomendación
Tamaño del producto	Normal	Se visualiza que los proyectos desarrollados deberán orientarse hacia software con cantidades de líneas de código que cada vez son mayores.
Tiempo de desarrollo	Un año y medio	El tiempo promedio de desarrollo cada vez es más corto, mantener un moderado nivel de comentarios en el código, y documentación.
Volatilidad de requisitos	Muy alto	Es importante definir al inicio de los proyectos un documento llamado "Visión del proyecto" que contiene la descripción de los objetivos, la visión del proyecto y las metas a alcanzar con el mismo. Enfatizar la definición y análisis de requerimientos
Ámbito	Reducido al único sistema a desarrollar	Análisis adecuado del contexto, incluir un esquema de la organización y casos de uso claros, de nivel global Identificación de servicios y planeación de cómo se llevarán a cabo la administración y control de los mismos.
Cultura de la organización	- La capacidad Reactiva, reacción posterior a dificultades - No se acostumbra hacer planificaciones previas	Conocimiento de objetivos y difusión de la planificación del proyecto Definición clara de roles y actividades que se efectuarán en el proyecto Solicitar definiciones completas de rendimiento y de interfaz
Problemas frecuentes	- No están definidos los procesos - Alcances diferentes a los inicialmente acordados - Información duplicada	Definir procesos y requerimientos Definir de forma clara la arquitectura de la aplicación, trabajado de forma colaborativa entre el conocedor del negocio y el arquitecto del sistema.
Potenciales riesgos	- Mayores alcances - Rotación de personal - Nuevas disposiciones generadas por el gobierno	Modelar el dominio del proyecto, identificando las principales entidades y sus relaciones Definición clara de roles y actividades que se efectuarán en el proyecto Definir el alcance que tendrá el proyecto, delimitándolo de la forma más clara posible y excluyendo que áreas quedan fuera

Tabla 20 Factores a considerar en la metodología planteada

5.2.2 Metodología RASDUC

En esta sección se detalla una de las principales contribuciones de esta tesis que es la integración de las metodologías propuestas. Luego de que se ha hecho un análisis previo y verificado los puntos que tienen en común, otros

puntos que podrían fusionarse, y aspectos en los que pueden complementarse las metodologías RUP, AUP, a los cuales se les ha incorporado el concepto de Arquitectura SOA, para conseguir los objetivos del negocio centrándonos en la utilización de servicios web; se propone la siguiente metodología a la que se ha procedido a llamar RASDUC y cuyo significado es:

RASDUC: metodología **R**up **A**up con enfoque **S**OA del **D**epartamento de Desarrollo Informático de la **U**niversidad de **C**uenca.

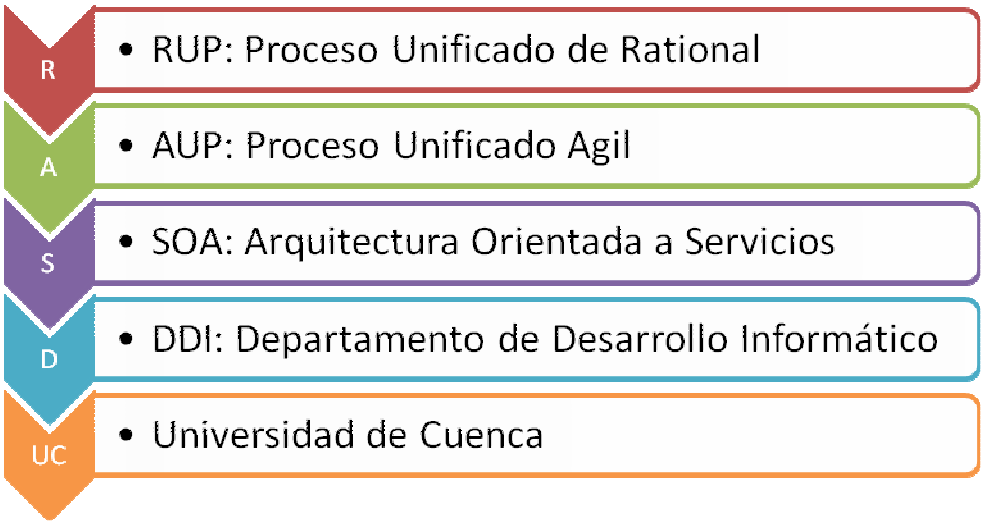


Figura 34. Significado de RASDUC

Aportes individuales de cada concepto

Hemos denominado RASDUC a la metodología planteada, debido a que se forma en base a los conceptos planteados en las metodologías RUP y AUP, más el enfoque de la arquitectura SOA con el objetivo de que sea aplicable a la realidad del Departamento de Desarrollo Informático de la Universidad de Cuenca.

La metodología propuesta al igual que el proceso unificado se basa en los tres conceptos clave: dirigido por casos de uso, centrado en la arquitectura e iterativo e incremental, además se definen actividades, roles y entregables de entrada y salida, así como iteraciones y disciplinas que conforman la metodología. Dentro de las fases de Inicio, Elaboración, Construcción y Transición, se cumplen una o más iteraciones, pues el desarrollo es iterativo e

incremental; en cada iteración se agrega algo más, y el sistema está en continuo crecimiento hasta su entrega. En cada iteración se llevan a cabo las disciplinas y se hace una coordinación de todos los artefactos.

Esta metodología al igual que el RUP, tiene dos dimensiones, el tiempo y las disciplinas; en la dimensión tiempo tiene cuatro fases: Inicial, Elaboración, Construcción y Transición; en la dimensión disciplinas se tiene igual que AUP dentro de las Disciplinas Base: Modelo, Implementación, Pruebas y Liberación y dentro de las llamadas Disciplinas de Soporte: Administración de proyecto, Configuración y Ambiente. El cambio respecto a RUP se visualizará al utilizar el enfoque de desarrollo ágil, en las disciplinas que conforman cada iteración, como puede observarse en la figura 35 que pertenece a AUP y que va a tomar se como base para el planteamiento de la metodología.

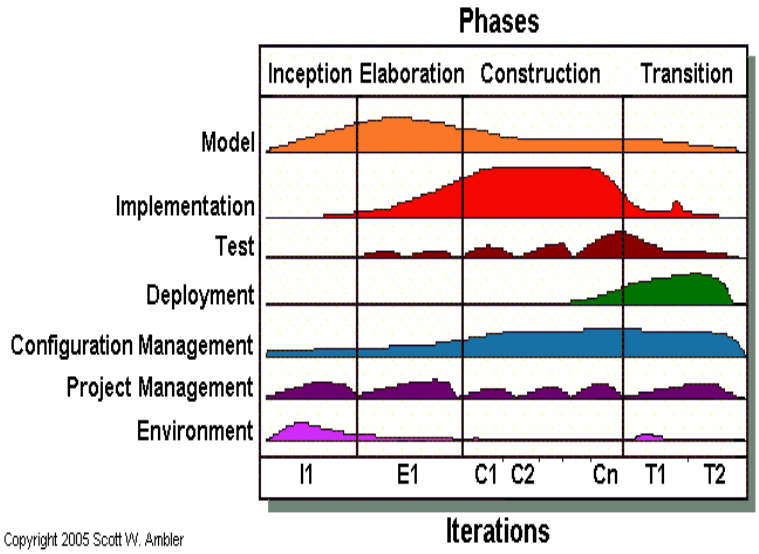


Figura 35.Esquema AUP utilizado para la metodología

En la figura se puede observar las fases que forman la metodología AUP, vemos que al igual que en RUP en la parte izquierda constan las disciplinas, las mismas que para la metodología RASDUC serán clasificadas en disciplinas base y en disciplinas de soporte. En la parte inferior se puede observar que en las fases se pueden realizar iteraciones, en las primeras fases de Inicio y Elaboración se realiza una sola iteración (I1 y E2), para la fase de construcción se puede realizar todas las iteraciones que sean necesarias (C1, C2…Cn) La fase de transición puede ser realizada en 2 iteraciones (T1,T2)

El concepto de Arquitectura Orientada a Servicios interviene en la metodología al introducir el principio mismo de SOA que consiste en determinar el conjunto de servicios tanto de negocios como tecnológicos que interactuando entre ellos

proporcionan la lógica necesaria para construir aplicaciones de una manera rápida; dentro de las etapas, iniciando desde la planeación se agregan elementos específicos para desarrollo SOA, como son las tareas propuestas en el *Anexo Modelo de Servicios.* La modelación del diseño empresarial y el diseño de la solución se la efectuará a través de un conjunto de roles, métodos y artefactos que describen un conjunto de procesos universitarios que lleven a la construcción de servicios, composición e integración de ellos.

5.2.2.1 Incrementos e iteraciones.

El desarrollo incremental genera versiones comenzando con un subsistema funcional pequeño, al cual se le va agregando funcionalidad con cada versión. El desarrollo iterativo entrega un sistema completo desde el principio, y luego cambia la funcionalidad de algún subsistema en cada nueva versión. Ambos enfoques pueden combinarse en un desarrollo iterativo e incremental. Se trata entonces de tener un sistema con ciertas funcionalidades, que se va completando con nuevas funcionalidades en fases sucesivas. Así, el usuario tiene en producción algunas funcionalidades mientras se van desarrollando las otras. Se tiene por lo tanto al menos dos sistemas funcionando en paralelo:

- *El sistema operacional o sistema en producción*, en uso por el usuario, puede ser una implementación parcial o una implementación anterior con funcionalidades nuevas o sustituidas.
- *El sistema en desarrollo*, corresponde a la siguiente versión, que está siendo preparada para reemplazar la versión en producción, que puede aún conservar partes de implementaciones anteriores o faltarle funcionalidades.

Se llamará una iteración, al software desarrollado en una unidad de tiempo, que deberá durar pocas semanas (basándonos en el principio de desarrollo ágil).Una iteración no se enfocará a demasiada funcionalidad para justificar que esté listo el programa para ser presentado al usuario, con una versión sin errores cuando se llegue al final de la iteración.

5.2.2.2 Fases

Fase de Inicio**.-** El objetivo principal es establecer la viabilidad del proyecto, para lo cual se delimita claramente el alcance del sistema propuesto con un

buen análisis de negocio, se trata de tener una visión aproximada, efectuar un análisis del quehacer de la empresa cliente ("el negocio"), determinar alcance del proyecto, estimaciones (imprecisas) de plazos y costos, tratando además de identificar riesgos críticos. En esta fase se trata de encontrar un bosquejo de una posible arquitectura que el ámbito del sistema puede soportar. Especifica una visión global del proyecto a construirse, en la que podemos saber con cierta certeza que es deseable y que es posible desarrollar en el sistema.

Fase de Elaboración.- En esta fase se obtiene una visión refinada, se trata de la resolver los riesgos más altos y lograr una implementación iterativa del núcleo central de la aplicación, luego de recopilar alrededor de un 80% de requerimientos funcionales aún no considerados, analizarlos, diseñarlos y obtener una línea base de la arquitectura sobre la cual se trabajará fases posteriores de construcción y transición. Se completan además detalles de la planificación del proyecto, con las estimaciones más ajustadas de las actividades y los recursos necesarios, llegando a nuevos alcances si es necesario.

Fase de Construcción.- En esta fase se realiza una implementación iterativa guiándose en la línea base de la arquitectura, solucionando los requisitos de los elementos más sencillos y de menor riesgo que lleva a generar un producto beta del sistema. También se debe detallar casos de uso y escenarios restantes a través de los flujos de trabajo fundamentales y las iteraciones que se requieran. Se requiere también hacer la preparación para el despliegue (entrega, instalación y configuración), modificar la descripción de la arquitectura y actualizar los modelos, integrar subsistemas, probarlos, para integrar el sistema y comprobarlo como un todo.

Fase de Transición**.-** En esta fase se implementa la versión actual del producto en su entorno de operación, se realizan las pruebas beta con un pequeño grupo de usuarios que evalúan el comportamiento del sistema de acuerdo a los requerimientos iniciales, se descubren problemas inesperados, se hace una lista de fallas y se hacen correcciones leves, los cambios significativos se dejan para una próxima versión. En esta fase también se descubre omisiones en la ayuda y documentación de usuario, y finalmente se termina con el despliegue.

5.2.2.3 Disciplinas

La metodología se organiza en disciplinas o flujos de trabajo, una disciplina es un conjunto de actividades realizadas en cierto momento del ciclo de vida de una aplicación. Dentro de una disciplina se realizan varias actividades que producen artefactos. Un artefacto es un término general empleado para referirse a cualquier resultado del trabajo, ya sea un texto, un diagrama, una página web, código en lenguaje de programación u otros. En el siguiente gráfico podemos observar que la estructura de la metodología que planteamos es similar al esquema planteado por la metodología AUP.

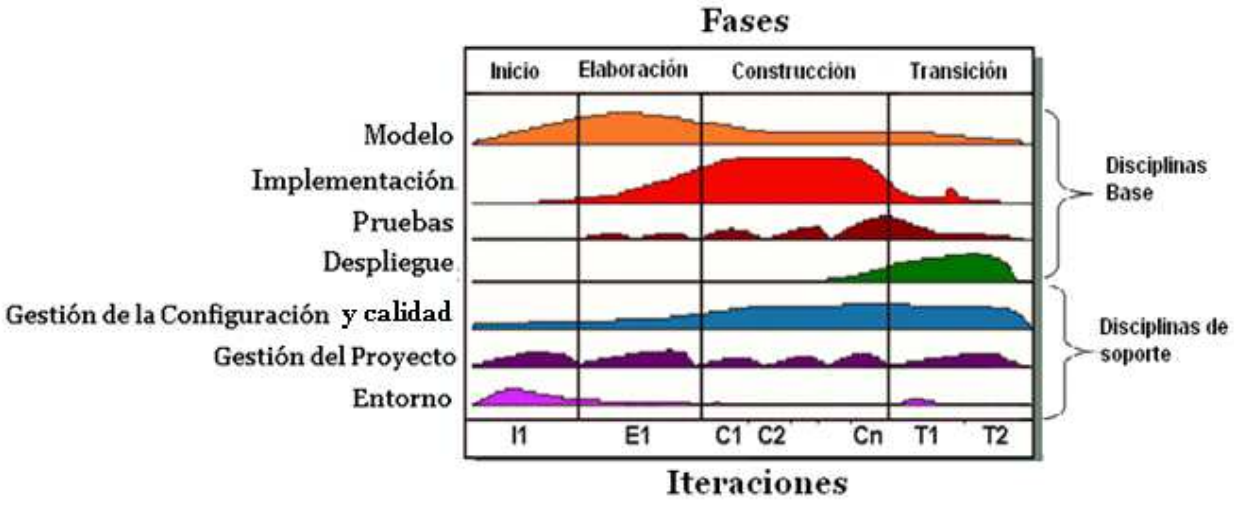


Figura 36 Disciplinas de la metodología RASDUC

En el gráfico anterior y en la parte izquierda se observa el detalle de las disciplinas que forman parte de todo el ciclo de vida, las mismas que se clasifican en Disciplinas Base y Disciplinas de Soporte. Las disciplinas base forman parte del ciclo de vida, mientras que las disciplinas de soporte son actividades de apoyo que estarán presente a lo largo del ciclo de vida de la aplicación.

Disciplinas Base	Disciplinas de Soporte
MODELO IMPLEMENTACION PRUEBAS DESPLIEGUE	GESTION DE LA CONFIGURACION Y CALIDAD GESTION DEL PROYECTO ENTORNO

5.2.2.4 Entregables

El proyecto que utilice esta metodología, tendrá una biblioteca compuesta de los llamados "modelos", los mismos que contienen ciertos entregables, descritos como anexos al final de este documento. En la siguiente figura observamos los principales componentes de la biblioteca que contendrá todo proyecto de la metodología RASDUC:

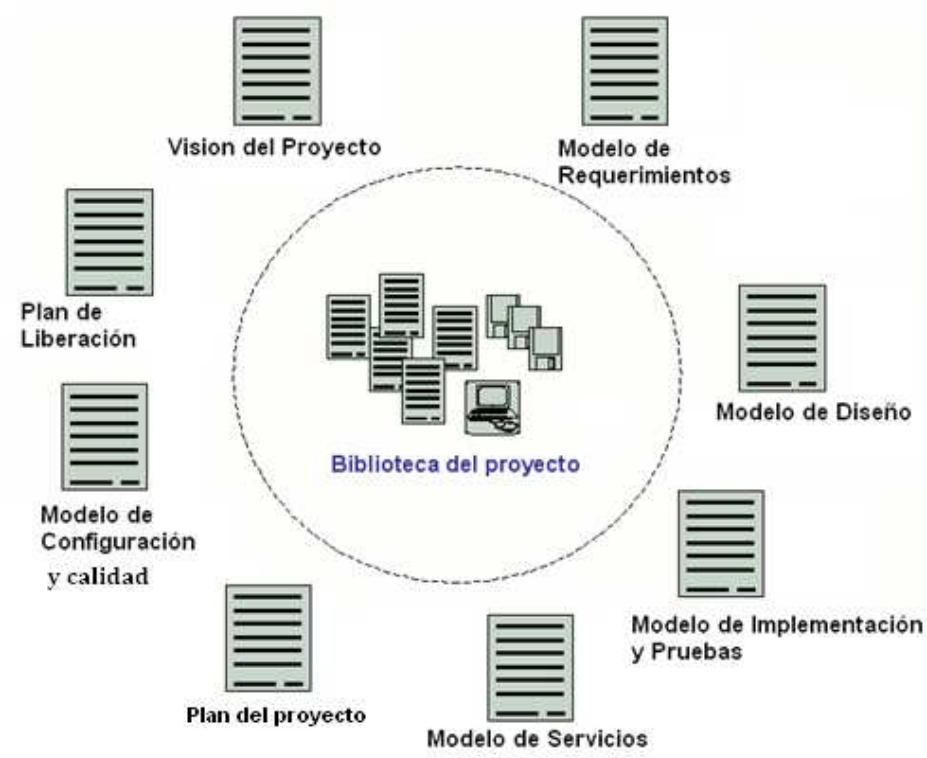


Figura 37. Biblioteca de la metodología RASDUC

Esta metodología plantea ciertos entregables resultantes de cada una de las disciplinas y las fases que posteriormente en la sección 5.3.1, se describen, en los anexos se propondrán plantillas para cada uno de estos. Los entregables propuestos son:

- Visión del proyecto
- Modelo de Requerimientos
- Modelo de Diseño
- Modelo de Desarrollo
- Glosario
- Esquema de interfaces
- Esquema de base de datos
- Código de la aplicación
- Modelo de Pruebas
- Plan de Capacitación

- Modelo de Liberación
- Paquete de instalación
- Sistema en ambientes de preproducción y de producción
- Manuales del sistema
- Material de capacitación
- Modelo de Configuración y Calidad
- Modelo de Planeación
- Plan del proyecto
- Guías y plantillas del proyecto
- Herramientas de trabajo

Para proponer estos entregables, se ha procedido a tomar conceptos, propuestas, plantillas y sugerencias realizadas por expertos como Karl Wigers y otros asequibles en la web, sin embargo la mayoría de entregables se adapta a las actividades propuestas por la metodología que se acopla a las necesidades del Departamento Informático de la Universidad de Cuenca, obteniendo esquemas más amigables, menos burocrátricos, más sencillos y prácticos.

Considerando que no todos los proyectos requieren todos los artefactos, ni con igual grado de profundidad o detalle, la metodología RASDUC propone actividades realizadas en cada una de las disciplinas y diferentes artefactos que se proponen, de los cuales se podrían elegir los de mayor valor práctico para cada proyecto. En la siguiente tabla se sintetiza todas las actividades y artefactos empleados en la metodología y los detalles de los entregables que se emplearán en cada una de las disciplinas se analizarán posteriormente en forma detallada, en la siguiente sección.

Tipo de Disciplina	Disciplina	Actividad	Artefacto
Disciplina Base	Modelo	• Entender el proyecto • Explorar el dominio del proyecto • Modelar la solución y la arquitectura	Visión del proyecto Modelo de Requerimientos Modelo de Diseño Modelo de Desarrollo Glosario

	Implementación	• Desarrollo • Generación de la Base de Datos • Construcción	Esquema de interfaces Esquema de base de datos Código de la aplicación
	Pruebas	• Administración de las pruebas • Verificación • Validación	Modelo de Pruebas Plan de Capacitación
	Liberación	• Planificación de puesta en producción • Distribución del sistema • Documentación del sistema	Modelo de Liberación Paquete de instalación Sistema en ambientes de preproducción y de producción Manuales del sistema Material de capacitación
Disciplina de soporte	Gestión de la Configuración y Calidad	• Establecimiento de la configuración • Administración de la Configuración • Ejecución del Plan de Calidad	Modelo de Configuración Sistema de Administración de la configuración Procedimientos de control de acceso al sistema de administración de la configuración Plan de Calidad
	Gestión del Proyecto	• Establecimiento de estimaciones • Plan del proyecto	Modelo de Planeación Plan del proyecto
	Entorno	• Plantillas de soporte • Herramientas de Soporte	Guías y plantillas del proyecto Herramientas de trabajo Configuración del proyecto

Tabla 21. Actividades y Artefactos por disciplina

5.2.2.5 ROLES

Otro aporte fundamental dentro de la presente tesis es el planteamiento de los roles que intervendrán para cada una de las actividades propuestas en cada una de las disciplinas de la metodología, luego de un análisis en los roles propuestos por RUP y AUP se ha llegado a la conclusión de que no son aplicables en su totalidad al Departamento de Desarrollo Informático.

El DDI es un departamento que tiene reducido personal, con mucha capacidad, y lo que ocurre con frecuencia es que el personal trabaja sobre varios proyectos en paralelo, y no siempre persona hace tareas similares en todos los proyectos, más bien se trabaja de manera colaborativa y de acuerdo a la dimensión del proyecto se le puede asignar a una sola persona todas las actividades del proyecto, si es necesario.

A pesar de que las personas pueden tener inclinación por laborar en un área determinada y poseen experiencia y dominio de ciertos temas, la realidad en nuestro medio es que al personal no se lo contrata como especialista en una sola área, por lo tanto dentro de la metodología se plantea que los roles puedan ser efectuados por una o varias personas, que una sola persona

puede tomar varios roles y un rol no necesariamente representa un puesto de trabajo.

Además los tipos de roles son considerados como de tipo Interno para las actividades que involucran personal que labora dentro del Departamento de Desarrollo Informático; mientras que los roles que son de tipo externo, son aplicables al personal fuera del DDI.

ROL	DESCRIPCION	DISCIPLINA	TIPO / SIMBOLO
Administrador de la Base de Datos	Colabora con el equipo del proyecto para diseñar, probar, mejorar y dar soporte al esquema de datos de la aplicación.	IMPLEMENTACION	Interno 
Modelador	Crea y desarrolla los modelos, en dibujos, en herramientas CASE complejos, trabaja de manera evolutiva y colaborativa. Apoya en la representación de los procesos, plantillas, guías, ejemplos.	MODELO IMPLEMENTACION ENTORNO	Interno 
Administrador de la Configuración	Responsable de proveer la infraestructura y ambiente para el equipo de desarrollo. Responsable de seleccionar, adquirir, configurar y dar soporte a las herramientas necesarias para el proyecto.	GESTION DE LA CONFIGURACION Y CALIDAD ENTORNO	Interno 
Revisor	Responsable de realizar las pruebas del sistema y de las actividades de calidad. Evalúa el producto en el trabajo real, retroalimentando al equipo. Implementa el sistema en entornos de producción.	PRUEBAS DESPLIEGUE	Interno 
Desarrollador	Escribe, Prueba y Construye software	MODELO IMPLEMENTACION DESPLIEGUE	Interno 
Coordinador del Proyecto	Administra y supervisa a miembros del equipo, construye relaciones con las partes interesadas, coordina interacciones con las partes interesadas; planea, administra y asigna recursos, organiza prioridades.	MODELO PRUEBAS DESPLIEGUE GESTION DEL PROYECTO	Interno 
ROL	**DESCRIPCION**	**DISCIPLINA**	**TIPO / SIMBOLO**
Documentador	Responsable de producir documentos para los involucrados como material de entrenamiento, documentación de las operaciones,	DESPLIEGUE	Interno

	documentación de soporte y documentación para el usuario final.		
Involucrado	Puede ser un usuario directo, usuario indirecto, jefe de los usuarios, miembro del equipo de trabajo, persona de atención al público, desarrolladores que trabajan en otro proyecto y que necesitan relacionarse con el sistema en elaboración o profesionales afectados por el desarrollo del sistema.	MODELO IMPLEMENTACION PRUEBAS DESPLIEGUE GESTION DEL PROYECTO	Externo 
Director del Departamento	Encargado de gestionar todos los proyectos	REVISION GESTION DEL PROYECTO MODELO	Externo 

Tabla 22. Roles en la metodología RASDUC

5.3 Marco de desarrollo de la metodología RASDUC

5.3.1 MODELO

El objetivo de esta disciplina es entender la organización, el dominio del problema que aborda el proyecto, y definir una solución viable para hacer frente al mismo, comprende los Requerimientos, Análisis y Diseño. En las fases de inicio y elaboración se hace un modelo inicial que no contiene muchos detalles, durante la etapa de construcción se realiza un detalle del modelo, en el que se especifica la solución obtenida luego de reuniones realizadas con las personas involucradas. En la disciplina de modelado se efectuarán las siguientes actividades:

5.3.1.1. Entender el proyecto

Es en las reuniones iniciales con los involucrados que el personal del Departamento identifica:

- **Objetivos de la organización para la que se realiza el proyecto**.- El primer paso es identificar con claridad los problemas o retos empresariales prioritarios, mientras más preciso, más fácilmente se podrá delimitar la dirección y el alcance de un proyecto que incorpore SOA. Disponer de una visión y un rumbo claros desde el principio hará mucho más fácil la ejecución de procesos cuya esencia es la integración de múltiples funciones.

- Definir el **alcance** y **restricciones** que tendrá el proyecto, delimitándolo de la forma más clara posible y excluyendo que áreas quedan fuera.
- **Definir las Metas y la visión del proyecto.-** El objetivo real debe responder a necesidades concretas de negocio y crear soluciones en pasos discretos, incrementales e iterativos, los servicios creados sin un significado de negocio claro, sin la granularidad adecuada o con demasiadas interconexiones, resultarán en una implementación compleja, inmanejable y costosa.

Roles involucrados:

- Director del Departamento
- Coordinador del Proyecto
- Involucrado
- Modelador

Artefactos Resultantes:

Documento llamado “Visión del proyecto” que contiene la descripción de los objetivos, la visión del proyecto y las metas a alcanzar con el mismo. *[Anexo VII. Visión del Proyecto]*

5.3.1.2. Explorar el Dominio del Problema

El modelador del proyecto en reuniones de trabajo con los involucrados y bajo la supervisión del coordinador del proyecto realiza las siguientes actividades con el objetivo de realizar un acercamiento y exploración del dominio del problema que el proyecto abarcará

- **Modelar la organización** para la que se realiza el proyecto
- Modelar **procesos de la organización**
- Capturar un **vocabulario** común
- Modelar el **dominio del proyecto**, identificando las principales entidades y sus relaciones
- Definir **pruebas de aceptación** del sistema
- **Identificación de servicios** y planeación de cómo se llevarán a cabo la administración y control de los mismos. Hay que determinar los procesos que se quieren exponer, es

recomendable que los mismos involucrados expongan las tareas de negocio prioritarias que podrán ser implementadas como servicios, luego el personal de TI podrá determinar la granularidad fina o gruesa del servicio.

- Modelar **requerimientos técnicos**

Roles involucrados:

- Modelador
- Involucrado
- Coordinador del Proyecto

Artefactos Resultantes:

Se generan varios documentos, que formarán parte del llamado "Modelo de Requerimientos", en donde estarán contenidos: Esquema de la Organización, Casos de Uso, Procesos de Negocio, Glosario, Dominio del Proyecto, Pruebas de aceptación, Requerimientos Técnicos. [Ver *Anexo VIII. Modelo de Requerimientos]*

5.3.1.3. Modelar la solución y la Arquitectura

En varias sesiones de trabajo el modelador junto al desarrollador capta los requerimientos detallados de los involucrados y bajo la revisión del coordinador del proyecto se logra:

- Analizar y **detallar requerimientos**
- Definir la **Arquitectura de la aplicación**, trabajado de forma colaborativa entre el analista del negocio y el arquitecto del sistema, lo que asegura una adecuada representación de la lógica del negocio
- Hacer un **modelo de datos físico**
- Definir **esquemas de calidad** de la aplicación
- Especificar aspectos de **seguridad**
- Modelar **Hardware y Middleware**
- Identificar los **riesgos** y aspectos de seguridad

Roles involucrados:

- Modelador
- Involucrado
- Desarrollador
- Coordinador del Proyecto

Artefactos Resultantes:

Se generan varios documentos, que formarán parte de los modelos resultantes de esta tarea.

En el llamado “Modelo de Diseño” constan: Arquitectura de la Aplicación, Esquema de clases, Diagrama Entidad Relación, detalle de requerimientos, casos de prueba de aceptación, definiciones de calidad, aspectos de riesgos y seguridad. El detalle de los requerimientos que puede representarse mediante diagramas de flujo, de actividad, de secuencia o mediante casos de prueba de aceptación y definiciones de calidad. [Ver *Anexo IX. Modelo de Diseño]*

5.3.2. IMPLEMENTACION

El objetivo de esta disciplina es la de transformar el modelo en código ejecutable y realizar un nivel básico de pruebas, en una unidad de pruebas en particular. En la disciplina de implementación se efectúan las siguientes actividades:

5.3.2.1. Preparación para el desarrollo

El personal desarrollador del sistema junto al modelador y a las personas involucradas necesarias proceden a:

- **Modelar prototipos** de las interfaces del sistema
- **Modularizar** el programa de tal forma que se pueda equilibrar el desarrollo por parte de todos los desarrolladores del proyecto

- **Definir unidades de prueba.-** Las pruebas de unidad son las que se aplican a una parte de un programa para comprobar que cumpla su función específica, ayuda al desarrollo de software de calidad.
- **Casos de prueba.-** Para los casos de uso principales o para las unidades de prueba, se procede a realizar casos de prueba, para darnos cuenta de errores de diseño, se trata de que para partes críticas se escriban pruebas en pseudocódigo o en el lenguaje de programación que ayuden a analizar y diseñar más detalladamente la unidad, así que en el momento de codificarla tenemos una idea mucho más clara y precisa de lo que debe hacer y cómo lo debe hacer.
- **Definir contratos de servicios.-** La creación del contrato de servicio Web antes de su desarrollo permite que los servicios web sean coherentes y estén alineados.

Roles involucrados:

- Desarrollador
- Modelador
- Involucrado

Artefactos Resultantes:

Documentos que contienen el Esquema de interfaces, la definición las unidades de prueba y los pseudocódigos o código fuente que contiene los casos de prueba [Ver *Anexo X. Modelo de Implementación y Pruebas y Anexo XVI. Catalogo de Servicios]*

5.3.2.2. Construcción del sistema

El personal desarrollador del sistema realiza:

- Digitar líneas de código, para la **construcción del sistema**
- Los desarrolladores se encargan de la **prueba y reparación** de su código particular
- **Chequeo de pantallas** y funcionalidad básica por parte del programador
- El personal que hace la administración de la base de datos procede con la **creación del esquema de la base de datos**.

- Administración de los **perfiles de acceso**, usuarios, permisos y accesos correctos hacia la base de datos.
- **Integración** con el proyecto que abarca todos los módulos especificados
- **Creación de servicios.-** Utilizando una tecnología comprobada que por el personal técnico que conoce de la implementación de estos

Roles involucrados:

- Desarrollador
- Administrador de la Base de Datos

Artefactos Resultantes:

El sistema con el código fuente y ejecutable de la aplicación y hojas de control y revisión básica para el programador [Ver *Anexo X. Modelo de Implementación y Pruebas]*. Esquema de la base de datos, scripts de instalación de la base.

5.3.3. PRUEBAS

La meta es realizar una evaluación objetiva para garantizar la calidad, lo que incluye encontrar defectos, validar que el sistema funciona según lo previsto, y verificar que se cumplan los requisitos iniciales.

5.3.3.1. Administración de las pruebas

El coordinador del proyecto junto al revisor determina lo necesario para realizar las pruebas del sistema:

- Definir y priorizar los requerimientos que deben ser probados
- Hacer un cronograma para efectuar el plan de pruebas

Roles involucrados:

- Coordinador del Proyecto
- Revisor

Artefactos Resultantes:

Modelo de pruebas [Ver *Anexo X. Modelo de Implementación y Pruebas]*

5.3.3.2. Verificación

El personal a cargo de las pruebas del sistema y en base al modelo de requerimientos definido inicialmente, procede a:

- Revisión de los documentos de modelado
- Ejecutar pruebas del sistema
- Revisión de la reacción del sistema ante los casos de prueba
- Revisión de la calidad del sistema
- Registrar y analizar los resultados de las pruebas

Roles involucrados:

- Revisor

Artefactos Resultantes:

Modelo de pruebas que contiene el reporte de fallas, la calidad del sistema y el conjunto de casos de prueba. [Ver *Anexo X. Modelo de Implementación y Pruebas]*

5.3.3.3. Validación

Usuarios seleccionados desde el personal final que utilizará el sistema, bajo la guía del personal Revisor, procede a:

- Validar el sistema una vez que ha pasado por la fase de verificación, ingresando a las pantallas y operando cada una de ellas.

Roles involucrados:

- Revisor
- Involucrado

Artefactos Resultantes:

Actualización del Modelo de pruebas con el reporte de sugerencias y fallas determinadas por el usuario final. [Ver *Anexo X. Modelo de Implementación y Pruebas]*

5.3.4. LIBERACION

El objetivo de esta disciplina es ejecutar que el sistema esté a disposición de los usuarios finales.

5.3.4.1. Planificación de puesta en producción

El director del departamento, el coordinador del proyecto y el desarrollador del sistema se ponen de acuerdo para poner el sistema en producción, con las actividades:

- Planear y priorizar las actividades necesarias para la puesta en producción del sistema
- Registro de las notas, características y observaciones necesarias en la versión del sistema que se libera

Roles involucrados:

- Director del Departamento
- Coordinador del Proyecto
- Desarrollador

Artefactos Resultantes:

Se obtiene el plan de implantación del sistema, [*Ver Anexo XI Plan de Liberación]*, que contiene el plan de liberación y las notas de la versión.

5.3.4.2. Distribución del sistema

El director del proyecto supervisa las actividades que el desarrollador del sistema realiza con el objetivo de hacer la distribución del sistema para los ambientes de producción

- Creación del ambiente de pre-producción necesario para las pruebas y capacitación, el mismo que contiene datos de prueba
- Puesta en producción del sistema
- Empaquetamiento del sistema y scripts de instalación

Roles involucrados:

- Coordinador del Proyecto
- Desarrollador

Artefactos Resultantes:

De esta disciplina se obtiene el paquete de instalación así como el sistema en los ambientes de preproducción y de producción.

5.3.4.3. Documentos del Proyecto

El documentador del proyecto y los involucrados del sistema proceden a:

- Revisión de la documentación de soporte
- Crear material de capacitación y entrenamiento para los involucrados
- Creación de manuales para el usuario

Roles involucrados:

- Involucrado
- Documentador

Artefactos Resultantes:

Los documentos de explicación para la utilización del sistema como manuales y material de capacitación [*Ver Anexo XI Plan de Liberación]*

5.3.5. GESTION DE LA CONFIGURACION Y CALIDAD

El objetivo de esta disciplina es la gestión de acceso a los artefactos del proyecto al mantener la integridad de los productos que se obtienen a lo largo del desarrollo del proyecto, garantizando que no se realicen cambios incontrolados y que todos los participantes en el desarrollo del sistema disponen de la versión adecuada de los productos No solamente se incluye el seguimiento de las versiones, sino también el control y la gestión de los cambios para ellos.

5.3.5.1. Establecimiento de la configuración

El coordinador del proyecto y el administrador de la configuración acuerdan:

- **Crear el ambiente de configuración** necesario para el proyecto. Se define el entorno tecnológico de soporte a la gestión de la configuración del sistema de información y se determinan los componentes hardware y software que van a permitir la mecanización de los procesos y controles que establece el plan. El entorno tecnológico en el que se apoyará el sistema de administración de la configuración puede ser diferente de aquel en el que se desarrollará el sistema de información
- **Crear una línea base,** El coordinador del proyecto y administrador de la configuración crean la línea base del proyecto, conformado por un conjunto de requerimientos, diseño, código fuente, código ejecutable y documentación al cual ha sido asignado un identificador único. Una línea base que es desarrollada para el cliente es llamada “release”, mientras que una línea base para un uso interno es llamada “build”.
- **Enlistar los ítems de configuración** a considerar, los mismos que pueden ser: documentación de procesos, requisitos de software, diseños de productos de software, código fuente y ejecutables, procedimientos de prueba, productos adquiridos, herramientas y otros ítems utilizados para crear y describir los productos de trabajo.

Roles involucrados:

- Coordinador del Proyecto
- Administrador de la Configuración

Artefactos Resultantes:

Se obtiene el Modelo de Configuración y Calidad ver [*Ver Anexo XII Modelo de* Configuración y Calidad*]*, con el listado de los ítems de configuración a considerar

5.3.5.2. Administración de la Configuración

El director del departamento, el coordinador del proyecto y el desarrollador del sistema se ponen de acuerdo para poner el sistema en producción, con las actividades:

- **Controlar Item de Configuración**, este control es mantenido sobre la configuración de la línea base del producto del trabajo, aprobando una nueva configuración si es necesario y actualizando la línea base.
- **Registrar Requerimientos de Cambio**, Las solicitudes de cambio y los reportes de problemas para todos los ítems de configuración, deben ser inicializados, registrados revisados, aprobados y monitorizados de acuerdo a un procedimiento documentado.
- **Monitorear y controlar** el proceso al establecer registros de Administración de la Configuración, se trata de monitorear y controlar el proceso de administración de la configuración en base al plan desarrollado para el proceso y tomar la acción correctiva apropiada.
- Ejecutar **Auditorías de Configuración**, Las actividades y procesos de auditoría registran posibles defectos encontrados en los ítems y confirman que las líneas base resultantes son precisas y registran los resultados apropiadamente

Roles involucrados:

- Administrador de la Configuración
- Modelador
- Desarrollador
- Documentador
- Administrador de la Base de Datos
- Revisor

Artefactos Resultantes:

El Sistema de Administración de la Configuración con los productos de trabajo controlados y los Procedimientos de control de acceso al Sistema de Administración de la Configuración

5.3.5.3. Ejecutar el plan de calidad

El coordinador del proyecto acuerdan:

- **Definir el plan**. Hay que definir las actividades del ciclo de vida de software cubierto por el plan de calidad. Considerando que si el software no sigue ninguna metodología siempre habrá falta de calidad.
- **Realización de actividades**.- En este punto habrá que ejecutar las actividades definidas en el plan, las mismas que podrán ser:
  - Revisión de métodos y herramientas de análisis, diseño, programación y prueba
  - Inspecciones técnicas en todos los pasos del proceso de desarrollo del software
  - Control de la documentación del software y de los cambios realizados
  - Registro de auditorías y realización de informes
- **Revisión de Requisitos de Software.** Los requisitos del software son la base de las medidas de calidad. La falta de concordancia con los requisitos es una falta de calidad.
- **Revisión de requisitos implícitos**. Los requisitos implícitos o expectativas que a veces no se mencionan pero se esperan.

Roles involucrados:

- Coordinador del Proyecto
- Administrador de la Configuración

Artefactos Resultantes:

Se obtiene el Modelo de Configuración y Calidad ver [*Ver Anexo XII Modelo de* Configuración y Calidad*]*, con el listado de los ítems de configuración a considerar

5.3.6. Gestión del Proyecto

El objetivo de esta disciplina es dirigir las actividades que se lleva a cabo en el proyecto, al establecer un plan que contenga las actividades del proyecto y realizar seguimiento continuo en base a este, incluye los riesgos de gestión y

dirección del equipo de trabajo para asegurarse que el sistema es entregado a tiempo y dentro del presupuesto.

5.3.6.1. Establecimiento de estimaciones

El director del departamento junto al coordinador del proyecto se ponen de acuerdo para:

- Estimar el alcance del proyecto, en base a las habilidades y recursos del departamento, se debe estimar el alcance que debe contemplar el proyecto, además de definir los objetivos y entregables que se considerará.
- Establecer estimaciones de trabajo y atributos de las tareas. A partir de los requerimientos del producto, especificar una definición más clara del alcance del proyecto. Es necesario realizar una representación con las metas a alcanzar enlazadas a las grandes áreas que formarán parte del proyecto, luego dividir las áreas en paquetes de trabajo, mientras más se profundice en el nivel de detalle de las tareas se podrá alcanzar un cronograma más realista y se facilitará la definición de responsabilidades.
- Determinar las estimaciones de tiempo, personal, esfuerzo y costo. Usar un método apropiado para determinar los atributos del producto de software y las tareas para estimar los recursos requeridos, Estimar materiales, equipos, trabajo y métodos requeridos para el trabajo.

Roles involucrados:

- Coordinador del Proyecto
- Director del Departamento

Artefactos Resultantes:

Se obtiene el Modelo de Planeación ver [*Anexo XIII. Plan del Proyecto*] con el alcance del proyecto que incluye los objetivos y entregables del proyecto; las metas, áreas y tareas que darán lugar al cronograma para el proyecto; finalmente las estimaciones de tiempo, personal, esfuerzo y costo.

5.3.6.2. Plan del proyecto

Se basa en los requerimientos del proyecto y en las estimaciones establecidas, considera las fases del ciclo de vida del proyecto de software y es efectuado por el coordinador del proyecto, trabajando en conjunto con todos los involucrados.

- Crear un documento que contenga el plan del proyecto
- Definir el ciclo de vida del proyecto, a través de un análisis exhaustivo sobre la naturaleza del proyecto, debe definirse el ciclo de vida del proyecto de software.
- Identificación y valoración de riesgos del proyecto, documentar los riesgos y realizar un análisis de impacto y ocurrencia de ellos, priorizándolos en base al impacto que puede generar sobre el proyecto, para establecer planes de contingencia que deben ser documentados, aprobados y socializados con el personal involucrado, y monitorear los riesgos continuamente, para determinar si se debe agregar o retirar alguno, o si el riesgo se hizo problema y documentar estos cambios.
- Seguimiento continuo al plan, se deben revisar todas las actividades que conforman el proyecto, para garantizar una comprensión y evaluación adecuada de cómo se va cumpliendo los alcances, objetivos, funciones y relaciones necesarias para el éxito del proyecto.
- Interactuar adecuadamente con los involucrados, se debe disponer de un listado de los involucrados más relevantes en cada fase del ciclo de vida del proyecto, con una definición de sus roles y responsabilidades. El coordinador del proyecto debe realizar un acta de compromiso para que cada involucrado asuma la responsabilidad de participar en el proyecto.

Roles involucrados:

- Coordinador del Proyecto

Artefactos Resultantes:

Se obtiene el documento Plan del proyecto [*Anexo XIII. Plan del Proyecto*] el mismo que contiene El propósito, alcance y objetivos del proyecto, identificación de los procedimientos, métodos y estándares para desarrollo y

mantenimiento del software, Identificación de los productos de software a ser desarrollados, estimaciones de tamaño de los productos de software en elaboración, Estimación de riesgos, Estimación de los esfuerzos y costos del proyecto, Estimación de uso de herramientas y recursos críticos de hardware. Cronograma del proyecto de software, incluyendo la identificación de hitos y revisiones.

5.3.7. Entorno

El objetivo de esta disciplina es apoyar el resto de esfuerzos, garantizar el proceso adecuado, normas de orientación (estándares y guías), y herramientas (hardware, software, etc.) que están disponibles para el equipo según sea necesario.

5.3.7.1. Metodología de Soporte

El modelador del proyecto proporciona las plantillas y guías necesarias que son el soporte de la metodología hacia el equipo de trabajo revisadas y aprobadas por el coordinador del proyecto.

- Adecuar hacia una metodología personalizada a los objetivos del proyecto de software y preparar la lista de los entregables necesarios para el proyecto.
- Revisar y aprobar la metodología propuesta para el proyecto
- Poner a disposición del equipo las guías y plantillas que dan soporte a la metodología

Roles involucrados:

- Documentador
- Coordinador del Proyecto

Artefactos Resultantes:

Se tienen las guías y plantillas que el grupo del proyecto utilizará como base

5.3.7.2. Herramientas de Soporte

El administrador de la configuración es responsable de proveer de las herramientas y utilitarios de software, necesarios para todo el equipo del proyecto, las tareas que debe realizar son:

- Seleccionar y adquirir las herramientas y licencias de software adecuadas
- Instalar y configurar las herramientas de trabajo necesarias para el proyecto
- Capacitar a los miembros del proyecto en la utilización de las herramientas

Roles involucrados:

- Administrador de la Configuración

Artefactos Resultantes:

Se obtienen e instalan las herramientas de trabajo, se realiza la instalación, configuración y capacitación en las mismas.

5.4 Metodología Detallada por Fases

5.4.1. INICIO

Esta fase trata de realizar la identificación del alcance y establecer la dimensión del proyecto, hacer la propuesta de la arquitectura y obtener el presupuesto para la aplicación. Se detalla que actividades y entregables se obtienen en la fase de inicio. Los involucrados participan activamente para conseguir el objetivo de tener una buena comprensión del proyecto y definición del alcance de la versión, en varias sesiones de trabajo se identifica los nombres de las principales entidades, requerimientos y requerimientos técnicos.

5.4.1.1. Modelo

En la fase de inicio para la disciplina del modelo se identifica:

- Objetivos de la organización para la que se realiza el proyecto
- Metas y visión del proyecto

- Modelo que representa la estructura de la organización
- Se empieza con el desarrollo del glosario que describe términos importantes para el problema
- Modelo del dominio conformado por las principales entidades y sus relaciones
- Requerimientos técnicos generales

5.4.1.2. Implementación

En la fase de inicio, para la disciplina de implementación, se procede a:

- Modelar los prototipos de las interfaces del sistema, para que los usuarios tengan un acercamiento inicial al futuro sistema y puedan determinar si sus requerimientos y necesidades se cubrirán en las pantallas que posteriormente operarán.

5.4.1.3. Pruebas

En la fase de inicio esta disciplina no realiza actividades

5.4.1.4. Despliegue

En la fase de inicio esta disciplina no realiza actividades

5.4.2. ELABORACIÓN

Confirmación de la idoneidad de la arquitectura

5.4.2.1 Modelo

En la fase de elaboración para la disciplina del modelo se trata de:

- Crear el esquema de casos de uso de la aplicación en base del modelo del dominio
- Modelar los procesos de la organización
- Se definen las pruebas de aceptación del sistema
- Se detallan los requerimientos técnicos necesarios
- Se define la arquitectura de la aplicación
- Se incluyen los esquemas necesarios para la representación de:
  - Hardware y middleware,
  - Diagramas de implementación y de red
  - Riesgos
  - Seguridad
- Se capta el vocabulario del proyecto y se lo agrega al glosario inicial

5.4.2.2 Implementación

En la fase de elaboración, se procederá a:

- Revisar con detalla la arquitectura de la aplicación
- Crear módulos de programación
- Definir unidades de prueba
- Realizar casos de prueba

5.4.2.3 Pruebas

En la fase de elaboración para la disciplina de pruebas, se procederá a:

- Definir y priorizar los requerimientos que deben ser probados
- Revisión de los documentos de modelado

5.4.2.4 Despliegue

En la fase de elaboración esta disciplina no realiza actividades

5.4.3. CONSTRUCCIÓN

Desarrollo incremental de sistema, siguiendo las prioridades funcionales de los implicados, esta fase propone realizar varias iteraciones, entonces se debe considerar que todos los artefactos considerados en las disciplinas, deberían ser actualizados como resultado de cada iteración.

5.4.3.1 Modelo

En la fase de construcción, para la disciplina del modelo se trata de:

- Detallar y especificar los casos de uso de la aplicación
- Se crea el esquema de base de datos necesaria para dar soporte a la aplicación
- Se realiza el esquema de las interfaces para la aplicación
- Se detalla los requerimientos de la aplicación y se procede a especificar procesos utilizando el lenguaje UML con los diagramas que se consideren necesarios según la complejidad de la aplicación:
    - Diagramas de Flujo
    - Diagramas de actividad
    - Diagramas de secuencia
    - Diagramas de estado
    - Casos de prueba de aceptación

5.4.3.2 Implementación

En la fase de construcción para la disciplina de implementación, se realiza:

- Construcción del sistema
- Crear y optimizar el esquema de la base de datos.
- Manejar los usuarios y accesos correctos hacia la base de datos.

5.4.3.3 Pruebas

En la fase de implementación, para la disciplina de pruebas se procederá a:

- Hacer un cronograma para efectuar el plan de pruebas
- Ejecutar pruebas del sistema

- Registrar y analizar los resultados de las pruebas
- Revisión de la reacción del sistema ante los casos de prueba

5.4.3.4 Despliegue

En la fase de construcción, la disciplina de despliegue realiza las actividades:

- Planear y priorizar las actividades necesarias para la puesta en producción del sistema
- Creación del ambiente de pre-producción en donde se puede validar que el sistema trabaja bien antes de ponerlo en producción final
- Empaquetamiento del sistema y scripts de instalación
- Registro de las notas, características y observaciones necesarias en la versión del sistema que se libera
- Crear material de capacitación y entrenamiento para los involucrados

5.4.4. TRANSICIÓN

Validación e implantación del sistema.

5.4.3.5 Modelo

En la fase de transición, para la disciplina del modelo se trata de:

- Especificar y detallar las definiciones de calidad necesarias para que la funcionalidad del sistema sea considerada como aceptada por parte de los programadores

5.4.3.6 Implementación

En la fase de transición la disciplina de implementación realizará:

- Prueba y reparación por parte de los desarrolladores, en su código particular
- Integración con el proyecto que abarca todos los módulos especificados

5.4.3.7 Pruebas

En la fase de transición, para la disciplina de pruebas se procederá a:

- Revisión de la calidad del sistema

5.4.3.8 Despliegue

En la fase de transición la disciplina de despliegue realiza las actividades:

- Finalizar el paquete de instalación
- Finalizar la documentación, de soporte del sistema y la documentación para el usuario final

- Anunciar con anticipación y entrenar a los usuarios
- Poner el sistema en producción

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

Al finalizar el presente proyecto de tesis, evaluando el objetivo planteado al inicio del documento, que consiste en plantear un método de integración y desarrollo de software hacia una Arquitectura Orientada a Servicios para el Departamento Informático de la Universidad de Cuenca; haciendo un recorrido por cada uno de los capítulos se puede concluir:

- Al buscar documentos que indiquen la arquitectura utilizada en las aplicaciones, se pudo ver que se tenían diferentes enfoques de lo que es la arquitectura, por eso en esta tesis se empezó aclarando ese concepto, para proponer una metodología que pueda definir bajo un criterio estándar y claramente especificado como se plantea la arquitectura para los sistemas que a futuro se construyan para el DDI
- Se ha realizado una contextualización de la situación del Departamento de Desarrollo Informático y se hace un análisis de los actores universitarios, introducción que es utilizada posteriormente en las codificaciones de participantes utilizadas en las tablas de códigos participantes dentro del Anexo XV. Modelo de Servicios
- Antes de implantar un proyecto dentro de una organización, es necesario determinar el nivel de madurez de la organización; para este trabajo de tesis, se trabajó con un pequeño cuestionario aplicado al área de desarrollo de software, que ayudó a conversar con las personas involucradas y se pudo evaluar la situación del departamento en cuanto a proyectos de software; no se ha profundizado en este aspecto pues el objetivo de la tesis no es el determinar ni explicar el nivel CMMI en que se encuentra el DDI.
- Al conocer con detalle la situación de los proyectos de software, se pudo determinar la necesidad de una metodología que sería parte del plan que permitiría trabajar con servicios y a largo plazo llegar a tener una arquitectura SOA.

- Se ha realizado un inventario de sistemas existentes en el DDI en base a información proporcionada por el director del área de desarrollo de software y el personal del departamento, información que no se encontraba documentada en el DDI; posteriormente se ha clasificado los sistemas por tipo de tecnología.
- Se ha planteado un método de integración para los sistemas ya existentes, en base a las entidades comunes a varios sistemas, pensando en función de los objetivos de negocio.
- Con un enfoque orientado a objetos se ha propuesto una manera en que las acciones de los objetos de negocio podrían encapsularse dentro de servicios web, gracias a los cuales se haría la integración de los sistemas ya existentes.
- Para los nuevos sistemas a desarrollarse, se ha planteado una metodología aplicable para el Departamento de Desarrollo, que mejor se adapta a sus necesidades, puesto que de la realidad del Departamento de Desarrollo Informático de la Universidad de Cuenca hace que el proceso de incorporación de SOA y servicios web no sea aplicable de la manera en la que se plantea en varios artículos de otras realidades.
- La metodología propuesta simplifica conceptos que RUP los extiende con artefactos y actividades innecesarias para una organización como el DDI.
- Dentro de la metodología planteada, se presentan roles acorde a la organización del DDI, en donde no se dispone de especialistas en una sola área, más bien las personas realizan en la práctica una diversidad de roles.
- La metodología simplifica la documentación al organizar en lo que se denomina “modelos”, los artefactos que resultan de varias actividades e incluso fases como el “Modelo de requerimientos” o el “Modelo de diseño”
- En el marco teórico se ha procedido a revisar los temas que abarca la presente tesis, partiendo de profundizar el concepto de arquitecturas de sistemas, y los estilos arquitectónicos que existen, para posteriormente

en el capítulo 3 clasificar y analizar los proyectos de software que están a cargo del DDI.

- Luego de profundizar en el marco teórico del tema SOA se ha podido determinar que el software orientado a servicios, es conveniente, porque mejora su sostenibilidad y su integración con otros componentes, y aunque la conveniencia de su utilización no depende del tamaño de la organización, el Departamento Informático aún no puede realizar tareas de coreografía y orquestación de servicios, sin haber madurado antes en la creación y utilización de servicios web.
- Una vez que se ha efectuado un estudio sobre las metodologías ágiles, tradicionales de construcción de software y metodologías utilizadas en SOA, se ha concluido que no son aplicables completamente en el DDI. El Departamento Informático debe primero documentar sus sistemas con una metodología que refuerce el concepto de programación orientada a objetos y programación en capas.
- Se ha hecho énfasis en disponer de una metodología que organice el proceso de creación de software, debido a que SOA no es solo tecnología, es una estrategia, una disciplina que ayuda a eliminar la diferencia entre los procesos de negocios y las tecnologías que permiten que los negocios funcionen.
- La realidad universitaria descarta la posibilidad de que absolutamente todos los sistemas desarrollados en el Departamento de Desarrollo Informático puedan integrarse en una plataforma de un único sistema operativo y única base de datos. Posteriores proyectos SOA podrían profundizar en las tecnologías necesarias para implementar servicios web, ESB, coreografía de servicios y otros, en base al planteamiento de unificación de sistemas que se hace en esta tesis y al haber madurado en programación orientada en objetos, creación de componentes y servicios web.
- Aunque SOA no pueda sea aplicado en su amplio concepto dentro de una organización como el DDI, puesto que las aplicaciones siguen siendo concebidas desde un principio como islas independientes, los servicios web son el medio que permitirá el enlace de estos

constituyendo el primer paso hacia la incorporación de SOA, para que en una segunda etapa se haga una sincronización de estos hacia los objetivos del negocio.

- En la sección anexos, quedan planteadas muchas plantillas que no existían en el departamento, es una propuesta que posteriormente puede mejorarse, y será útil para que la información quede documentada y en próximos proyectos se pueda agilitar el trabajo de desarrollo de software.

HIPOTESIS INICIALES

En un mundo globalizado, en donde la tendencia organizacional es fusionarse y crecer, SOA empieza a tener más vigencia y a representar un enfoque que supera a nivel corporativo otras arquitecturas, en términos de retorno de inversión, puesto que se apalanca en desarrollos existentes, independientemente del fabricante o la tecnología empleada en su construcción, sin embargo al iniciar el presente proyecto de tesis, existía la hipótesis de que SOA no era aplicable para el DDI, pues la orientación era unificar todos los sistemas universitarios bajo una sola base de datos, sin embargo al estar en una organización universitaria, que involucra varios sistemas construidos en varias facultades o departamentos existe la posibilidad de que se empleen diversas tecnologías que necesiten compartir puntos comunes de información, por lo que se debería dejar una puerta abierta para interrelacionar sistemas comunicados por servicios web, empleando plantillas estándar proporcionadas por el DDI.

Al existir sistemas desarrollados sin un estándar metodológico, y no existir una definición única de arquitectura ni documentación de los sistemas, se puede confirmar la hipótesis de que no existe una metodología empleada en el Departamento de Desarrollo Informático, lo cual no favorece la arquitectura SOA.

RECOMENDACIONES

- En la Universidad de Cuenca no se puede aplicar el concepto de SOA tal y como está el nivel de madurez del área de desarrollo de software en este momento, por lo que se recomienda que se incorpore una metodología como RASDUC, que simplifica la exagerada e innecesaria documentación propuesta por RUP.
- Se recomienda que el Departamento de Desarrollo Informático tome un proceso incremental en la adopción de SOA, de forma que se vayan obteniendo resultados útiles en base a pocos proyectos, de otro modo puede ocurrir que al querer enlazar de una sola vez todos los sistemas el impacto sea profundo y la adopción no pasará de estado de propuesta o puede ocurrir que la adopción no llegará a completarse.
- Se recomienda que en base a plantillas estándar los nuevos sistemas hagan uso de los servicios web para propiciar una integración de sistemas mediante servicios web.
- Si bien los casos de prueba no han sido utilizados dentro del DDI, pueden agilitar en gran manera la codificación de los puntos críticos del sistema, por lo que dentro de la metodología se recomienda incorporar esta buena práctica al desarrollo de todo proyecto de software.
- Se recomienda cambiar la organización funcional del departamento informático para ajustarse a lo que la metodología RASDUC sugiere, tratando de acoplar los roles que tienen en el departamento a lo que se requiere.
- Lo que ocurría generalmente en el Departamento de Desarrollo Informático es que al hablar de arquitectura los desarrolladores pueden estar refiriéndose a algo distinto de lo que esperamos, por lo que se recomienda estandarizar una plantilla que puede ser la recomendada dentro de la metodología RASDUC, en el Anexo Modelo de Diseño.
- La preparación y conocimiento que el personal del Departamento de Desarrollo Informático puedan tener sobre los temas servicios web, SOA, metodologías de desarrollo, son muy importantes y normalmente no se resuelve planificando meras acciones puntuales de formación,

sino que requiere de una planificación, seguimiento y asignación a algún agente activo para educar, crear y adaptar normas y asistir en su cumplimiento.

- Debido a que los proyectos SOA no son un proyecto únicamente del Departamento de Desarrollo Informático, se recomienda que la gente del negocio o los niveles directivos universitarios deber estar realmente involucrados, pues se está implementando algo más grande, que estratégicamente esté buscando cumplir los objetivos de negocio, que comprometen a la alta gerencia de la organización.
- En el DDI se hacen diagramas a nivel general, sin entrar en detalles para cada uno de estos, por lo tanto no se dispone de documentación de cómo se realizan ciertas actividades en detalle, solamente se tiene en la documentación del código fuente la explicación de los procesos, por lo tanto se hace la recomendación de una reorganización de las actividades que desempeña el personal para especializarse en actividades o roles específicos.
- Para hacer análisis de los posibles servicios, se puede seguir dos caminos posibles: TOP-DOWN y BOTTOM-UP. Cuando los requerimientos provienen de sistemas ya existentes y los procesos de negocio no están definidos, es recomendable seguir la estrategia Bottom-Up; en cambio la alternativa TOP-DOWN, es buena cuando se parte del análisis basado en entidades o procesos de negocio.
- El Departamento de Desarrollo Informático debería disponer de una metodología estándar para ser aplicada en todos los proyectos que se proponen a cargo de esta dependencia, con plantillas en los que los desarrolladores se basan. Se recomienda por lo tanto que al finalizar los sistemas se lleve un control de documentos y artefactos entregados, en una especie de actas entrega-recepción.
- Debido a que la Universidad de Cuenca es un espacio que genera conocimientos, recomendamos la fusión de varios conceptos para la propuesta de una metodología de desarrollo de software en lugar de tomar un esquema propuesto o vendido por una organización externa a

la Universidad de Cuenca con procesos y artefactos aplicables a otra realidad.

- El Departamento de Desarrolla Informático debería adoptar un sistema de gestión de la calidad, el cual se aplicará a todos los productos de software desarrollados a futuro, que contemple las diferentes necesidades, objetivos particulares, productos que proporciona, procesos que emplea y el tamaño y estructura de la organización

En conclusión, el fracaso o el éxito dependen en mucho de entender el alcance de este paradigma, saber cuándo emplearlo y cuándo evitarlo. Así como de elegir el camino SOA, entender el reto técnico, funcional y de manejo de proyecto que esta decisión implica, elegir estándares de integración y honrarlos en todo momento a través de un sólido contrato entre las partes, implementado por un personal altamente capacitado en las tecnologías elegidas.

Bibliografía y fuentes de información

Achíg Subía, L. (2007). *Perfil del Plan Estratégico de la Universidad de Cuenca 2006-2010.* Cuenca: Abril productos gráficos.

Alvez, P., Foti, P., & Scalone, M. (Junio de 2006). *Proyecto Batuta.* Recuperado el 30 de 04 de 2009, de www.willydev.net

Arceo, C. M. (11 de 07 de 2006). *SOA levanta vuelo.* Recuperado el 16 de 1 de 2009, de infoworld: http://www.iworld.com.mx/iw_SpecialReport_read.asp

Bass, Clements, & Kazman. (2003). *Software Architecture in Practice.* Addiso-Wesley.

Booch, Rumbaugh, & Jacobson. (1999). *The Unified Modeling Language User Guide.* Addison-Wesley.

Bustacara, C. (2008). *Universidad Javeriana.* Recuperado el 23 de 03 de 2009, de sitio Universidad Javeriana: sophia.javeriana.edu.co/~cbustaca/ArquitecturaSoftware/arquitecturas_software_ADLs

CIENTEC. (11 de 12 de 2008). *CIENTEC, Respuestas rápidas Souciones permanentes*. (CIENTEC) Recuperado el 11 de 12 de 2008, de www.cientec.com/index.html

CMMI, T. (August 2002). *Capability Maturity Model Integration (CMMISM) Version 1.1.* Pittsburgh: Carnegie Mellon University.

COMPUTING-ES. (29 de 09 de 2004). Recuperado el 18 de Noviembre de 2008, de Arquitecturas SOA: www.computing-es.com

Cortizo, J. (2009). *Service Oriented Architecture.* Recuperado el 20 de 4 de 2009, de http://www.esp.uem.es/jccortizo

Dure, Octavio (3 de Marzo de 2010). SOA*: Un Modelo de Dominios que Ataca Todos los Aspecto de una Organización.* Recuperado de http://www.willydev.net /InsiteCreation /v1.0/ descargas/soa/WillyDev_Nota_Revista_Code.pdf

Erl, T. (2005). *Service-Oriented Architecture: Concepts, Technology, and Design.* Crawfordsville, Indiana: Prentice Hall.

Espacio SOA. (02 de 04 de 2007). *La Importancia de la Gobernabilidad en SOA.* Recuperado el 25 de 05 de 2009, de http://www.espaciosoa.net/2007/04/02/la-importancia-de-la-gobernabilidad-en-soa/

Fowler, Martin (21 de 07 de 2000). The New Methodology . www.martinfowler.com

Folgado, S. I. (29 de 09 de 2004). *Un paradigma de funciones como servicios.* Recuperado el 2008 de noviembre de 19, de www.computing-es.com

Garlan, D., & Shaw, M. (1994). *An Introduction to Software Architecture.* New Jersey: CMU/SEI.

Gea, M., Gutierrez, F., Garrido, J., & Cañas, J. (4-8. 11-13 de Junio de 2003). *Teorías y Modelos Conceptuales para un Diseño basado en Grupos.* Recuperado el 10 de Diciembre de 2008, de IV Congreso Internacional de Interacción Persona-Ordenador: http://lsi.ugr.es/~mgea/invest/articulos/gea_interacc03.pdf

Lugo Gonzalez, Carlos Andrés. *Servicios Web y Orquestación.* http:carlosandreslugo.googlepages.com/ServiciosWebyOrquestacion.pdf Recuperado: 15 de septiembre de 2009

Jacobson, Ivar, Grady Booch, and James Rumbaugh. El Proceso Unificado de Desarrollo de Software. México: Addison-Wesley, 1999.

Krafzig, D., Banke, K., & Slama, D. (2004). *Enterprise SOA Service Oriented Architecture Best Practices .* Prentice Hall PTR.

Kruchten, P. (1995). *Architectural Blueprints-The "4+1" View Model of Software Architecture.* IEEE Software.

Lugo Gonzalez, A. (s.f.). *Servicios Web y Orquestacion.* Recuperado el 25 de 05 de 2009, de WillyDev: http://carlosandreslugo.googlepages.com/ServiciosWebyOrquestacion.pdf

Microsoft Corporation. (12 de 2006). *microsoft msdn.* Recuperado el 11 de 12 de 2008, de www.msdn.microsoft.com

Navarro, R. (2006). *Rest vs Servicios Web.* España: ELP-DSIC-UPV.

Parnas, D. (1972). *On the Criteria for Decomposing Systems into Modules.* Communications of the ACM.

Perry, D., & Wolf, A. (1992). *Foundations for the study of software architecture.* ACM SIGSOFT Software Engineering Notes.

Reynoso, C. B. (26 de 06 de 2006). *Introducción a la Arquitectura de Software.* Recuperado el 18 de 2008 de 08, de spanish/msdn/arquitectura/default.mspx

Rigoni Brualia, C. CMMI: mejora del proceso en fábricas del software. Madrid: AEC Asociacion Española para la calidad-Ministerio de Industria Turismo y Comercio.

Rodríguez, L., Vignaga, A., & Zipitría, F. (2006). Estudio de Interoperabilidad .NET/J2EE. Montevideo, Uruguay: Universidad de la República, Instituto de Computación.

Rodriguez, N. (2002). *Ciencia, Tecnología & Sociedad.* Quito: Editorial Universitaria Universidad Central del Ecuador.

Romero, M. A. (2005). ARQUITECTURA DE SOFTWARE, ESQUEMAS y SERVICIOS. La Habana, Cuba: Procyon, Softel.

Urrutia, J. (19 de 09 de 2006). *La vida en bytes*. Recuperado el 16 de 01 de 2009, de http://www.misbytes.com/wp/2006/09/19/arquitecturas-empresariales-bpm-y-soa/

Glosario de términos y abreviaturas

ABCD	Automatización de Bibliotecas y Centro de Documentación, software libre y de código abierto FOSS.
ADLs	Lenguajes de Descripción de Arquitectura
ADMINUC	Sistema de Administración de Procesos Comunes
API	Interfaz de programación de aplicaciones (Application Programming Interface)
AS	Arquitectura de Software
CDJBV	Centro Documental Juan Bautista Vásquez
COM	COM es el Modelo de Objetos Basados en Componentes de Microsoft (Component Object Model), una tecnología para construir aplicaciones a partir componentes binarias de software.
ESB	Bus de servicios empresariales (Enterprise Service Bus). Intermediario que combina publicación/suscripción de mensajería, mensajería síncrona y asíncrona, transformación básica, enrutamiento y soporte nativo de Servicios..
DDI	Departamento de Desarrollo Informático
Dependencia	Facultad, Escuela, Carrera, Centro o Departamento universitario
DMZ	Zona desmilitarizada de una red
Edición	Término empleado para la actualización, modificación o eliminación de una entidad
HTTP	HyperText Transfer Protocol (Protocolo de transferencia de hipertexto)
IDL	Lenguaje de Descripción de Interfaz
RASDUC	Metodología Rup Agil con enfoque SOA del Departamento de Desarrollo Informático de la Universidad de Cuenca
REST	Representational State Transfer, célebre modelo presentado en la tesis de Roy Fielding
RUP	Rational Unified Process
SACF	Sistema de Activos Fijos
SBIB	Sistema de gestión de Biblioteca
SCGI	Sistema de Control de Gasto Interno
SCOP	Sistema de Contabilidad y Presupuesto
SEI	Software Engineering Institute en la Universidad Carnegie Mellon de

	Pittsburgh, Pennsylvania
SGA	Sistema de Gestión Académica
SGF	Sistema de Gestión Financiera
SGSE	Sistema de Gestión Socio Económica
SGP	Sistema de Gestión de Personal
SOAP	Simple Object Access Protocol (Protocolo de acceso simple a objetos)
RPC	Remote Procedure Calls (Llamadas a Procedimientos Remotos)
UDDI	Universal Description, Discovery, and Integration (Descripción, descubrimiento e integración universal)
UML	Lenguaje de modelado unificado
XML	Extensible Markup Languaje (Lenguaje de marcado extensible)
WBS	Work Breakdown Structure, Estructura de División del Trabajo
W3C	World Wide Web Consortium
WSTK	IBM Web Service ToolKit

Anexos

ANEXO I. FINES DE LA UNIVERSIDAD DE CUENCA

En el documento de carácter estratégico como es el estatuto de la Universidad de Cuenca, publicado en Octubre de 2003, Capítulo I. Art. 2.- Se declara que son fines de la Universidad:

a) La búsqueda libre de la verdad
b) Formar, capacitar y especializar a estudiantes en los niveles de pregrado y postgrado, en las diversas áreas del conocimiento, proporcionándoles una educación humanista e integral.
c) Fomentar la creación, preservación, desarrollo y transmisión de la ciencia, la técnica, las artes y la cultura mediante la investigación
d) Aplicar los conocimientos al desarrollo social del país y de la religión, en orden a establecer una sociedad justa, equitativa y solidaria
e) Establecer la vinculación permanente con la colectividad, en orden a difundir la ciencia, la técnica, las artes y la cultura; así como fomentar y fortalecer la unidad entre los fines de la Universidad, las necesidades sociales y los proyectos de desarrollo del país.
f) Establecer la adecuada evaluación y acreditación de sus actividades como proceso permanente de rendición de cuentas a la sociedad y de revisión de los procesos y de la estructura universitaria, para adecuarla a las cambiantes demandas científicas y sociales
g) Promover el estudio y respeto de los valores consustanciales a la persona, en particular, la libertad, la igualdad, la solidaridad, el pluralismo, la tolerancia y el espíritu crítico
h) Velar por el patrimonio cultural nacional
i) Contribuir al fortalecimiento de la identidad en el contexto pluricultural del país, a la afirmación de la democracia y la libertad, a la paz entre los pueblos, a la plena vigencia de los derechos humanos, a la integración latinoamericana, a la protección del medio ambiente y a la promoción del desarrollo sustentable
j) Adoptar políticas y mecanismos específicos para promover y garantizar una participación equitativa de las mujeres en todos sus niveles e instancias.
k) Los demás establecidos en la Constitución y la Ley.

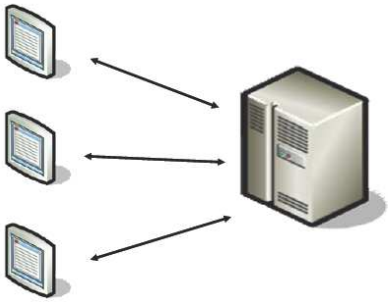
De los fines descritos se desprende la naturaleza de las actividades universitarias enfocadas en la docencia, investigación, extensión y otras, así como su carácter disciplinario o interdisciplinario (Estatuto de la Universidad de Cuenca, 2003).

ANEXO II. TECNOLOGIAS EN LA IMPLEMENTACION DE SISTEMAS

1.- Procesamiento Central

Todo el procesamiento se lo hace en una sola computadora. Se necesita instalar completamente la aplicación y la base de datos en un mismo sitio.

2.- Monolítica

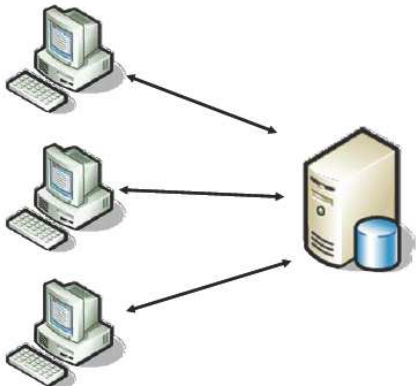


Se tiene un solo equipo de hardware, en el que se comparten múltiples aplicaciones. Las aplicaciones corren sobre el servidor.

El cliente tiene “terminales tontas” con una interfaz visual basada en caracteres o los usuarios emplean sencillos ordenadores personales, con modernas interfaces gráficas de usuario.

3.- Arquitecturas Cliente-Servidor

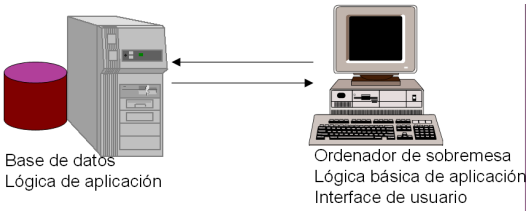
Distribuye el procesamiento entre máquina cliente y máquina servidor. Un sistema se divide en dos capas claramente diferenciadas: un proceso de dos capas llamado cliente que hace la solicitud de información y un proceso llamado servidor que proporciona la respuesta solicitada. Se dispone de una capa que contiene el interface de presentación más lógica de la aplicación y la otra capa contiene generalmente el acceso a la base de datos. En el lenguaje de programación se vincula generalmente la interfaz de usuario a un elemento de la base de datos que permanece constantemente abierta.



Variantes del esquema cliente/servidor

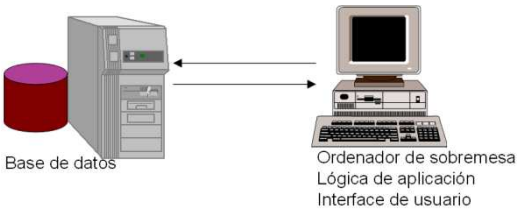
Lógica Distribuida

En el cliente se llevan a cabo la interacción con el usuario y la parte más trivial de la lógica de la aplicación, como controles básicos de rango de campos, campos obligatorios, etc, mientras que el grueso de la lógica permanece en el servidor.

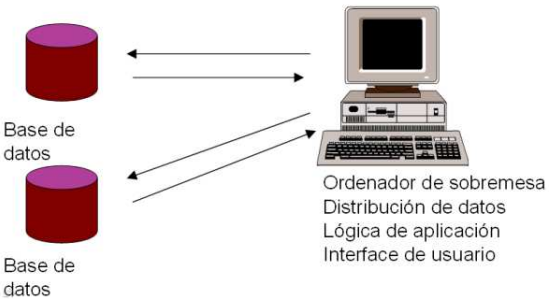


Gestión Remota de Datos

El cliente realiza tanto las funciones de presentación como los procesos, el servidor maneja los datos que permanecen en una base de datos centralizada. En esta situación se dice que hay una gestión de datos remota. Tanto la interacción con el usuario como la aplicación residen en el cliente, siendo el servidor el depositario de los datos.

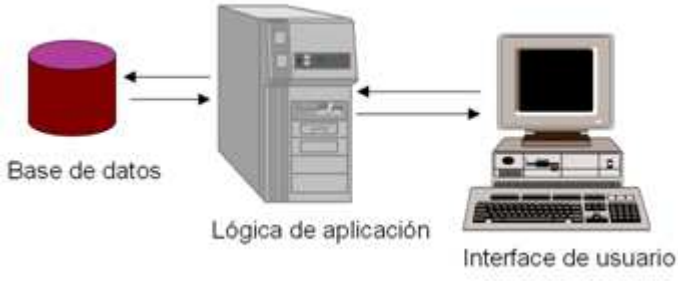


Bases de Datos Distribuidas



El cliente realiza la interacción con el usuario, ejecuta la aplicación (que debe conocer la topología de la red), así como la disposición y ubicación de los datos. Se delega parte de la gestión de la base de datos al cliente.

4.- Tres Capas



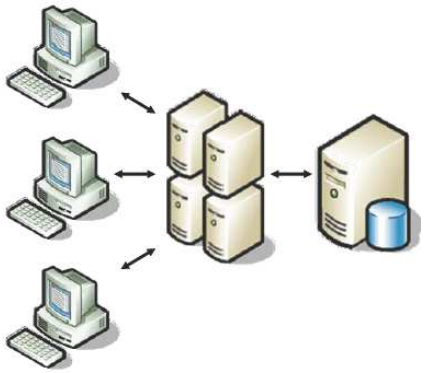
El cliente se encarga de la interacción con el usuario, el servidor de la lógica de aplicación y la base de datos puede estar en otro servidor.

CAPA DE INTERFAZ. La capa de la presentación es la interfaz gráfica que muestra los datos a los usuarios. No desarrolla ningún procesamiento de negocios o reglas de validación de negocios, se puede desarrollar una aplicación de escritorio con Windows o Java y una interfaz Web, pudiendo coexistir varias interfaces a la vez.

CAPA DE NEGOCIOS. La capa de la lógica de negocios es responsable de procesar los datos recuperados y enviarlos a la capa de presentación y los procesos propios del negocio

CAPA DE DATOS. La capa de datos almacena los datos de la aplicación en un almacén persistente, tal como una base de datos relacional o archivos XML.

5.- N Capas



Esquema utilizado en muchas aplicaciones que operan sobre Internet. Se comparte información a nivel de la capa de negocios, con lenguajes orientados a componentes y mediante servicios web.

ANEXO III. CUESTIONARIO DE ACERCAMIENTO AL DDI

Cada pregunta se marca con una X en la casilla SI cuando se tiene más de las 2/3 partes de casos afirmativos y en NO cuando es menos de 2/3 de la realidad de casos.

No.	Pregunta	SI	NO	Nivel	PESO %
ALCANCE					
1	¿El equipo del proyecto conoce los objetivos del proyecto?			1	10
2	¿Tiene documentado el alcance y requerimientos del proyecto?			1	10
3	¿Se tiene definida una metodología para la administración del alcance?			2	20
4	¿Participan todos los actores en la definición del alcance del proyecto?			3	10
5	¿Se tiene procesos de administración de proyectos estandarizados y documentados?			3	10
6	¿Los proyectos tienen objetivos claros y medibles en relación a calidad?			4	20
7	¿Se han definido medidas de eficiencia y efectividad que orienten decisiones en nuevos proyectos?			5	20
				TOTAL	100%
TIEMPO					
8	¿Se realizan actividades de planificación y programación de tiempos?			1	20
9	¿Se tiene definido un proceso estándar para planificación y programación de tiempos?			2	20
10	¿El proceso para planificación y programación de tiempos está documentado y es usado en la mayoría de proyectos?			3	10
11	¿La integración en la organización considera la administración de tiempo en proyectos dependientes entre sí?			3	10
12	¿Se realiza toma de decisiones en base a información histórica de otros proyectos?			4	20
13	¿Se examinan lecciones aprendidas para mejorar proceso de administración de tiempos?			5	20
				TOTAL	100%
COSTOS					
14	¿Se considera presupuesto para el proyecto?			1	20
15	¿Está definido un proceso para estimar costos, realizar reportes y medir rendimientos?			2	20
16	¿Existen plantillas de definición de costos?			3	10

No.	Pregunta	SI	NO	Nivel	PESO %
17	¿En la mayoría de proyectos se usa procesos estándares para manejo de costos?			3	10
18	¿Se realiza seguimiento de costos de acuerdo a lo presupuestado y está integrado a las otras áreas de la organización?			4	20
19	¿La toma de decisiones está basada en las métricas de eficiencia y efectividad en el manejo de costos?			5	20
				TOTAL	100%
COMUNICACION					
20	¿Existe comunicación constante relacionada al proyecto entre los miembros del equipo?			1	20
21	¿Existe algún proceso formal definido para la Comunicación?			2	10
22	¿Existen revisiones del avance del proyecto de forma permanente?			2	10
23	¿Se tiene establecido un plan de comunicación?			3	20
24	¿Los planes de comunicación se integran al plan de comunicación de la organización?			4	20
25	¿Existen procesos de mejoramiento continuo de comunicación basado en lecciones aprendidas?			5	20
				TOTAL	100%
RIESGOS					
26	¿Su organización considera el riesgo asociado al proyecto?			1	25
27	¿Tiene un proceso documentado para determinar la probabilidad e impacto de los riesgos?			2	25
28	¿Se usan métricas para soporte a la toma de decisiones de riesgos a nivel de proyecto?			3	12,5
29	¿Se realiza un seguimiento permanente de los riesgos mediante un esquema estándar aplicado a la mayoría de proyectos?			3	12,5
30	¿El sistema de riesgos está completamente integrado al sistema de tiempo, costos y recursos?			4	25
				TOTAL	100%

ANEXO IV. DETALLE DE SISTEMAS DEL DDI

AREA FINANCIERA

Sistema de Contabilidad y Presupuesto

El sistema que opera en los departamentos Financiero y de Contabilidad, fue desarrollado en el año 1974 en RPG II, se enlaza hacia archivos planos en el sistema AS/400 y trabaja con interface de caracteres

Este sistema se lo puede subdividir en dos subsistemas, CONTABILIDAD Y PROFORMA PRESUPUESTARIA DE GASTOS, cada uno de estos sistemas se asocia a una biblioteca organizada por años, y el usuario indicado luego de colocar la biblioteca activa invoca al menú que necesita. Los procesos necesarios para ejecutar las opciones de estos menús actúan independientemente, no se considera una programación en donde la salida de un proceso pueda ser considerada como entrada para otro.

Opciones de CONTABILIDAD

- Archivos de nombres
- Archivos de Mayor, Subcuentas/Auxiliares, Ingresos, Gastos
- Control de Movimientos contables
- Informes
- Control de Bancos
- Pagos de la Asociación al Almacén Universitario
- Informes Almacén Universitario
- Proforma Presupuestaria de Gastos
- Proforma Presupuestaria de Ingresos
- Selecciona año de contabilidad
- SIGEF INTEGRADOR

Opciones de PROFORMA PRESUPUESTARIA DE GASTOS

- Crea Archivo para Proforma
- Elimina Archivo para Proforma
- Ingreso, actualización y listado de Nombres
- Organización del Archivo de Nombres
- Pasa nombres a Archivo para Proforma
- Actualización y Consulta al Archivo para Proforma
- Clasificación del archivo para Proforma
- Organización del archivo para Proforma
- Informes
- Pasa distributivo de sueldos a Proforma
- Pasa datos de PROFORMA de GASTOS a Archivo de GASTOS
- Proforma presupuestaria de Ingresos

Sistema de Tesorería

El sistema de tesorería fue desarrollado en el año 2006 en lenguaje RPG II y funciona en el área de Tesorería para el control de especies y vales, se enlaza con archivos planos en el sistema AS/400 y trabaja con interface de caracteres

Opciones del sistema de TESORERIA

- Control de bancos / CONTROL DE CHEQUES
  - Ingreso y actualización de datos
  - Ingreso de datos a través de un archivo en disco
  - Consulta al archivo de cheques - Por Cheque - En Alfabético
  - Listado del archivo de cheques
  - Libro de bancos (Emisión)
  - Parte diario de caja
  - Elaboración de cheques (Emisión)
  - Pasa de registrado a girado
  - Pasa de girado a registrado
  - Pasa de girado a cobrado
  - Pasa de cobrado a girado
  - Creación de un archivo de cheques
  - Adiciona cheques
  - Actualiza cheques anter; poster. y saldo
  - Ingresa y actualiza nombres de bancos
  - Consulta bancos
  - Control de DEPOSITOS/TRANS/NOTAS DEBITO
- Edición de documentos / DOCUMENT
  - Trabajar con documentos
  - Salvar un documento
  - Restaurar un documento
  - Control de Especies
  - Mantenimiento de Especies Compras
  - Reporte de Compras
  - Ventas
  - Reporte de Ventas
  - Acumulado
  - Mantenimiento de Movimientos
  - Reporte de Stock
  - Reporte de Movimientos por Item
  - Visualizar Mensajes
  - Visualizar Salidas Impresas
  - Trabajar con Impresoras

Sistema de Activos Fijos

Sistema desarrollado para el control de los activos de la Universidad, utilizando la arquitectura de 3 capas con interface Web, desarrollado como proyecto de tesis, para corregir deficiencias que presentaban sistemas anteriores de activos fijos e incluir mejoras, se interconecta con el sistema de Recursos Humanos, y con el sistema SCGI (para generar órdenes de ingreso sin tener que volver a digitar los activos adquiridos, además posee un módulo para realizar consultas dinámicas con la facilidad de reconocer códigos de barras para el ingreso de nuevos códigos de activos. El sistema tiene una arquitectura de 3 capas con Interfaz web, un módulo generador de reportes, un enlace con el sistema de Recursos Humanos, adicionalmente tiene un sistema para la impresión de códigos de barras, el mismo que está construido en una plataforma Win32, con Visual Basic 6.0 como herramienta de desarrollo.

La funcionalidad del sistema, representada cada una por su interfaz de usuario es:

Mantenimientos

- Mantenimiento de activos
- Mantenimiento de actas
- Levantamiento físico
- Mantenimiento de usuarios
- Mantenimiento de dependencias
- Mantenimiento de órdenes

Activos
- Cambios de Cuenta
- Transferencias

Actas
- Acta de constatación física
- Acta de entrega-recepción

Levantamientos Físicos
- Realizar levantamientos de activos
- Ejecutar levantamientos

Consultas
- Listados , Reportes
- Generador de reportes

Usuarios
- Mantenimiento de usuarios
- Creación de Perfiles
- Cambio de contraseña

Dependencias
- Mantenimiento de Dependencias

Ordenes
- Orden de Ingreso
- Orden de Egreso

Subsistema de Activos.- Encargado de la transferencia entre dependencias y cuentas de los activos fijos y de control.

Subsistema de Actas.- Encargado del mantenimiento de las actas de constatación física y de entrega recepción.

Subsistema de Levantamiento Físico.- Encargado de la creación y ejecución de los levantamientos físicos que se realizan sobre los activos.

Subsistema de Consultas.- Encargado de los reportes y listados de los activos fijos y de control, además posee un generador de reportes para consultas no programadas.

Subsistema de Usuarios.- Encargado del mantenimiento de los usuarios del sistema.

Subsistema de Dependencias.- Encargado del mantenimiento de las dependencias existentes en la Universidad de Cuenca.

Subsistema de Ordenes.- Encargado de generar órdenes de ingreso y egreso de los activos.

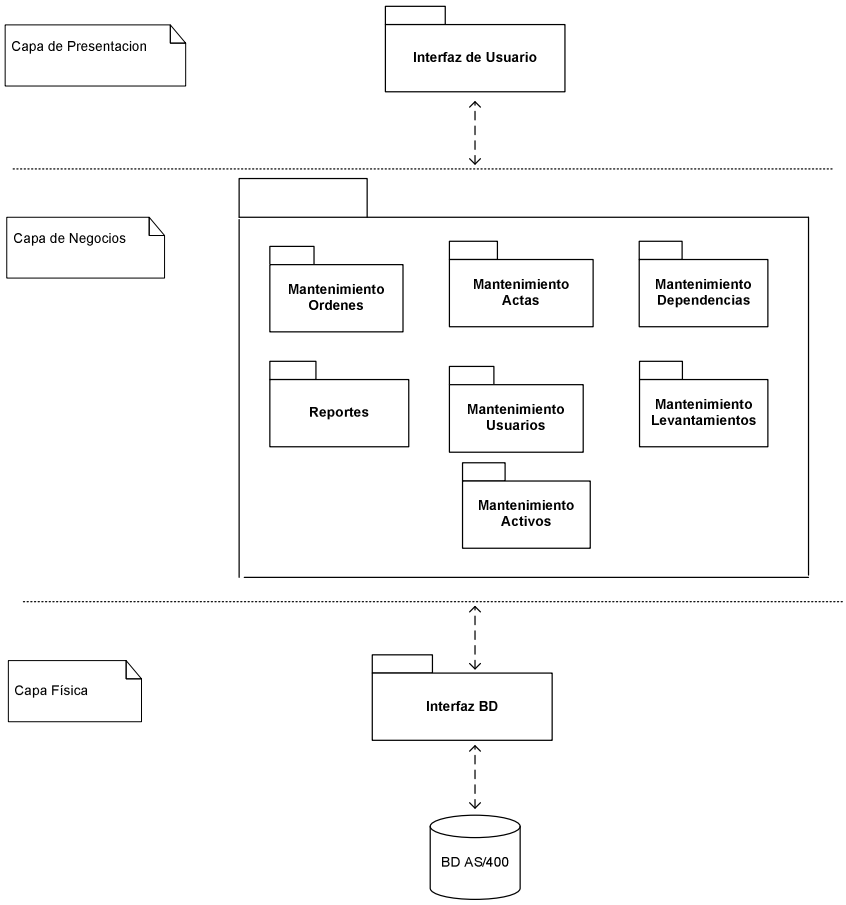


Figura 38. Diagrama de Paquetes Sistema Control Activos Fijos. (de Auquilla y Otros, 2005)

Sistema de Gestión Socio Económica

Este sistema denominado de Gestión Socio Económica, fue desarrollado bajo un análisis conjunto por personal del Departamento de Desarrollo Informático, por personal del Departamento de Admisión y Becas y por estudiantes egresados de la carrera de Ing. de Sistemas, bajo el objetivo de elaborar un sistema de Gestión Socio Económica, que permitirá realizar de manera eficiente las tareas del departamento de Admisión y Becas y su vinculación con las demás dependencias con las que tiene relación.

El proyecto de desarrollo propuso crear un sistema que facilite y sea eficiente en: el manejo de los procesos de cálculo de costos de matriculas, cuentas por cobrar, pagos realizados, actividades del área de trabajo social (becas, exoneraciones, y verificaciones).

Se utilizó la herramienta Visual Studio .NET 2005 para desarrollar el sistema, tomando la arquitectura de tres capas, para crear una aplicación orientada hacia el web, que utiliza como fuente de información la base de datos DB2 que reside en un servidor AS/400 y almacena información en ORACLE.

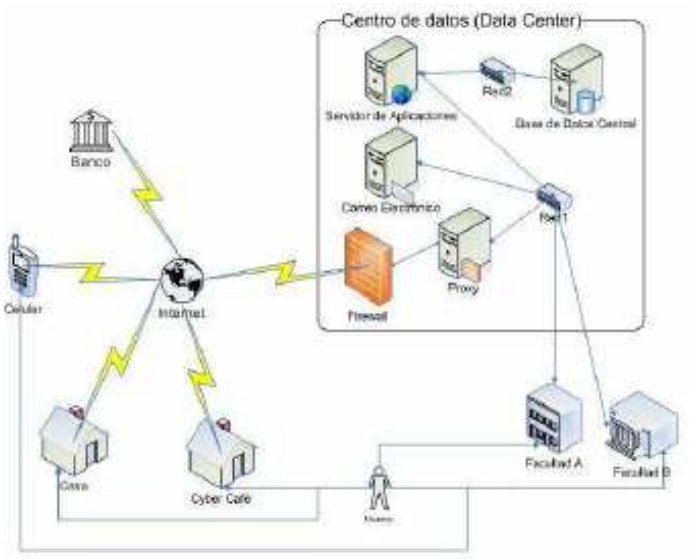


Figura 39. Esquema de Interconexión SGSE (de Robles y otros, 2008)

Paquetes del Sistema

Paquete de comunes
- Buscar Alumno
- Activar cuenta de usuario
- Solicitar cuenta de usuario

Paquete de gestión de la configuración
- Configurar parámetros de fórmulas de puntaje
- Configurar Costos
- Definir descuentos/recargos por criterios
- Establecer período ordinario y extraordinarios
- Mantener planes de financiamiento
- Configurar planes de financiamiento
- Mantener tipos de proceso

Paquete de gestión de matrícula
- Mantenimiento de carreras
- Mantenimiento de niveles académicos
- Mantenimiento de períodos lectivos
- Mantenimiento de ámbitos
- Registrar proceso
- Aprobar inscritos

Paquete de gestión de cuentas
- Configurar plan de cuentas
- Definir conceptos de cobro
- Definir conceptos de descuentos/recargos
- Aplicar exoneración individual
- Realizar refinanciamiento de deudas
- Enviar información a canales de cobro
- Procesar información de canales de cobro
- Generar emisión de pago

- Generar interés por mora
- Fijar interés y fecha máxima de pago
- Generar Facturas
- Anular Factura
- Generar reporte económico
- Emitir certificado
- Imprimir comprobante de pagos realizados
- Obtener listados de deudores

Paquete de Gestión Socio Económica

- Registrar situación socio económica
- Registrar petición de exoneración
- Situación socio económica para informes
- Aprobar informe de exoneración
- Actualizar información de alumno
- Anular beneficio de beca
- Anular informe de beca
- Registrar solicitud de exoneración
- Prefijar descuentos y recargos

Arquitectura del SGSE

La arquitectura del sistema está dividida en cuatro grandes paquetes

- Paquete de Gestión de la Configuración
- Paquete de Gestión de Gestión de la Matrícula
- Paquete de Gestión de Gestión de Cuentas
- Paquete de Gestión de Gestión Socio Económica

El sistema de gestión socio económica fue diseñado para ser parte del sistema SIUC (Sistema Integrado de la Universidad de Cuenca) que en el tiempo de implementación se lo venía desarrollando a cargo del DDI. El SIUC fue desarrollado con una arquitectura de 3 capas (Robles, y otros, 2008)

Capa de acceso a datos.- Formada por proyectos de clases de Visual Basic .Net, uno por cada sistema que forma parte del SIUC. Se tiene un proyecto general para acceder a datos que son comunes para todos los sistemas llamado AD_Generales. Dentro de este sistema SIUC, para este sistema se utiliza el proyecto AD_GestionSE, que genera extensiones .dll que serán referenciadas por las reglas de negocio. La capa de acceso a datos accede a dos bases DB2 y Oracle, la una base para tomar información compartida con otros módulos como el de recursos humanos, la otra para almacenar información del proceso de matrículas

Capa de reglas de negocio.- La capa de reglas de negocio está formado por proyectos de VB.NET del tipo de librerías de clases que utiliza archivos .vb que al igual que en la capa de acceso a datos existe un proyecto por cada sistema del SIUC y uno adicional para procedimientos comunes para todos los sistemas llamado RN_Generales, el proyecto utilizado es RN_GestionSE, dividido en 3 paquetes:

- Configuración.- Contiene las reglas para la configuración del sistema, de acuerdo al período lectivo
- Cuentas.- Contiene los métodos y atributos de una cuenta
- Recaudación o Gestión de la Matrícula.- En esta clase se encuentran las reglas de gestión de matrículas

Capa de Interfaz.- Consta de aplicaciones Web, una por cada sistema del SIUC y además 2 adicionales que son utilizadas para la administración del SIUC y para controlar el ingreso a las distintas aplicaciones

Para el correcto funcionamiento se debió integrar el Sistema SGSE con el sistema académico de ciencias económicas y con el antiguo sistema del Departamento de Admisión y Becas, para tener acceso a la información de los estudiantes, la misma conexión que fue solucionada accediendo directamente a la base de datos del DAB mediante una consulta sobre una vista. La integración con el sistema de ciencias económicas se realizó mediante un servicio que envía información de los números de créditos en los que el alumno se debe matricular y para informar a ciencias económicas si los alumnos han cancelado el valor de las matrículas. (Robles, y otros, 2008)

Sistema de Gasto Interno: SCGI

Sistema desarrollado como tesis para llevar el control del gasto interno de todas las dependencias de la universidad de cuenca, mediante la interacción de Bodega General, que lleva el inventario de los productos internos de la Universidad, Proveeduría para la interacción con proveedores de los productos, Contabilidad para la aprobación de trámites y Dependencias para la generación de trámites de Gasto Interno.

El proyecto de implementación del sistema contemplaba el análisis, diseño, desarrollo e implantación de un Sistema de Información con arquitectura 3 capas para el Control del Gasto Interno de la Universidad de Cuenca, desarrollado en el lenguaje Visual Basic de la plataforma .NET en la versión 2003, con la interface operando hacia el Web.

El sistema comprende los siguientes subsistemas:

Gestión de Dependencias.- Subsistema que funciona mediante interface Web, desde cualquier dependencia, está conformado por 4 módulos principales:

a) *Requerimiento Interno:* Permite realizar un requerimiento de materiales o suministros de Bodega General, que la dependencia necesita para su consumo.
b) *Solicitud de Compra de Bienes:* Permite realizar una solicitud de compra de bienes, que la dependencia o facultad necesita.
c) *Solicitud de órdenes de Pago:* Permite realizar una solicitud de orden de pago por concepto de algún servicio que la facultad o dependencia a solicitado.
d) *Solicitud de movilización:* Permite realizar una solicitud de movilización de personal que pertenece a una dependencia de la Universidad de Cuenca

Gestión de Contabilidad.- Subsistema que funciona mediante una interface web, permite aprobar y deshacer la aprobación de un trámite generado por una dependencia de la Universidad, también permite realizar el mantenimiento de información como: grupos

geográficos, lugares de destino, niveles de autoridad y coeficientes para el cálculo de viáticos.

Gestión de la Bodega General.- Subsistema que funciona mediante una interface web y permite controlar el ingreso y egreso de suministros y materiales solicitados a la bodega general para el consumo interno en la Universidad

Gestión de Proveeduría: Subsistema que funciona mediante interface web y permite controlar los pedidos que se realicen a los diferentes proveedores con los cuales se relaciona la Universidad de Cuenca, mediante el departamento de proveeduría, además hace mantenimientos de proveedores, clasificadores de ítems de proveeduría, ítems de proveeduría.

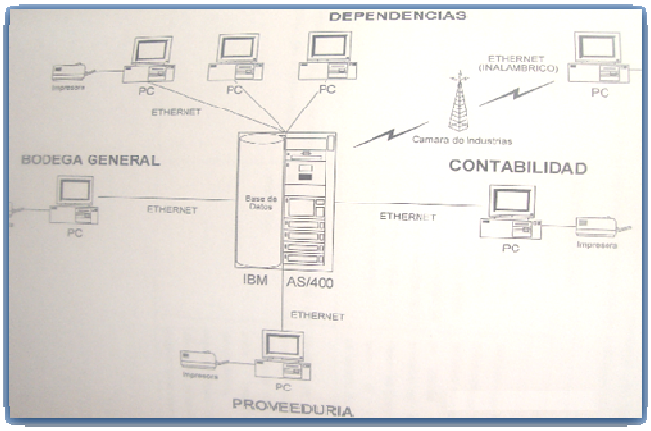


Figura 40. Modelo Geográfico Sistema Control de Gasto Interno (de Piedra y otros, Pg 64, 2004)

La Base de Datos utilizada para el desarrollo del Sistema de Control de Gasto Interno es Db2, nativa del equipo AS400, que agrupa objetos de Base de Datos (tablas, vistas, índices, procedimientos almacenados, etc) en bibliotecas. La biblioteca que contiene todos los objetos de base de datos del sistema SCGI puede accederse mediante el uso del navegador de operaciones que forma parte del paquete de productos del Client Access V5.2. de IBM instalado bajo windows (Piedra, y otros, 2004)

AREA ADMINISTRATIVA

Sistema de Gestión de Personal

El Departamento de Recursos Humanos utilizó por muchos años un sistema basado en programación RPG II sobre una plataforma IBM AS400 con tecnología empleada obsoleta, impidiendo brindar servicios de calidad, para su funcionamiento y mantenimiento se requería de personal especializado; como un intento de solventar esta situación se desarrolló una tesis de la Escuela de Informática de la Facultad de Ingeniería para la administración de personal de la Universidad de Cuenca, que por falta de un correcto seguimiento y control no se ajustó a los requerimientos de funcionalidad e integración

necesarias para la Universidad, por lo que se presentó un proyecto para implementar un Sistema de Gestión de Personal al tener una base de información íntegra, confiable, oportuna y transparente; útil para el manejo eficiente del personal de la Universidad de Cuenca. Los objetivos del proyecto fueron desarrollar un Sistema de Gestión de Personal que integre los procesos del Departamento de Recursos Humanos y elaborar un sistema confiable, seguro, útil y flexible a posteriores necesidades de la institución.

Los subsistemas en los que se divide son los siguientes:

- Administración del sistema
- Gestión de personas
- Gestión de servidores
- Nómina

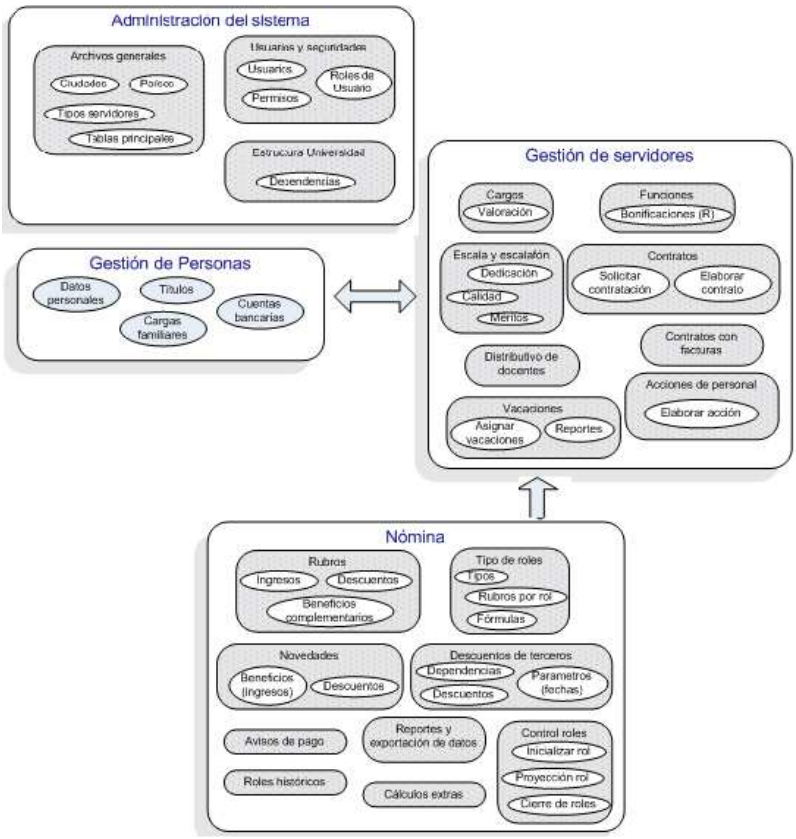


Figura 41. Subsistemas del sistema de gestión de personal

Administración del Sistema.- En el subsistema para Administración se gestiona todas las tablas necesarias para el funcionamiento del proyecto.

- Manejo las tablas generales para todos los subsistemas.
- Administración de usuarios y seguridades donde se permite parametrizar las diversas funciones a las que los usuarios pueden tener acceso
- Manejo flexible de los diversos menús utilizados en el sistema
- Manejo de la Estructura Universitaria (dependencias, departamentos, facultades, escuelas, etc.)

Gestión de Personas.- En el subsistema Gestión de Personas se maneja la información general para toda persona.

- Datos personales gestiona información como: nombres, apellidos, cédula, nacionalidad, fecha de nacimiento, estado civil, etc.
- Títulos, permite manejar un historial de los títulos académicos de una persona.
- Cargas familiares
- Cuentas bancarias, permite gestionar un historial de cuentas de una persona, con la condición de tener solamente una cuenta activa para realizar los pagos.

Gestión de Servidores.- En el subsistema gestión de servidores se maneja información adicional que vincula a los servidores con la Universidad.

- Cargos: registro de datos del cargo, sueldo básico y valoración del mismo.
- Funciones: permite diferenciar aquellos cargos que tienen bonificaciones funcionales (representación, residencia, responsabilidad).
- Escala y escalafón: manejo de méritos del servidor.
- Contratos: permite gestionar todos los contratos que realiza la Universidad con sus servidores.
- Contratos con facturas: módulo que tiene que ser analizado en pos del control de de contratos profesionales por obra cierta.
- Acciones de personal: permite gestionar toda la información y los cambios que realiza una acción sobre un servidor.
- Distributivo de docentes: gestiona el total de horas que cada profesor dedica a docencia, investigación, dirección de tesis, extensión y difusión, consultoría y servicios, planificación y administración académica.
- Vacaciones: gestiona y registra las vacaciones de empleados y trabajadores.

Nómina.- En el subsistema de nómina se gestiona información para generación de roles.

- Rubros: maneja lo referente a los diferentes campos que intervienen en el cálculo del rol de pagos entre descuentos e ingresos.
- Tipos de roles: a través de esta tipificación se permite parametrizar diversos roles que tienen su propia fórmula de cálculo inclusive cada rubro tiene su fórmula. Por ejemplo rol para personal administrativo, rol para jubilados, rol para docentes contratados, etc.
- Novedades: permite el registro por parte de Recursos Humanos de todos los ingresos y descuentos que se debe realizar a cada servidor. Por ejemplo una cuota voluntaria del servidor para cierta actividad en la Universidad y que debe ser descontada en roles.
- Descuentos de terceros: se gestionan todos los descuentos que realizan entidades externas como son APUC, AETUC, Comisariato, etc.
- Avisos de pago: generación del rol individual a cada servidor y emitir un aviso sea por email, impreso, etc.
- Control de roles: permite realizar el cierre mensual de cada rol y a la vez inicializar los archivos para el siguiente mes.
- Cálculos extras: permite realizar los pagos para décimo tercero, aguinaldo, décimo quinto, etc. Bonificaciones que no se pagan mensualmente.
- Reportes y exportación de datos: se maneja todo reporte referente a la información de los roles y generación de archivos para contabilidad, tesorería, impuesto a la renta, etc.
- Roles históricos: permite tener acceso a reportes de todo rol generado con anterioridad.

AREA ACADEMICA

Sistema Académico anterior, operando al momento

Tradicionalmente la información académica en la Universidad de Cuenca se la llevaba de forma independiente en cada Facultad, si bien durante la décadas de los años 1980 y 1990 ya se contaban con medios que automatizaban estas tareas, la información continuaba manteniéndose de forma separada en cada dependencia universitaria. En el año 2006 al presentarse la necesidad de contar con un sistema global que abarque a toda la universidad, sus facultades y escuelas, se propuso un tema de tesis que tenía la característica de poder adaptarse (tras ciertas configuraciones) a cualquier facultad de la universidad en cuanto a sus variantes en el manejo de lo académico (Zumba, y otros, 2006).

El sistema se desarrolló en una arquitectura Web de Tres Capas, con el propósito de ser un sistema centralizado, brindando la posibilidad de acceder a información desde cualquier pc en la red interna de la universidad, pues provee de ciertas opciones que los usuarios puedan utilizar a través del Portal Web de la universidad y de Internet. Pero no pudo cumplir con su propósito de constituirse en una fuente de datos troncal que pretendía alimentar a otros sistemas de la universidad puesto si bien el sistema se implementó en muchas Facultades, no se pudo implementar en la totalidad de estas al no poder cumplir con el objetivo inicial de ser diseñado para ser adaptable y flexible a las posibles variantes que se dan en cada facultad. En la práctica se encontró que las variantes, procesos y reglamentos que se manejaban en cada dependencia hacían necesaria una reimplementación del sistema para personalizarlo y reprogramar mucho código adaptándolo a cada Facultad. Las herramientas de desarrollo fueron previamente elegidas antes de asignarse el tema de tesis, según los productos con los que contaba la Universidad de Cuenca, se desarrolló con tecnología de Visual Studio .NET: Visual Basic .NET y ASP.NET, además con el DBMS IBM DB2 para AS/400. Debido principalmente a las necesidades en el acceso a la información Académica, se optó por desarrollar el Sistema Académico en base a una arquitectura de Sistema Web de Tres Capas, y dada la naturaleza de la programación propuesta en .NET, es decir por objetos, se optó por seguir el PUD (Proceso Unificado de Desarrollo) junto a UML (Unified Modeling Language) para organizadamente enfrentar la complejidad del desarrollo del Sistema Académico, además de ser totalmente apropiado para una programación orientada *a objetos* como la que propone *.NET* (Zumba, y otros, 2006)

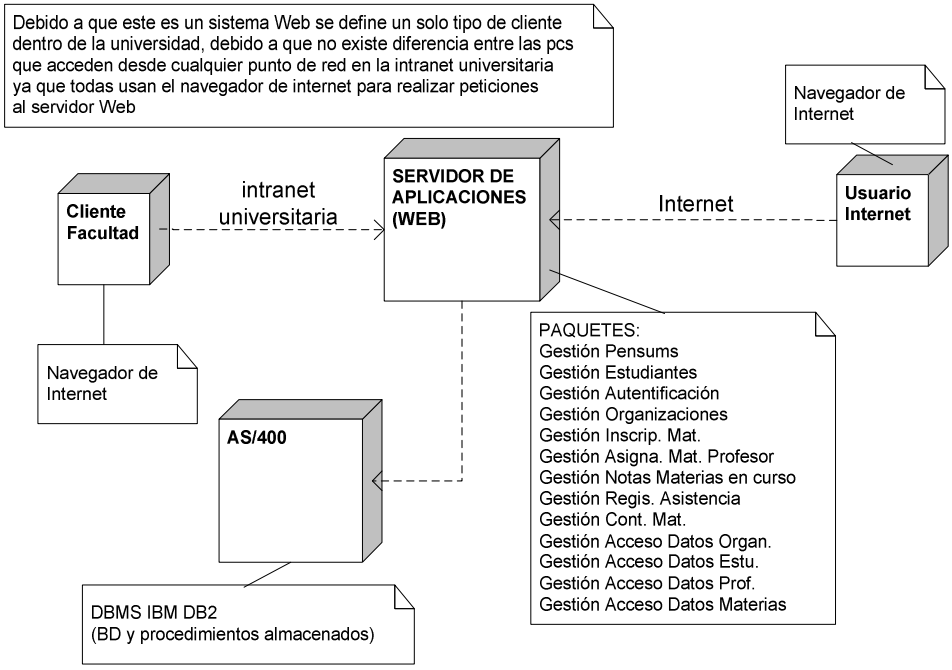
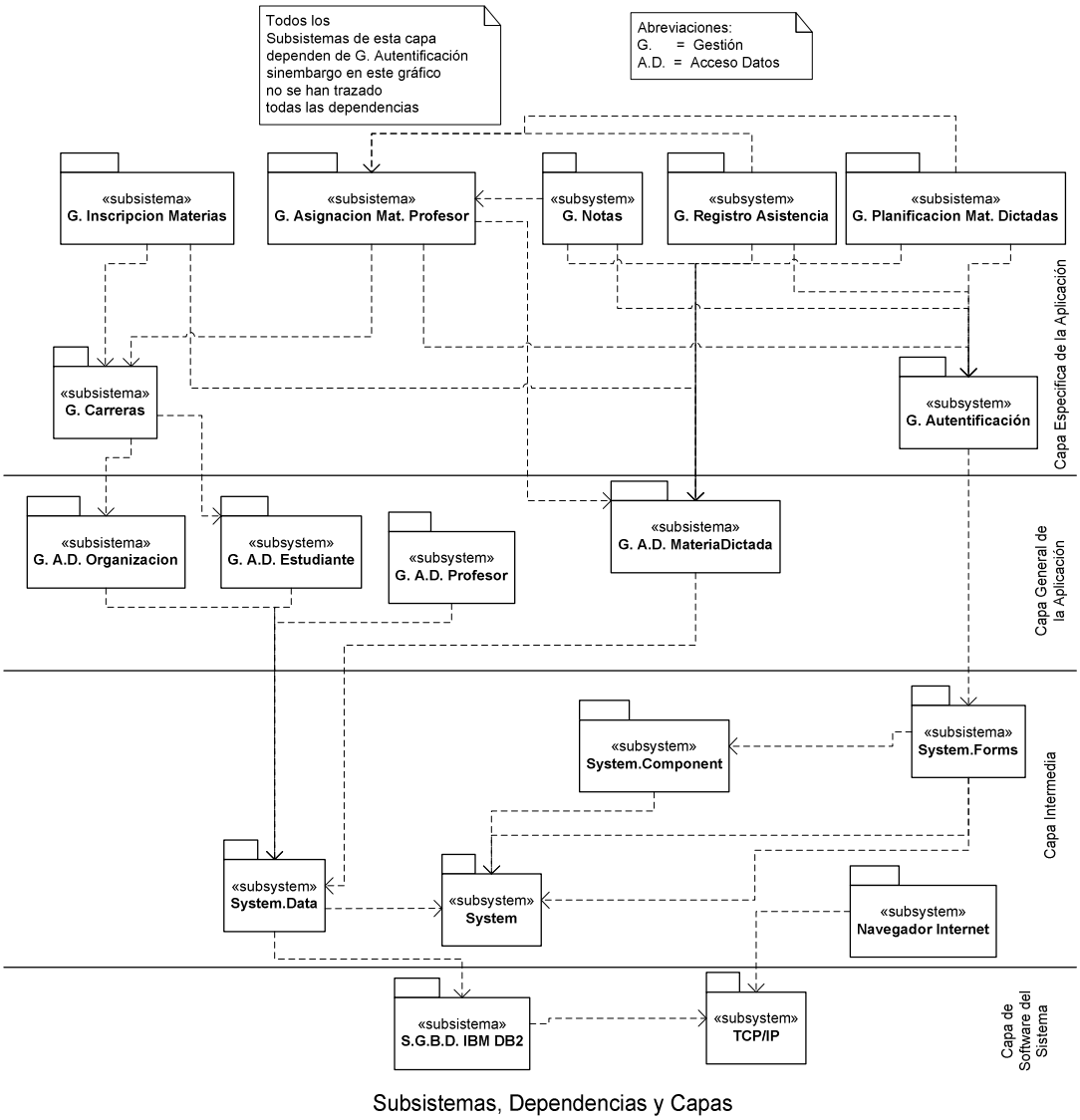


Figura 42. Organización de nodos del Sistema Académico Anterior (de Zumba José y otros, 2006)

Los actores Identificados para el sistema fueron Administrador, Profesor, Director de Escuela, Secretario de Facultad, Secretario de Escuela, Estudiante

Los subsistemas de los que constaba el sistema y su ubicación dentro del esquema de capas que se empleó son:

1. Subsistema Autentificación
2. Subsistema Carreras
3. Subsistema Control Materias Dictadas
4. Subsistema Entidades
5. Subsistema Horarios
6. Subsistema Labores Universitarias
7. Subsistema Materias Dictadas
8. Subsistema Paralelos
9. Subsistema Pénsums
10. Subsistema Periodos Lectivos
11. Subsistema Reglamentos
12. Subsistema Reglas
13. Subsistema Acceso Datos



Subsistemas, Dependencias y Capas

Figura 43. Subsistemas en las diferentes capas Sistema Académico anterior (de Zumba José y otros, 2006)

Sistema Académico en desarrollo

Debido a que la Universidad de Cuenca está en el proceso de implementación de una estructura curricular en base a créditos y un reglamento único de facultades, se requiere de un sistema de información sólido, eficiente e integrado para el soporte a los procesos académicos que demandan los cambios curriculares. Si bien existe un sistema académico, que no fue utilizado en todas las Facultades, el presente pretende ser un sistema integrado, a ser implementado en todas las facultades de la Universidad de Cuenca, solventando así las necesidades existentes, para automatizar las labores

académicas de manera adecuada, con una visión institucional y que brinde información exacta y oportuna para la toma de decisiones. (Olivo, y otros, 2008)

Este sistema podrá dar soluciones tanto administrativas como académicas.

En cuanto a las soluciones administrativas, se administrará los datos estratégicamente para incrementar la eficiencia y optimizar la entrega de servicios de los procesos de negocio a todos los tipos de participantes dentro de nuestros campus universitarios; soportando las funciones básicas de misión-crítica sin distraerle de su misión real – la enseñanza, el aprendizaje y la investigación.

En cuanto a lo educativo, soportan funciones como administración de cursos, matrículas y planeación académica, cada uno de los cuales soportan las metas de la institución con el fin de ayudar a los estudiantes a alcanzar sus metas.

Diagrama de Casos de Uso del Negocio que intenta reflejar con una visión muy global, los actores principales del negocio y los empleados que participan.

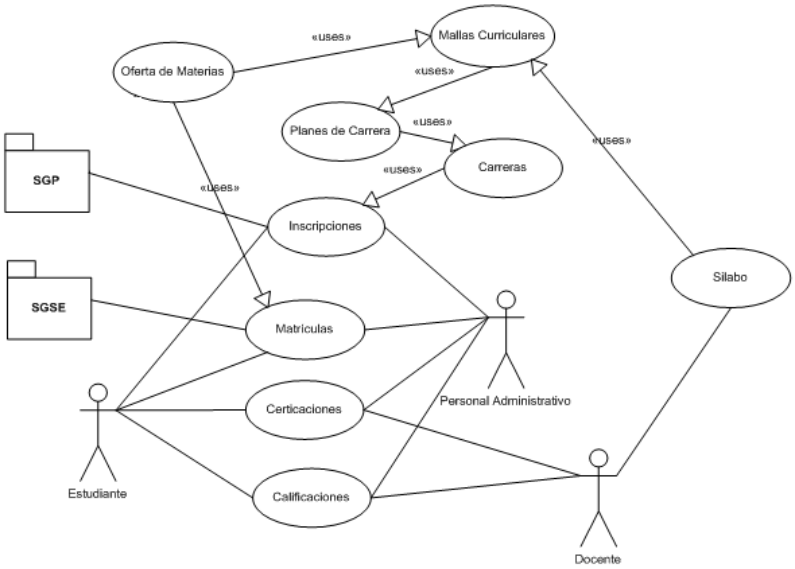


Figura 44. Casos de Uso del Negocio para el Sistema Académico en Desarrollo (tomado de Olivo y Otros, 2008)

- **Archivos Generales**.- Archivos generales y parámetros del SGA.
- **Carreras**.- Administración de todas las Carreras de Pregrado y Posgrado de La Universidad.
- **Planes de Carreras**.- Registro de los Planes de Carrera (macrocurriculum) de cada una de las Escuelas y Departamentos Académicos.
- **Mallas Curriculares**.- Registro de Asignaturas, Pensums, Tipos de Calificaciones, Prerrequisitos.
- **Inscripciones**.- Proceso de inscripciones de aspirantes a estudiar en la Universidad.
- **Oferta de Asignaturas**.- Registro de distributivos dentro de un periodo lectivo. Incluye docentes, espacios físicos y horarios de clase.
- **Sílabos**.- Registro de microcurriculums de cada una de las asignaturas de las mallas curriculares.
- **Matrículas**.- Proceso de Matriculas y auto matrículas a través de Internet

- **Calificaciones**.- Registro de calificaciones de estudiantes a través de Internet, cambios de calificaciones y convalidaciones de materias.
- **Certificaciones.**- Certificados académicos

Sistema de Bibliotecas

El sistema de bibliotecas que inicialmente fue desarrollado como un proyecto de tesis, que se lo ha venido utilizando por alrededor de 8 años, sin embargo ha surgido un nuevo requerimiento para el Centro Documental Juan Bautista Vasquez, que tiene almacenados títulos de catálogos y libros que dispone en formato CEPAL[8], los cuales necesitan ser mudados al formato MARC21[9] con lo que se une a la utilización a nivel mundial de estándares para la creación de catálogos, haciendo necesaria la actualización de la base de datos existente en el CDJBV a estos estándares, para mantenerse a la par con las tecnologías utilizadas por la mayoría de bibliotecas; con este motivo se encuentra en implementación la tesis de grado de Ingeniería de Sistemas de los señores Mauricio Brito y Andrés Baquero.

La nueva forma de codificación está soportado por un nuevo sistema informático denominado ABCD (Automatización de Bibliotecas y Centro de Documentación, software libre y de código abierto FOSS), ya desarrollado en Brasil. El personal de la biblioteca luego de realizar las investigaciones y pruebas respectivas obtuvo resultados que indican ser un sistema acorde a las necesidades actuales (Brito, y otros, 2008), y se ha planteado que el Centro de Documentación integre el sistema ABCD al Portal del CDJBV, la integración del sistema D-Space para la ayuda en el manejo de archivos digitales, la integración de los diferentes lugares que tienen información útil para consulta estudiantil o de personal universitario dentro del campus y la implementación de servicios especializados necesarios para el funcionamiento del CDJBV y difusión del material existente dirigidos a la comunidad universitaria y personas externas que requieren de servicios bibliotecarios.

[8] *CEPAL: Formato normalizado para procesamiento de información bibliográfica, desarrollado por la Comisión Económica para América Latina y El Caribe.*
[9] *MARK21: Registro Cátalo gráfico Legible por Maquina. Formato diseñado para contener información bibliográfica, utilizado en la mayor parte del mundo*

El proyecto implementado plantea los objetivos:

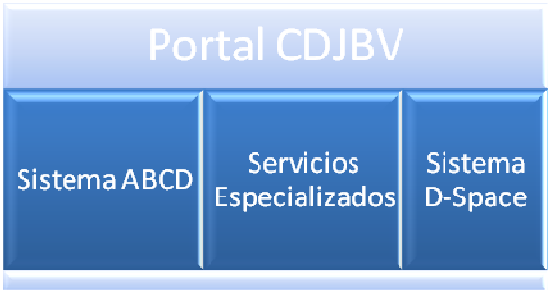


Figura 45. Portal Integrado del CDJBV

- Integración del portal WEB del CDJBV
- Implementación del sistema ABCD y unificación de los diferentes repositorios de información bibliográfica en la Universidad
- Implementación del sistema D-Space.
- Diseño e Implementación de servicios especializados.

Módulo D-Space.- Implementación del sistema D-Space para facilitar al personal del CDJBV el tratamiento y gestión de los documentos PDF, como son las tesis digitales y Libros Coloniales, así como también facilitar el uso de las mismas a los usuarios permitiendo realizar búsquedas dentro de estos documentos.

Módulo ABCD.- Adaptación del sistema ABCD a las necesidades del CDJBV, lo que incluye ingreso de los documentos bibliográfico-documentales, reportes personalizados de la información del CDJBV, registro de préstamos de los documentos bibliográfico-documentales del CDJBV, actualización de la información en los registros de los documentos bibliográfico-documentales del CDJBV, conversión de la base de datos existente del CDJBV del formato CEPAL al formato MARC 21, brindar acceso a la información de los documentos bibliográfico-documentales del CDJBV a nivel de la red de Internet y obtener acceso a la información compartida de los documentos bibliográfico-documentales por otras bibliotecas con la Integración de los diferentes centros de información de la Universidad de Cuenca.

Módulo servicios.- Servicios especializados orientados a brindar un determinado soporte a los usuarios en el servicio que buscan, cada uno de ellos según sus necesidades; atendiendo a los diferentes usuarios al obtener información sobre cualquier elemento bibliográfico, desde cualquier lugar en donde se cuente con acceso a Internet, también debe implementarse un módulo de préstamos de libros e información de estado de deudores

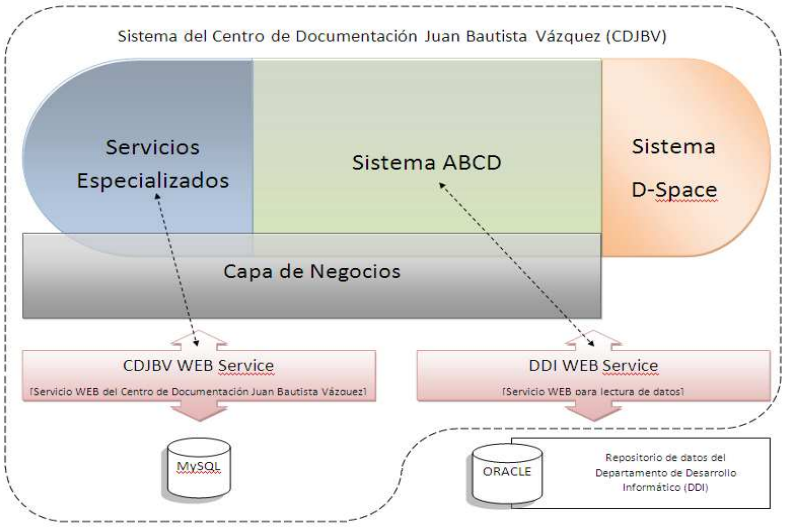


Figura 46. Esquema de Arquitectura a Emplear en CDJBV

Sistema de Administración ADMINUC

Se lo creo a partir del sistema socio económico, por la necesidad de logearse de forma única en todos los sistemas universitarios, se tendrá los accesos que tiene cada usuario sobre que facultades, escuelas y carreras, se tiene un esquema de administración generalizado, ahora viene un sistema de administración.

Está en desarrollo como parte del Sistema Académico, no existe documentación sobre el mismo, sin embargo por información verbal se conoce que tiene la estructura general:

- Dependencias
- Personas
- Ubicaciones (ecuador, país, ciudad
- Espacios físicos (dependencia, bloques, pisos, aulas. No hay una estructura firme tratando de no atarle a una facultad
- Parámetros generales
- Usuarios
- Roles
- Permisos
- Sistemas y Menús universitarios

ANEXO VI. METODOLOGIAS DE DESARROLLO EMPLEADAS EN EL DDI

En esta sección se presenta de forma detallada algunas metodologías con las que se llevaron a cabo algunas propuestas de desarrollo de algunos sistemas dentro del DDI.

a) SISTEMA DE GESTIÓN FINANCIERA – SGF

Propuesto por el Ing. Rodrigo Padilla en el año 2006, plantea seguir un tradicional ciclo de vida compuesto por las fases

- *Análisis de requerimientos y documentación de procedimientos*, en donde se plantea elaborar el manual de procedimientos (MP) y el documento de documento de especificación de requerimientos de software (SRS), mediante reuniones de trabajo con todo el personal involucrado en cada proceso para definir el proceso óptimo y las necesidades de información.
- *Diseño del Sistema,* se plantea elaborar el diseño del sistema creando prototipos, validados con los usuarios finales en una reunión de trabajo antes de iniciar la elaboración de los programas.
- *Programación.-* El equipo de desarrollo del sistema elabora los módulos del Sistema, realiza las pruebas de calidad necesarios y creará los manuales de usuario del sistema o módulo.
- *Pruebas.-* Se plantea que el sistema será instalado y probado en sus diferentes módulos y opciones para verificar si cumple con los requerimientos solicitados para ello se planea elaborar un plan de pruebas del sistema o módulo elaborado por el equipo de desarrollo del sistema en coordinación con el equipo de coordinación del SGF. Luego se piensa realizarlos ajustes necesarios al sistema en caso de existir para ser sometidos nuevamente a una validación por los usuarios, hasta que estos estén satisfechos.
- *Transición en paralelo*.- Finalmente se planifica instalar los módulos del sistema, un proceso de capacitación a usuarios y un proceso de migración de datos de los sistemas anteriores. Los módulos del sistema se piensan serán usados paralelamente con los sistemas actuales para garantizar su funcionamiento y utilidad antes de reemplazar a los sistemas antiguos.

Equipo de Coordinación planteado

- 3 Autoridades y Directores Departamentales
- Director del Departamento Informático

- 2 Usuarios directos del sistema
- Ingeniero de Sistemas

b) Sistema Socio Económico - SGSE

Propuesto por el Ing. Pablo Garrido y la Ing. Karol Guzmán en el año 2008, plantea utilizar la metodología de Cascada y UML como lenguaje de descripción. La metodología propuesta divide al proyecto en los siguientes flujos de trabajo:

- Requerimientos
- Análisis
- Diseño
- Implementación
- Pruebas
- Transición

Se plantea llevar un registro de las versiones y revisiones de cada elemento del proyecto. Al final de cada bimestre se plantea redactar un informe de actividades y hacer una revisión al final de cada flujo de trabajo y cada etapa. Los documentos que se obtienen al final de cada fase de acuerdo a la metodología a usarse debían ser entregados al final de cada etapa.

Organización del equipo de trabajo

- El Director del Proyecto encargado de dirigir y gestionar el proyecto,
- El Coordinador del Proyecto encargado de la coordinación y control del desarrollo del mismo y vinculación con los demás proyectos que ejecutan el Equipo de Desarrollo de Software del DDI
- El Ingeniero de Sistemas encargado del análisis y diseño del sistema conjuntamente con el Coordinador así como el control de cambios.
- Dos programadores encargados de construir e implementar código.

c) Sistema para la Gestión de Personal – SGP

En el año 2007 en el documento en el que consta la propuesta para la elaboración del Sistema para la Gestión de Personal, se plantea la siguiente metodología de trabajo: (Olivo, 2007)

- *Análisis de requerimientos y documentación de procedimientos* Plantea extraer los requisitos del producto de software como primera etapa, realizando reuniones de trabajo con todo el personal involucrado en cada proceso para definir el proceso óptimo y las necesidades de información. Estas reuniones se piensa deberán ser fuera del lugar de trabajo para garantizar la participación efectiva de las personas. Se propone que el resultado del análisis de requisitos se plasmará en el documento de Especificación de Requerimientos del Sistema

- *Diseño del Sistema* Planifica que una vez definidos los procedimientos y los requerimientos del software se debe elaborar el diseño del sistema creando prototipos, que serán validados antes de iniciar la elaboración misma de los programas, con los usuarios finales del sistema en una reunión de trabajo por funciones del sistema, fuera del lugar de trabajo.
- *Programación* El equipo de desarrollo del sistema debe elaborar los módulos del Sistema realizando las pruebas de calidad necesarios y se crearán los manuales de usuario del sistema o módulo.
- *Pruebas* Plantea que el sistema debe ser instalado y probado en sus diferentes módulos y opciones para verificar si cumple con los requerimientos solicitados por los usuarios para ello se elaborará un Plan de pruebas del sistema o módulo elaborado por el equipo de desarrollo del Sistema, para luego realizar los ajustes necesarios al sistema en caso de existir y ser sometidos nuevamente a una validación por los usuarios, hasta que estos estén satisfechos.
- *Transición en paralelo* Finalmente se planea instalar los módulos del sistema y realizar un proceso de capacitación a usuarios.

Recursos Humanos necesarios

- Un Coordinador de Desarrollo, que debe ser un Ingeniero de Sistemas
- Un Analista de Sistemas Desarrollador, que debe ser un Ingeniero de Sistemas
- Tres analistas programadores que deben ser analistas o ingenieros de sistemas
- Un experto en gestión de personal, que debe ser una persona con los conocimientos necesarios de los procesos en la gestión de recursos humanos

d) Sistema de Gestión Académica de la Universidad de Cuenca – SGA

El enfoque desarrollo propuesto por el equipo del área de desarrollo del departamento informático de la Universidad de Cuenca en el año 2008, constituye una configuración del proceso RUP. El proyecto fue planificado en dos fases: “Análisis técnico, funcional y propuesta de Diseño” y “Desarrollo e Implantación del Nuevo Sistema”. La primera fase corresponde a las fases Inicio y Elaboración en RUP. La otra fase que se plantea ofertarla posteriormente corresponde en RUP a las fases de Construcción y de Transición. (Olivo, y otros, 2008)

El plan del proyecto presentado considera que el proyecto se realizará en tres etapas, las mismas que comprenderán un número de módulos a desarrollar. Cada entrega es organizada en fases e iteraciones aplicando la tecnología RUP para cada una de ellas. Dentro del plan de las fases consta:

Primera etapa

- Fase de Inicio (Duración: 2 ½ semanas)
  - Documento de visión
  - Plan de desarrollo de software
  - Modelado del Negocio
  - Modelo de casos de uso y especificaciones de caso de uso correspondientes a los módulos de la primera etapa (10-20% completado).
  - Lista de riesgos
  - Especificaciones adicionales
  - Glosario inicial
- Fase de Elaboración (Duración: 2 semanas)
  - Modelo de casos de uso y especificaciones de casos de uso (al menos hasta el 80%)
  - Lista de riesgos y caso de negocio revisados
  - Modelo de datos preliminar (será revisado y completado en las etapas siguientes)
  - Se actualizan todos los productos de la fase de inicio.
- Fase de Construcción (Duración: 5 semanas)
  - Modelos completos (casos de uso, análisis, diseño y datos)
  - Riesgos presentados (de la primera etapa) mitigados
  - Construcción de los módulos correspondientes a la primera etapa
  - Plan de proyecto para la fase de transición
  - Manual inicial de usuario.
- Fase de Transición (Duración: 2 ½ semanas)
  - Prototipo operacional.

Segunda etapa

- Fase de Inicio (Duración: 3 semanas)
  - Modelo de casos de uso y especificaciones de caso de uso correspondientes a los módulos de la segunda etapa (10-20% completado).
  - Lista de riesgos para esta etapa
  - Glosario actualizado hasta esta etapa
- Fase de Elaboración (Duración: 4 semanas)
  - Modelo de casos de uso y especificaciones de casos de uso (al menos hasta el 80%)
  - Lista de riesgos revisados
  - Modelo de datos (se seguirá analizando en la última etapa)
  - Se actualizan los productos de la fase de inicio.
- Fase de Construcción (Duración: 7 semanas)
  - Modelos completos (casos de uso, análisis, diseño y datos)
  - Riesgos presentados (de la segunda etapa) mitigados
  - Construcción de los módulos correspondientes a la segunda etapa
  - Plan de proyecto para la fase de transición
  - Manual de usuario (revisado y actualizado).
- Fase de Transición (Duración: 3 semanas)
  - Prototipo operacional

Tercera etapa

- Fase de Inicio (Duración: 3 semanas)
  - Modelo de casos de uso y especificaciones de caso de uso correspondientes a la tercera etapa (10-20% completado).
  - Lista de riesgos para esta etapa
  - Glosario actualizado
- Fase de Elaboración (Duración: 3 semanas)
  - Modelo de casos de uso y especificaciones de casos de uso (al menos hasta el 80%)
  - Lista de riesgos revisados
  - Modelo de datos final
  - Se actualizan los productos de la fase de inicio.
- Fase de Construcción (Duración: 7 semanas)
  - Modelos completos (casos de uso, análisis, diseño y datos)
  - Riesgos presentados mitigados
  - Construcción de los módulos correspondientes a la tercera etapa
  - Plan de proyecto para la fase de transición
  - Manual de usuario final
- Fase de Transición (Duración: 1 semanas)
  - Prototipo operacional

Los hitos que marcan el final de cada fase se describen:

Fase de Inicio: plan de fases, identificación de los principales casos de uso e identificación de los riesgos. Además definición del alcance del proyecto.

Fase de Elaboración: Plan de proyecto, se contemplará los casos de uso y se eliminarán los riesgos.

Fase de Construcción: Terminara de analizar y diseñar todos los casos de uso, refinando el Modelo de Análisis/Diseño. Obtener un producto totalmente operativo que pueda ser entregado a los usuarios para las pruebas y el material de apoyo para los mismos (manual de usuario).

Fase de Transición: Entrega de toda la documentación del proyecto con los manuales de instalación y todo el material de apoyo al usuario, finalización del entrenamiento de los usuarios y empaquetamiento del producto.

Artefactos generados o modificados durante las fases:

Modelado del Negocio
Modelo de Casos de Uso del Negocio y Modelo de Objetos del Negocio
Requisitos
Glosario
Visión
Modelo de Casos de Uso
Especificación de Casos de Uso

Análisis / Diseño
Modelo de Análisis / Diseño
Modelo de Datos
Implementación
Prototipos de Interfaces de Usuario
Pruebas
Plan de Pruebas
Despliegue
Modelo de Despliegue
Gestión de Cambios y Configuración
Gestión del proyecto
Plan de Desarrollo del Software en su versión 1.0 y planes de las Iteraciones
Ambiente

Seguimiento y Control del Proyecto

Gestión de Requisitos.- Son especificados en el artefacto Visión. Cada requisito tendrá los atributos: importancia, estado, iteración, para realizar un efectivo seguimiento de cada requisito. Los cambios en los requisitos serán gestionados mediante una Solicitud de Cambio, que serán evaluadas y distribuidas para asegurar la integridad del sistema y el correcto proceso de gestión de configuración y cambios.

Control de Plazos.- El calendario del proyecto tendrá un seguimiento y evaluación semanal por el director y el coordinador del proyecto.

Control de Calidad.- Los defectos detectados en las revisiones y formalizados también en una Solicitud de Cambio tendrán un seguimiento para asegurar la conformidad respecto de la solución de dichas deficiencias. Para la revisión de cada artefacto y su correspondiente garantía de calidad se utilizarán las guías de revisión y checklist (listas de verificación) incluidas en RUP.

Gestión de Riesgos.- A partir de la fase de Inicio se mantendrá una lista de riesgos asociados al proyecto y de las acciones establecidas como estrategia para mitigarlos o acciones de contingencia. Esta lista será evaluada al menos una vez en cada iteración.

Gestión de Configuración.- Se realizará una gestión de configuración para llevar un registro de los artefactos generados y sus versiones, se incluirá la gestión de las Solicitudes de Cambio y modificaciones, informando y publicando los cambios para que sean accesibles a todos los participantes en el proyecto. Al final de cada iteración se establecerá una "base line" (un registro del estado de cada artefacto, estableciendo una versión), la cual podrá ser modificada sólo por una Solicitud de cambio aprobada.

Roles y Responsabilidades

Las principales responsabilidades de cada uno de los puestos durante las fases de Inicio y Elaboración, de acuerdo con los roles que desempeñan en RUP se plantean:

- Director del Proyecto quien dirija y gestione los recursos del proyecto
- Coordinador del Proyecto encargado de supervisar y controlar el desarrollo del mismo y la vinculación con los demás proyectos de software.
- Ing. de Sistemas y Programador Los Ingenieros de Sistemas y Programadores: serán los que implementen el Sistema de Gestión Académica

ANEXO VII. VISION DEL PROYECTO

Fundada en 1867
UNIVERSIDAD DE CUENCA

DEPARTAMENTO DE DESARROLLO INFORMATICO

VISION DEL PROYECTO

Versión:	[Número]
Fecha creación:	[día] de [mes] de [año]
Última actualización:	[día] de [mes] de [año]
Estado del Documento:	[Propuesta/Borrador/Validado/Aprobado]
Cliente:	[Nombre de la institución para la cual se desarrolla el proyecto]
Elaborado por:	[Nombre de la persona que elaboró el documento]
Revisado por:	[Nombre de la persona que revisó el documento]

Registro de Cambios del Documento

Liste en la matriz los cambios que se han llevado a cabo sobre el documento de "Visión del Proyecto", detallando cuándo se realizaron, quién modificó el documento, sobre qué versión se trabajó y el detalle de los cambios efectuados.

Fecha	Autor	Versión	Estado	Cambios realizados

* Estado se refiere al estado del documento y puede ser: Borrador / Propuesta / Validado / Aprobado.

Revisores

Liste en la matriz los nombres de las personas encargadas de revisar el documento de "Visión de Proyecto", indicando la información solicitada.

Nombre	Versión Aprobada	Cargo/Rol en el proyecto	Fecha

Lista de Distribución

Proporcione una lista de las personas a quienes se distribuye el documento y recolecte las firmas respectivas como muestra de su aceptación.

Nombre	Cargo	Firma de Aceptación

Tabla de Contenidos

1. Introducción

1.1 Visión General del Documento

El documento de *Visión de Proyecto* tiene por objeto reconocer formalmente la existencia del proyecto [Nombre del Proyecto], en donde se debe incluir la necesidad del negocio para lo cual el proyecto fue propuesto, la descripción del producto a crear. Este documento deberá ser generado por niveles directivos y le dará al administrador del proyecto la autoridad necesaria para aplicar recursos organizacionales a las actividades del proyecto. A través de este documento se establece la base para el proceso formal de planeación posterior.

1.2 Audiencia

La audiencia hacia la cual está dirigido el documento de *Visión*, incluye:

- [Miembro de la audiencia]
- [Miembro de la audiencia]

Añada viñetas como personas/grupos pertenezcan a la audiencia del documento de Visión de Proyecto.

1.3 Definiciones, Siglas y Abreviaturas

[***Item**: descripción*]
[***Item**: descripción*]

1.4. Referencias

- [referencia]
- [referencia]

2. Sumario Ejecutivo

En esta sección se abarca una declaración de los motivos que justifican el proyecto, la problemática de negocio y los beneficios que se obtienen de una manera concreta. Una descripción de las características del producto a crear, las cuales serán superficiales pero servirán para la toma de decisiones en las siguientes fases así como para el entendimiento de los niveles directivos.

[Inicie el texto aquí]

3. Intención del Proyecto

Incluya aquí el propósito del proyecto, la vinculación hacia otros proyectos así como las restricciones y supuestos del mismo. Lo antes mencionado constituye la intención del proyecto, la cual guiará el proceso para su justificación y aprobación. Puede incluir información referente a los grupos de interés, cuales son los resultados más importantes.
[Inicie el texto aquí]

4. Alineación y Posicionamiento

4.1.ALINEACION

Defina los Objetivos de Negocio con los que el proyecto se correlaciona a través de las iniciativas estratégicas o mediante una alineación al plan.

Objetivo del Negocio o dependencia:	[Defina aquí el objetivo del negocio o dependencia]
Plan Estratégico:	[Cite el plan estratégico, de la institución o área particular, al cual pertenece la iniciativa o aspecto estratégico mencionado anteriormente. Indique el período de vigencia del plan]

4.2. POSICIONAMIENTO

4.1.1 Enunciación del Problema

En esta sección incluya el enunciado y antecedentes del problema

4.1.2 Enunciación de la Posición del Producto

En esta sección describa la posición esperada del producto respecto a otros que existan

5. Descripción del Proyecto y Alcance

En esta sección se proporciona la estimación básica del alcance, costos y tiempos aproximados para entregar la solución al usuario.

5.1 Descripción del Proyecto

En esta sección se describen las características que tendrá el producto en su versión a ejecutaren función de ¿Qué?, ¿Por qué?, ¿Cuándo?, ¿Cómo?; además de las posibles alternativas, los resultados más importantes, como se sabe cuando se alcanzó el objetivo, cuales son los hitos claves (informes, servicios, productos específicos, capacitación, documentación).

[Inicie el texto aquí]

5.2 Objetivos del Proyecto

Objetivo General

[Incluir el objetivo general del proyecto]

Objetivos Específicos

- [*objetivo1*]
- [*objetivo2*]

Añada viñetas como objetivos específicos sean necesarios para el proyecto

5.3 Alcance y Restricciones del Proyecto

En esta sección se define el alcance y se establece los límites de lo que exactamente se va a hacer y lo que no se va a hacer en el proyecto. En este sentido, cuanto más acotado esté el alcance habrá menos riesgo en el proyecto, pues se conocerá con mayor exactitud lo que está incluido y lo que no. Es decir, hay que delimitar el proyecto de acuerdo con las especificaciones dadas por las partes involucradas.

[Incluya aquí el alcance

Alcance del Producto → características y funciones.

Alcance del Proyecto → trabajo a realizar para obtener el producto.]

Una restricción es una limitación aplicable que afectará el desempeño del proyecto. Cuando un proyecto se realiza bajo contrato, las provisiones contractuales serán restricciones.

[Incluya aquí las restricciones

Descripción: En este campo debe escribir una breve descripción de la restricción que debe ser considerada durante el proyecto.

Fase: En este campo debe escribir el nombre de la fase o fases del proyecto en que se debe considerar o en que aplica la restricción relacionada.

Efecto: En este campo debe escribir la consecuencia o resultado de la restricción.]

5.4 Características del Producto

Características Funcionales

[Descripción de las características funcionales que contendrá el producto]

Características no Funcionales

[Descripción de las características no funcionales que contendrá el producto]

5.5 Requerimientos de Documentación

En esta sección se detalla los documentos que se obtendrán al finalizar el producto tales como manuales de usuario, del sistema, Guías de Instalación y Configuración, etc.

6. Descripción de Administración, Interesados y Usuarios

6.1 Responsable del Proyecto

Determine la persona responsable del seguimiento del proyecto en primera instancia, aunque posteriormente se determinará los roles reales.

[Inicie el texto aquí]

6.2 Resumen de Interesados

Se entiende por interesados a todas aquellas personas, unidades o instituciones directamente involucradas en construir o tomar decisiones acerca de la funcionalidad y propiedades del sistema.

Nombre	Cargo	Incumbencias y Responsabilidades
[Nombre de la persona]	*[Cargo de la persona]*	*[Responsabilidad]*
[Nombre de la persona]	*[Cargo de la persona]*	*[Responsabilidad]*
[incluya otros interesados en una línea adicional]		

6.3 Resumen de Usuarios

Se entiende por usuarios a todos aquellos individuos o unidades que hagan uso del sistema y soporten sus operaciones.

Nombre	Cargo	Incumbencias y Responsabilidades
Nombre de la persona	*Cargo de la persona*	*Responsabilidad*

6.4 Ambiente de los Usuarios

Descripción del ambiente en que se operará el sistema

6.5 Compromisos

Compromiso	Asumido por	Realizado para	Fecha de vencimiento	¿Alcanzable?	Intereses propios	Intereses de involucrados

6.6 Aproximación de Recursos

Liste aquí los recursos y costos estimados para la conclusión del proyecto. Los recursos incluyen personas, equipos, facilidades y material. A partir de la estimación será posible calcular el presupuesto aproximado requerido.

Recurso	Cantidad	Costo
Hardware		
Software		
Recursos humanos		
Facilidades y materiales		
Otros		
	Costo Total:	[Σ costos]

Agregue filas adicionales según recursos involucre el proyecto.

6.7 Cronograma Propuesto

En esta sección se detallarán los tiempos necesarios para el desarrollo del proyecto. El cronograma inicial de trabajo, estimado para la ejecución del proyecto, el presente tiene por objeto definir las aproximaciones iniciales en tiempo, para cada fase.

Evento	Fecha Estimada [mmm/aaaa]	Duración Estimada
Aprobación de Proyectos		
Visionamiento		
Planeación		
Desarrollo		
Estabilización		
Despliegue		
Cierre de Proyecto		
TOTAL		

6.8. Restricciones

Detalle de las restricciones a las que se sujeta el proyecto

6.9. Riesgo

Coloque en la columna Nivel, las palabras Bajo, Medio, Alto. En función del riesgo que cada uno de los componentes de proyecto tiene.

COMPONENTE	NIVEL
Riesgo Financiero	
Riesgo Externo	
Riesgo Administrativo	
Riesgo de Misión Crítica	
Riesgo de Fracaso	
Riesgo de Complejidad	

7. Firmas de Aprobación

Las firmas a continuación significan que están de acuerdo con los términos expresados en la visión del proyecto de TI.

…….	…….
[Nombre del Cliente]	[Nombre del responsable]
Cliente de la Organización	Responsable del Proyecto

ANEXO VIII. MODELO DE REQUERIMIENTOS

DEPARTAMENTO DE DESARROLLO INFORMATICO

MODELO DE REQUERIMIENTOS

Versión:	[Número]
Fecha creación:	[día] de [mes] de [año]
Última actualización:	[día] de [mes] de [año]
Estado del Documento:	[Propuesta/Borrador/Validado/Aprobado]
Cliente:	[Nombre de la institución para la cual se desarrolla el proyecto]
Elaborado por:	[Nombre de la persona que elaboró el documento]
Revisado por:	[Nombre de la persona que revisó el documento]

Registro de Cambios del Documento

Liste en la matriz los cambios que se han llevado a cabo sobre el documento de "Modelo de Requerimientos", detallando cuándo se realizaron, quién modificó el documento, sobre qué versión se trabajó y el detalle de los cambios efectuados.

Fecha	Autor	Versión	Estado	Cambios realizados

* Estado se refiere al estado del documento y puede ser: Borrador / Propuesta / Validado / Aprobado.

Revisores

*Liste en la matriz los nombres de las personas encargadas de revisar el documento de "*Modelo de Requerimientos*", indicando la información solicitada.*

Nombre	Versión Aprobada	Cargo/Rol en el proyecto	Fecha

Lista de Distribución

Proporcione una lista de las personas a quienes se distribuye el documento y recolecte las firmas respectivas como muestra de su aceptación.

Nombre	Cargo	Firma de Aceptación

Tabla de Contenidos

1. Introducción

1.1. Visión General del Documento

El documento de Modelo de Requerimientos tiene por objeto disponer de un documento en donde se integre formalmente la documentación de los requerimientos a solucionar para el proyecto: [Nombre del Proyecto], en donde se debe incluir la necesidad del negocio para lo cual el proyecto fue propuesto, la descripción de las necesidades del producto a crear. Este documento deberá ser generado por coordinador del proyecto, el modelador y personal involucrado en el proyecto.

1.2 Audiencia

La audiencia hacia la cual está dirigido el documento Modelo de Requerimientos, incluye:

- [Miembro de la audiencia]
- [Miembro de la audiencia]

Añada viñetas como personas/grupos pertenezcan a la audiencia del documento de Visión de Proyecto.

1.3 Definiciones, Siglas y Abreviaturas

[***Item**: descripción*]
[***Item**: descripción*]

1.4. Referencias

- [referencia]
- [referencia]

2. Contextualización

2.1 Objetivos de la Organización

En esta sección se detalla el tipo de organización para la que se desarrolla el proyecto y los objetivos que la organización persigue

2.2. Esquema de la Organización

En esta sección se detallará el esquema establecido bajo el cual se constituye la organización

2.3. Modelo de Negocio

En esta sección se detallará el modelo de negocio en forma descriptiva o en un diagrama que exprese de forma general el contexto en el cual se desarrollará el proyecto

3. Procesos del Negocio

3.1. Procesos establecidos

En esta sección debe incluir el esquema de los procesos que ya se efectúan en la organización, incluir el detalle de los actores, actividades y la secuencia en las que se vienen efectuando en la práctica

3.2. Procesos por establecer

Para los procesos nuevos y luego de talleres en los que se llega a acuerdos, en esta sección debe definir los actores, actividades y la secuencia de procesos futuros

4. Dominio del Proyecto

Incluir en esta sección el modelo del dominio del proyecto, identificando las principales entidades y sus relaciones

4.1. Esquema de Clases

Crear el esquema de clases de la aplicación en base del modelo del dominio, presentado como uno o más diagramas de clases y que contiene, no conceptos propios del sistema de software sino de la propia realidad física, será es utilizado como un medio para comprender el departamento, entidad o institución al cual el sistema va a servir.

Ejm:

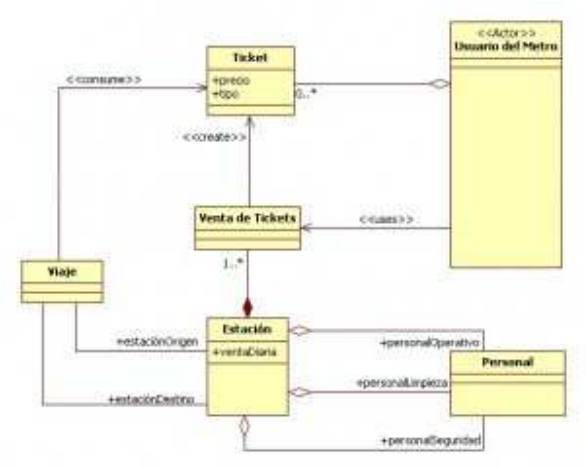


Figura 47. Ejemplo de diagrama de clases

4.2. Casos de Uso

Crear el esquema de casos de uso de la aplicación en base del modelo del dominio. Para el detalle de los casos de uso se puede utilizar la plantilla

Caso de Uso

ID Caso del Uso:			
Nombre Caso de Uso:			
Creado Por:		Actualizado por:	
Fecha Creación:		Fecha última actualizació	
Actor:			
Descripción:			
Disparador:			
Precondición:	1.		
Postcondición:	1.		
Flujo Normal:	1.		
Flujo Alternativo:			
Excepción:			
Incluye:			
Prioridad:			
Frecuencia of Uso:			
Regla de Negocio:			

Requerimientos Especiales:	
Se asume:	
Notas y Dificultades:	

5. Pruebas de Aceptación

Se definen las pruebas de aceptación del sistema las cuales serán utilizadas para posteriormente verificar que el sistema puede ser aprobado y calificado como desarrollado satisfactoriamente

6. Identificación de Servicios

Antes de que los servicios sean en realidad construidos, es conveniente establecer un modelo conceptual de todos los servicios previstos o en caso de existir ya algunos, hay que proceder con el inventario de los mismos.

7. Requerimientos Técnicos

Describe el comportamiento en temas como la usabilidad, la seguridad y el rendimiento. Se llaman también requisitos "no funcionales"

8. Glosario

Se capta el vocabulario del proyecto y se lo agrega en forma ordenada alfabéticamente como glosario explicativo de los términos utilizados en el proyecto.

ANEXO IX. MODELO DE DISEÑO

Fundada en 1867
UNIVERSIDAD DE CUENCA

DEPARTAMENTO DE DESARROLLO INFORMATICO

MODELO DE DISEÑO

Versión:	[Número]
Fecha creación:	[día] de [mes] de [año]
Última actualización:	[día] de [mes] de [año]
Estado del Documento:	[Propuesta/Borrador/Validado/Aprobado]
Cliente:	[Nombre de la institución para la cual se desarrolla el proyecto]
Elaborado por:	[Nombre de la persona que elaboró el documento]
Revisado por:	[Nombre de la persona que revisó el documento]

Registro de Cambios del Documento

Liste en la matriz los cambios que se han llevado a cabo sobre el documento de "Modelo de Diseño", detallando cuándo se realizaron, quién modificó el documento, sobre qué versión se trabajó y el detalle de los cambios efectuados.

Fecha	Autor	Versión	Estado	Cambios realizados

* Estado se refiere al estado del documento y puede ser: Borrador / Propuesta / Validado / Aprobado.

Revisores

Liste en la matriz los nombres de las personas encargadas de revisar el documento de "Modelo de Diseño", indicando la información solicitada.

Nombre	Versión Aprobada	Cargo/Rol en el proyecto	Fecha

Lista de Distribución

Proporcione una lista de las personas a quienes se distribuye el documento y recolecte las firmas respectivas como muestra de su aceptación.

Nombre	Cargo	Firma de Aceptación

Tabla de Contenidos

1. Introducción

1.1. Visión General del Documento

El documento de Modelo de Diseño tiene por objeto disponer de un documento en donde se integre formalmente el diseño del sistema para el proyecto: [Nombre del Proyecto], en donde se debe incluir la Arquitectura de la Aplicación, Esquema de clases, Diagrama Entidad Relación, Detalle de requerimientos, Casos de prueba de aceptación, Definiciones de calidad, aspectos de riesgos y seguridad. Este documento deberá ser generado por el modelador, el desarrollador, administrador de la base de datos y personal involucrado en la especificación y construcción del sistema. A través de este documento se establece la base para el desarrollo de software posterior.

1.2 Audiencia

La audiencia hacia la cual está dirigido el documento de Modelo de Diseño, incluye:

- [Miembro de la audiencia]
- [Miembro de la audiencia]

Añada viñetas como personas/grupos pertenezcan a la audiencia del documento de Visión de Proyecto.

1.3 Definiciones, Siglas y Abreviaturas

[***Item**: descripción*]
[***Item**: descripción*]

1.4. Referencias

- [referencia]
- [referencia]

2. Detalle de Requerimientos

En esta sección se realiza una descripción detallada para los requerimientos que lo necesiten (cuya funcionalidad no sea evidente o que no baste con una simple descripción narrativa), para casos de uso cuyo flujo de eventos complejos podrá adjuntarse una representación gráfica detallada mediante

- *Diagramas de flujo,*
- *Diagramas de actividad*
- *Diagramas de secuencia*

3. Esquema de Clases

En esta sección plantear un esquema con las clases a emplear en el sistema, este modelo establece la realización de los casos de uso en clases y pasando desde una representación en términos de análisis, sin incluir aspectos de implementación, hacia una de diseño que incluye una orientación hacia el entorno de implementación.

4. Modelo de Datos Físico

En este punto crear un modelo de datos físico, previendo que la persistencia de la información del sistema será soportada por una base de datos relacional, este modelo describe la representación lógica de los datos persistentes, de acuerdo con el enfoque para modelado relacional de datos. Para expresar este modelo se utiliza un diagrama entidad-relación para conseguir la representación de tablas, claves, etc. Para la Descripción de Tablas y Campos utilice el siguiente esquema:

Tabla *<Nombre de la Tabla, de acuerdo a estándares definidos en el DDI>*			
<Descripción de la información que se almacenará en la tabla>			
Campo	**Descripción**	**Tipo**	**Es Nulo**
< Nombre del Campo>	*<Descripción del campo>*	*< Numérico, Char, Date, Lógico>*	*< SI,*NO>
Clave primaria: *<Especificación del campo que será clave primaria>* *Clave de unicidad*: *<Especificación del campo que será clave de unicidad>*			

5. Arquitectura de la Aplicación

En este punto hay que incluir un prototipo de arquitectura, al especificar los componentes de la misma, el detalle de esos componentes y las relaciones entre ellos. Incluyendo las partes más relevantes y / o críticas del sistema. Hay que tomar en cuenta las siguientes consideraciones:

- *Validar para qué servidor de aplicaciones y base de datos se está desarrollando el sistema*
- *Tratar lo más que se pueda de no depender de los tipos de datos, componentes o librerías de dichas plataformas. De hecho, sólo por requerimientos de proceso se deberían utilizar estos elementos.*
- *Usar Opensource. Frameworks y herramientas de desarrollo opensource facilitan el trabajo de desarrollo.*
- *Separar siempre las arquitecturas en N-Capas:*
  - *Presentación*
  - *Negocio*
  - *Persistencia*
  - *Infraestructura*
  - *Opcional: Integración*
- *Utilizar Patrones de diseño, mejores prácticas y blueprints. Tanto las plataformas .NET como Java EE tienen mejores prácticas y blueprints publicados para mejorar características tales como facilidad de desarrollo, desempeño, usabilidad y reutilización de código*

Uno de los patrones de diseño más utilizado para cualquier tipo aplicaciones es el de Capas donde, se divide los elementos de diseño en paquetes de Interfaz de Usuario, Lógica de Negocio y Acceso a Datos y Servicios.

La definición de la arquitectura necesita la definición de las vistas conceptual, lógica, física, de implementación

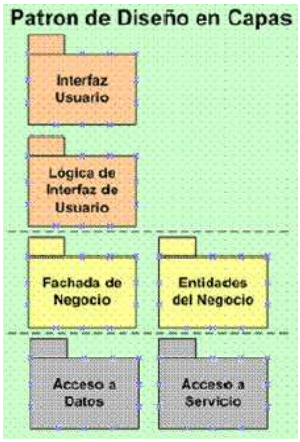


Figura 48. Diagrama de Paquetes, organización en capas

5.1. Vista Conceptual

En este punto hay que definir definir los requerimientos funcionales y la visión que los usuarios del negocio tienen de la aplicación y describir el modelo de negocio que la arquitectura debe cubrir. Esta vista muestra los subsistemas y módulos en los que se divide la aplicación y la funcionalidad que brinda dentro de cada uno de ellos. Para lo cual puede utilizarse: Casos de Uso, Diagramas de Actividad, Procesos de Negocio, Entidades del Negocio, etc.

Por ejemplo:

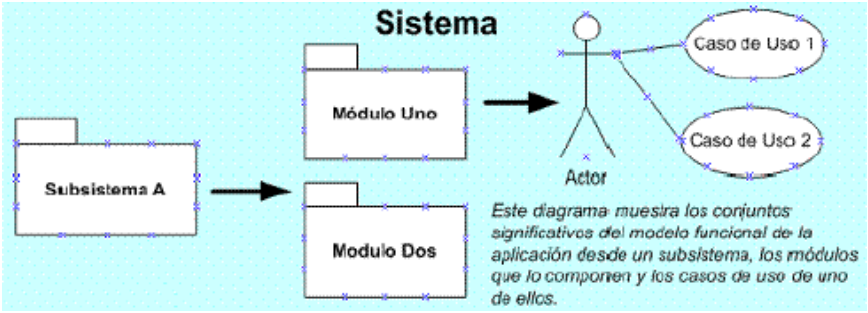


Figura 49. Vista Conceptual (de propuestas de Arquitectura, Universidad Antioquia)

5.2. Vista Lógica

Esta vista muestra los componentes principales de diseño y sus relaciones de forma independiente de los detalles técnicos y de cómo la funcionalidad será implementada en la plataforma de ejecución. Los arquitectos crean modelos de diseño de la aplicación, los cuales son vistas lógicas del modelo funcional y que describen la solución. Se pueden utilizar Realización de los Casos de Uso, subsistemas, paquetes y clases de los casos de uso más significativos arquitectónicamente.

Por ejemplo:

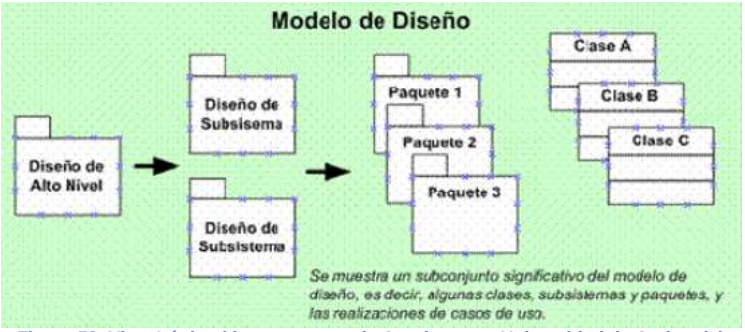


Figura 50. Vista Lógica (de propuestas de Arquitectura, Universidad de Antioquia)

5.3. Vista Física

En este punto se ilustra la distribución del procesamiento entre los distintos equipos que conforman la solución, incluyendo los servicios y procesos de base. Los elementos definidos en la vista lógica se "mapean" a componentes de software (servicios, procesos, etc.) o de hardware que definen más precisamente como se ejecutará la solución. Un ejemplo de esta vista se observa en la figura

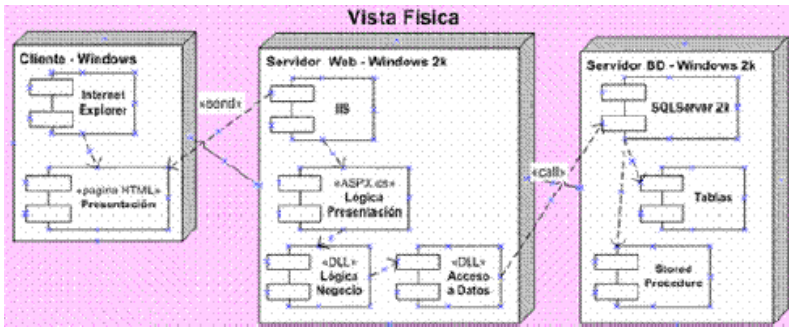


Figura 51. Diagrama de Despliegue

5.3. Vista de Implementación

Describe cómo se implementan los componentes físicos mostrados en vista de distribución agrupándolos en subsistemas organizados en capas y jerarquías, ilustra, además las dependencias entre éstos. Básicamente, se describe el mapeo desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos.

Un ejemplo de esta vista:

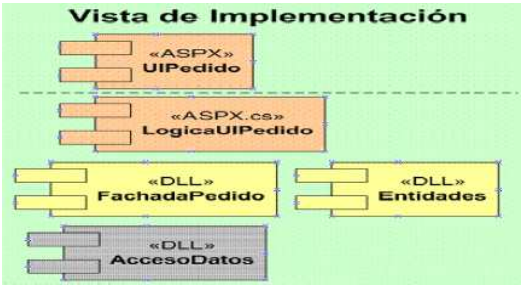


Figura 52. Diagrama de Componentes

6. Casos de Prueba de Aceptación

En esta sección especifique los casos de prueba de aceptación que han sido planteados para el sistema. Cada prueba es especificada mediante un documento que establece:

- *las condiciones de ejecución,*
- *las entradas de la prueba,*
- *los resultados esperados.*

Estos casos de prueba son aplicados como pruebas de regresión en cada iteración. Cada caso de prueba llevará asociado un procedimiento de prueba con las instrucciones para realizar la prueba, y dependiendo del tipo de prueba dicho procedimiento podrá ser automatizable mediante un script de prueba.

7. Definiciones de Calidad

En esta sección incluya una descripción de los parámetros a tener en cuenta para llevar un control de calidad, los cuales pueden ser todos o algunos de los siguientes:

Funcionalidad

a. adecuación a las necesidades
b. precisión de los resultados
c. interoperabilidad
d. seguridad de los datos

Confiabilidad

a. madurez
b. tolerancia a faltas
c. recuperabilidad (Ver si aplica)

Usabilidad

a. comprensible
b. aprendible
c. operable
d. atractivo

Eficiencia

a. comportamiento respecto al tiempo (Ver si aplica)
b. utilización de recursos

Mantenibilidad

a. analizable
b. modificable
c. estable, no hay efectos inesperados luego de modificaciones
d. verificable

Portabilidad

a. adaptable (Ver si aplica)
b. instalable
c. co-existencia
d. reemplazante (Ver si aplica)

8. Aspectos de Riesgos y Seguridad

Este punto incluye una lista de los riesgos conocidos y vigentes en el proyecto, ordenados en orden de importancia decreciente y con acciones específicas de contingencia o para su mitigación.

ANEXO X. MODELO DE IMPLEMENTACION Y PRUEBAS

Fundada en 1867
UNIVERSIDAD DE CUENCA

DEPARTAMENTO DE DESARROLLO INFORMATICO

MODELO DE IMPLEMENTACION Y PRUEBAS

Versión:	[Número]
Fecha creación:	[día] de [mes] de [año]
Última actualización:	[día] de [mes] de [año]
Estado del Documento:	[Propuesta/Borrador/Validado/Aprobado]
Cliente:	[Nombre de la institución para la cual se desarrolla el proyecto]
Elaborado por:	[Nombre de la persona que elaboró el documento]
Revisado por:	[Nombre de la persona que revisó el documento]

Registro de Cambios del Documento

Liste en la matriz los cambios que se han llevado a cabo sobre el documento de "Modelo de Implementación y Pruebas", detallando cuándo se realizaron, quién modificó el documento, sobre qué versión se trabajó y el detalle de los cambios efectuados.

Fecha	Autor	Versión	Estado	Cambios realizados

* Estado se refiere al estado del documento y puede ser: Borrador / Propuesta / Validado / Aprobado.

Revisores

Liste en la matriz los nombres de las personas encargadas de revisar el documento de "Modelo de Implementación y Pruebas", indicando la información solicitada.

Nombre	Versión Aprobada	Cargo/Rol en el proyecto	Fecha

Lista de Distribución

Proporcione una lista de las personas a quienes se distribuye el documento y recolecte las firmas respectivas como muestra de su aceptación.

Nombre	Cargo	Firma de Aceptación

Tabla de Contenidos

1. Introducción

1.1. Visión General del Documento

El documento de Modelo de Implementación y Pruebas tiene por objeto disponer de un documento en donde se integre formalmente la documentación referente a la implementación del sistema y las pruebas realizadas sobre este para el proyecto: [Nombre del Proyecto], en donde se debe incluir el esquema de interfaces, la definición de las unidades de prueba, los pseudocódigos o código fuente que contiene los casos de prueba Este documento deberá ser generado por el modelador, el desarrollador y personal involucrado en la especificación y construcción del sistema.

1.2 Audiencia

La audiencia hacia la cual está dirigido el documento de Modelo de Implementación y Pruebas, incluye:

- [Miembro de la audiencia]
- [Miembro de la audiencia]

Añada viñetas como personas/grupos pertenezcan a la audiencia del documento de Visión de Proyecto.

1.3 Definiciones, Siglas y Abreviaturas

[***Item**: descripción*]

[***Item**: descripción*]

1.4. Referencias

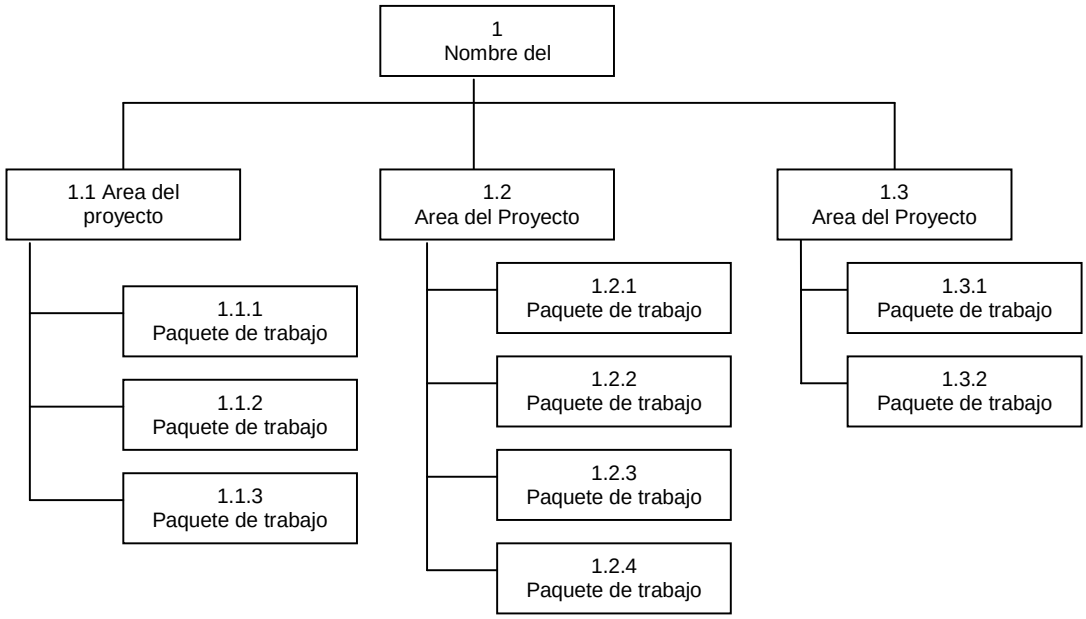
- [referencia]
- [referencia]

2. Prototipos de Interfaces

En este punto se incluye los esquemas de las principales pantallas a implementar, se puede incluir el esquema de los principales menús y submenús propuestos u otras interfaces importantes para el sistema

3. Modularización del Proyecto

Este punto contiene una descripción jerárquica del trabajo que debe ser realizado, llegando a identificar en su último nivel los "paquetes de trabajo", llegando a un esquema de descomposición similar al de la siguiente figura:



Para definir los paquetes de trabajo hay que considerar:

- El estado y finalización es medible.

- El inicio y final están definidos

- Se conocen los entregables.

- Se pueden estimar costos y tiempos.

- La duración entre límites es aceptable.

- Debe existir Independencia entre actividades.

4. Casos de Prueba de Aceptación

Para partes críticas se escriben extractos de código en el lenguaje de programación con sólo la producción de código suficiente para satisfacer las pruebas, que ayuden a analizar y diseñar más detalladamente la unidad, así que en el momento de codificarla tenemos una idea mucho más clara y precisa de lo que debe hacer, constituye parte del desarrollo dirigido por pruebas o TDD empleado en los métodos ágiles

El formato en el que se documentarán los casos de prueba es el siguiente:

ID	*<Código del caso de prueba>*
Descripción	*< Breve descripción del desarrollo y objetivos del caso de prueba >*
Condiciones de Ejecución	*< Descripción de las condiciones de ejecución que se deben cumplir antes de iniciar el caso de prueba, por ejemplo, que se haya realizado correctamente el login en el sistema >*
Instrucciones	*< Descripción paso a paso de la ejecución del caso de prueba >*
Resultados esperados	*< Descripción del resultado que se esperaba de la prueba de test en la aplicación >*
Evaluación de la Prueba	*< Estado del caso de prueba, que puede ser por ejemplo: propuesta, pendiente de evaluación, realizada y satisfactoria, etc >*

5. Plan de Pruebas

El presente plan de pruebas contiene la descripción de la ejecución de los casos de prueba definidos con el fin de validar y verificar que el desarrollo cumple con los requisitos funcionales

5.1. Requerimientos y Priorización

Se trata de definir y priorizar los requerimientos que deben ser probados y hacer un cronograma para efectuar el plan de pruebas

5.2 Equipo de pruebas

Definir el personal del equipo que intervendrá en las pruebas, quienes poseerán la capacidad necesaria para determinar el éxito/fallo de la prueba, aceptar la transformación en caso de éxito y asumir y definir los plazos de reparación en caso de fallo

5.3 Control de Calidad de pantallas

Al finalizar cada pantalla o interface hay que proceder a la revisión de la misma en base a las siguientes plantillas que ya han venido utilizándose en el Departamento de Desarrollo Informático:

B) Plantilla proporcionada para revisión por parte del programador

SISTEMA < *Nombre* > - **PROGRAMADOR** < *Nombre* >

1. Nombre de la interfaz _____

2. Programador _____

3. Fechas de Elaboración
Fecha de Inicio _____ Fecha Final _____ Fecha de Prueba _____

4.	**El título de la ventana es claro y conciso?**	SI ☐ NO ☐
5.	**Se diferencian y están validados los campos obligatorios?**	SI ☐ NO ☐
6.	**La navegación entre los controles de la pantalla cumple con la secuencia correcta?**	SI ☐ NO ☐
7.	**El número de caracteres ingresados en un cuadro de texto concuerda con la longitud del campo de la base de datos?**	SI ☐ NO ☐
8.	**El tamaño de los controles concuerda con la longitud del campo?**	SI ☐ NO ☐
9.	**Cada uno de los controles presenta un mensaje de ayuda para el usuario? (tooltips)**	SI ☐ NO ☐
10.	**Los enlaces de la página funcionan correctamente?**	SI ☐ NO ☐
11.	**La pantalla está ligada al menú de la aplicación?**	SI ☐ NO ☐
12.	**Se cumple con estándares de programación**	SI ☐ NO ☐
13.	**Existe documentación del código fuente y cumple con los estándares?**	SI ☐ NO ☐
14.	**Existe paginación con el número adecuado de registros?**	SI ☐ NO ☐
15.	**Existen errores gramaticales, ortográficos y/o tipográficos?**	SI ☐ NO ☐

16. Funcionalidad de la página

Realiza inserción de datos?	SI ☐	NO ☐	No existe esta función ☐
Realiza modificación de datos?	SI ☐	NO ☐	No existe esta función ☐
Realiza eliminación de datos?	SI ☐	NO ☐	No existe esta función ☐
Se ejecuta el reporte correspondiente?	SI ☐	NO ☐	No existe esta función ☐
La impresión funcionó correctamente?	SI ☐	NO ☐	No existe esta función ☐

17. Funciona en diferentes navegadores? Internet Explorer ☐ Mozilla Firefox ☐

18. Comentarios del programador y del coordinador

_____ Firma del Programador

_____ Nombre y Firma del Coordinador

B) Plantilla proporcionada para revisión por parte del usuario final

SISTEMA < *Nombre* > - **USUARIO** < *Nombre* >

1. **Nombre del usuario:** _____
2. **Nombre de la opción:** _____
3. **Fecha de Prueba:** _____
4. **El título de la ventana es claro y preciso para usted?** SI ☐ NO ☐
5. **El acceso a la página y tiempo de respuesta es óptimo?** SI ☐ NO ☐
6. **Se distingue claramente cuales son los campos obligatorios?** SI ☐ NO ☐
7. **Está claro el texto alternativo presentado en cada uno de los campos?** SI ☐ NO ☐
8. **Si presenta gráficas y/o tablas que contengan datos estadísticos, se encuentran claramente rotuladas y son fáciles de comprender** SI ☐ NO ☐
9. **La información que se presenta en la pantalla está libre de errores gramaticales, ortográficos y tipográficos?** SI ☐ NO ☐
10. **La página es visualmente agradable y fácil de usar?** SI ☐ NO ☐
11. **La navegación entre los elementos de la aplicación se realiza con facilidad, sin perderse o confundirse?** SI ☐ NO ☐
12. **Los títulos en los campos y los mensajes que presenta la aplicación están claros y precisos para usted?** SI ☐ NO ☐
13. **En caso de realizar una búsqueda los resultados son los esperados?** SI ☐ NO ☐
14. **Funcionalidad de la página**
    - Realiza inserción de datos? SI ☐ NO ☐ No existe esta función ☐
    - Realiza modificación de datos? SI ☐ NO ☐ No existe esta función ☐
    - Realiza eliminación de datos? SI ☐ NO ☐ No existe esta función ☐
    - Se ejecuta el reporte correspondiente? SI ☐ NO ☐ No existe esta función ☐
    - La impresión funcionó correctamente? SI ☐ NO ☐ No existe esta función ☐
15. **La página funciona correctamente en su navegador?** SI ☐ NO ☐

    Indique cuál es su navegador _____
16. **Escriba aquí las correcciones que se deben hacer a la página**

    _____
17. **Recomendaciones para mejorar esta página**

_____

Firma del Usuario

5.4 Reporte de pruebas

Cada una de las distintas ejecuciones del plan de pruebas será reflejada en un documento cuya organización se hará conforme a la siguiente plantilla de reporte de pruebas:

CASO DE PRUEBA:	(Código de la prueba)
FECHA:	
RESULTADO:	(Correcto o No correcto)
OBSERVACIONES:	
RESPONSABLES DE LA PRUEBA:	(Nombre y Cargo)
Firma:	

5.5 Reporte de Fallas y Sugerencias

Este punto incluye una lista de fallas encontradas en las diferentes revisiones y las sugerencias que se proporcionan por personal del equipo de pruebas

6. MODELO DE SERVICIOS

Este punto incluye una descripción de los contratos definidos para los servicios desarrollados, el detalle se puede revisar en el Anexo XVI. Modelo de Servicios

ANEXO XI. PLAN DE LIBERACION

DEPARTAMENTO DE DESARROLLO INFORMATICO

PLAN DE LIBERACION

Versión:	[Número]
Fecha creación:	[día] de [mes] de [año]
Última actualización:	[día] de [mes] de [año]
Estado del Documento:	[Propuesta/Borrador/Validado/Aprobado]
Cliente:	[Nombre de la institución para la cual se desarrolla el proyecto]
Elaborado por:	[Nombre de la persona que elaboró el documento]
Revisado por:	[Nombre de la persona que revisó el documento]

Registro de Cambios del Documento

Liste en la matriz los cambios que se han llevado a cabo sobre el documento de "Plan de Liberación", detallando cuándo se realizaron, quién modificó el documento, sobre qué versión se trabajó y el detalle de los cambios efectuados.

Fecha	Autor	Versión	Estado	Cambios realizados

* Estado se refiere al estado del documento y puede ser: Borrador / Propuesta / Validado / Aprobado.

Revisores

Liste en la matriz los nombres de las personas encargadas de revisar el documento de "Plan de Liberación", indicando la información solicitada.

Nombre	Versión Aprobada	Cargo/Rol en el proyecto	Fecha

Lista de Distribución

Proporcione una lista de las personas a quienes se distribuye el documento y recolecte las firmas respectivas como muestra de su aceptación.

Nombre	Cargo	Firma de Aceptación

Tabla de Contenidos

1. Introducción

1.1. Visión General del Documento

El documento *Plan de Liberación* tiene por objeto disponer de un documento en donde se integre formalmente la documentación referente a la implementación del sistema y las pruebas realizadas sobre este para el proyecto: [Nombre del Proyecto]. En este plan se describe las actividades que se deben realizar para llevar el producto al ambiente del cliente. Las actividades incluyen planificación, verificación de versión “Beta”, preparar los elementos que serán distribuidos al cliente, forma de envío, instalación, capacitación y soporte. Este documento deberá ser generado por director del departamento, el coordinador del proyecto, el desarrollador y el documentador.

1.2 Audiencia

La audiencia hacia la cual está dirigido el documento de Modelo de Implementación y Pruebas, incluye:

- [Miembro de la audiencia]
- [Miembro de la audiencia]

Añada viñetas como personas/grupos pertenezcan a la audiencia del documento de Visión de Proyecto.

1.3 Definiciones, Siglas y Abreviaturas

[***Item**: descripción*]
[***Item**: descripción*]

1.4. Referencias

- [referencia]
- [referencia]

2. Actividades de implantación

En esta sección se hace una enumeración de las actividades que serán efectuadas para implantar el proyecto, se puede tomar como guía las siguientes actividades:

- *Planificar la Implantación*
- *Desarrollar el material para soporte al usuario*
- *Desarrollar el material para capacitación*
- *Realizar la presentación del sistema al cliente y realizar la capacitación a los usuarios del sistema*
- *Verificación en ambiente de desarrollo*
- *Producir la versión "Beta" del producto*
- *Verificación en ambiente del usuario*
- *Producir la versión "Final" del producto*
- *Hacer que el producto esté accesible por Internet, instalado en el entorno del usuario o empaquetado para la venta; según sea la modalidad de distribución del mismo >*

3. Cronograma

En esta sección se procede a describir el cronograma y los hitos para conducir las actividades de la implantación en el ambiente de usuario. Indique actividades, responsables y fechas de realización de las mismas.

4. Recursos

En esta sección registre la lista de recursos y sus orígenes, requeridos para llevar a cabo las actividades planificadas para la implantación; describa las facilidades requeridas para verificar y liberar el software, las facilidades pueden incluir aulas, salas o edificios especiales, requerimientos de energía eléctrica, además se debe identificar el hardware requerido para ejecutar y apoyar al software que será liberado. Si es aplicable, describir todo el software de apoyo necesario para el producto que se va a liberar, como herramientas, compiladores, herramientas de verificación, herramientas de Gestión de Configuración, bases de datos, archivos de datos, etc. Incluir también una descripción del personal, y sus habilidades, requerido para apoyar el producto que se va a liberar.

5. Documentos de apoyo

En esta sección se describe la documentación de apoyo necesaria para el producto que se va a liberar, incluyendo manuales del sistema, descripciones de diseño, casos y procedimientos de prueba, manuales de usuario, presentaciones y material de entrenamiento.

6 Notas de la Versión

Esta sección contiene las notas de entrega para la versión X.Y.Z. del PRODUCTO

CUANDO EL NÚMERO X EN LA VERSIÓN CAMBIA) Esta es una entrega mayor con muchas nuevas características. Los usuarios de entregas anteriores deberían de revisar la sección de "Compatibilidad de Versión" más abajo por instrucciones en cómo usar la información existente con esta nueva versión.

(CUANDO EL NÚMERO Y EN LA VERSIÓN CAMBIA) Esta es una actualización con algunas correcciones importantes. Los usuarios de entregas anteriores deberían de actualizar su sistema.

(CUANDO EL NÚMERO Z EN LA VERSIÓN CAMBIA) Esta es una entrega de mantenimiento que mejora la calidad, confiabilidad y desempeño sin añadir ninguna nueva funcionalidad. Todos los usuarios de las entregas X.Y anteriores deberían actualizarse a esta nueva entrega.

7 Capacitación

Describir el plan para capacitar a los usuarios finales para que puedan usar y adaptarse al producto como se requiera

ANEXO XII. MODELO DE CONFIGURACION Y CALIDAD

Fundada en 1867
UNIVERSIDAD DE CUENCA

DEPARTAMENTO DE DESARROLLO INFORMATICO

MODELO DE CONFIGURACION Y CALIDAD

Versión:	[Número]
Fecha creación:	[día] de [mes] de [año]
Última actualización:	[día] de [mes] de [año]
Estado del Documento:	[Propuesta/Borrador/Validado/Aprobado]
Cliente:	[Nombre de la institución para la cual se desarrolla el proyecto]
Elaborado por:	[Nombre de la persona que elaboró el documento]
Revisado por:	[Nombre de la persona que revisó el documento]

Registro de Cambios del Documento

Liste en la matriz los cambios que se han llevado a cabo sobre el documento "Modelo de Configuración y Calidad", detallando cuándo se realizaron, quién modificó el documento, sobre qué versión se trabajó y el detalle de los cambios efectuados.

Fecha	Autor	Versión	Estado	Cambios realizados

* Estado se refiere al estado del documento y puede ser: Borrador / Propuesta / Validado / Aprobado.

Revisores

Liste en la matriz los nombres de las personas encargadas de revisar el documento de "Modelo de Configuración y Calidad", indicando la información solicitada.

Nombre	Versión Aprobada	Cargo/Rol en el proyecto	Fecha

Lista de Distribución

Proporcione una lista de las personas a quienes se distribuye el documento y recolecte las firmas respectivas como muestra de su aceptación.

Nombre	Cargo	Firma de Aceptación

Tabla de Contenidos

1. Introducción

1.1. Visión General del Documento

El documento *Modelo de Configuración y Calidad* tiene por objeto disponer de un documento en donde se integre formalmente la documentación referente a la implementación del sistema y las pruebas realizadas sobre este para el proyecto [Nombre del Proyecto]. En este plan se describe las actividades que se deben realizar para llevar el producto al ambiente del cliente. Las actividades incluyen planificación, verificación de versión “Beta”, preparar los elementos que serán distribuidos al cliente, forma de envío, instalación, capacitación y soporte.

Además en este documento se debe especificar la porción del ciclo de vida del software que será cubierta por el Plan de Calidad.

Este documento deberá ser generado por director del departamento, el coordinador del proyecto, el desarrollador y el documentador.

1.2 Audiencia

La audiencia hacia la cual está dirigido el documento de *Modelo de Configuración y Calidad* incluye:

- [Miembro de la audiencia]
- [Miembro de la audiencia]

Añada viñetas como personas/grupos pertenezcan a la audiencia del documento de Visión de Proyecto.

1.3 Definiciones, Siglas y Abreviaturas

[***Item**: descripción*]

[***Item**: descripción*]

1.4. Referencias

- [referencia]
- [referencia]

2. Selección de Items de Configuración

En este punto se procede a seleccionar los Items de configuración y los productos de trabajo que los componen, los mismos que pueden ser: documentación de procesos, requisitos de software, diseños de productos de software, código fuente y ejecutables, procedimientos de prueba, Descripciones de Interfaces, Compiladores, productos adquiridos, herramientas, descripciones de procesos, descripción de requisitos, productos de diseño, planes de proyecto y otros ítems utilizados para crear y describir los productos de trabajo. Para cada Item de Configuración descubierto hay que definir:

Identificador	***Nombre***	***Características***	***Control de Configuración***	***Responsable***
Asignar un identificador único	*Nombre identificativo del item*	*Especificar las características de cada elemento de configuración que aportan detalles para identificar más claramente al elemento, como autor, tipo de archivo, lenguaje de programación (en caso de ser código fuente), etc.*	*Especificar cuando el Item se pone bajo control de la Administración de la Configuración:* *a. Nombre* *b. Etapa del ciclo de vida* *c. Versiones* *d. Estado* *e. Localización*	*Especificar el dueño responsable del Elemento de Configuración.*

Para cada ítem se propone la siguiente nomenclatura que corresponde a la disciplina:

Modelo: (MD)

Nomenclatura	**Entregable**
MDDRQ	Especificación de Requerimientos
MDMOD	Modelo de Casos de Uso
MDGLO	Glosario
MDMDO	Modelo de Dominio

Implementación: (IM)

Nomenclatura	**Entregable**
IMEI	Estándar de Implementación
IMPR	Prototipo
IMOORRP	Reporte de Revisión por Pares
IMOOMIM	Modelo de Implementación

Pruebas: (PR)

Nomenclatura	**Entregable**
PRPVV	Plan de Verificación y Validación
PRMCP	Modelo de Casos de Prueba
PRIVS	Informe de Verificación del Sistema
PRRPR	Reportes de Pruebas

Liberación: (LI)

Nomenclatura	Entregable
LIMSU	Materiales para Soporte al Usuario
LIMCA	Materiales para Capacitación
LIPLA	Plan de Implantación
LIVPR	Versión del Producto

Gestión de Configuración y Control de Cambios (SCM):

Nomenclatura	Entregable
SCMPLA	Plan de Configuración
SCMRV	Registro de Versiones
SCMILB	Informe de la Línea Base del Proyecto
SCMIF	Informe Final de SCM

Gestión de Proyecto (GP):

Nomenclatura	Entregable
GPPLA	Plan de Proyecto
GPDRI	Documento de Riesgos
GPARE	Acta de la Reunión de Equipo
GPPIT	Plan de la Iteración

3. Línea Base

Una línea base de configuración es la configuración de un producto o sistema establecida en un momento concreto en el tiempo, que capta la estructura y los detalles de una configuración, es una referencia para más actividades, para proporcionar la base para una auditoria y regresión después de un cambio.

Se debe generar una línea base por iteración o en cada fase, de acuerdo a lo siguiente:

- Citar los eventos que dan origen a la línea base.
- Enlistar los elementos que serán controlados en la línea base.
- Procedimientos usados para establecer y cambiar la línea base.
- La autorización requerida para aprobar cambios a los documentos de la línea base

4. Requerimiento de cambio

En esta sección se incluye un requerimiento para cambios en un ítem de configuración, que deberá ser recibido en el siguiente formulario propuesto:

Fundada en 1867
UNIVERSIDAD DE CUENCA

FORMULARIO REQUERIMIENTO DE CAMBIO

FECHA	
NOMBRE DEL SOLICITANTE	
DEPARTAMENTO DEL SOLICITANTE	
DESCRIPCIÓN DEL CAMBIO	
SISTEMA AFECTADO	
RAZONES DEL CAMBIO	
EVALUACION DEL REQUERIMIENTO	
IMPLICACIONES DE NO REALIZAR EL CAMBIO	

PERSONAL DE INFORMATICA

CAMBIOS REALIZADOS		
SISTEMAS ALTERADOS		
HORAS NECESARIAS	**DISEÑO**	
	IMPLEMENTACIÓN	
TABLAS Y CAMPOS AFECTADOS		
NUEVAS VERSIONES REALIZADAS		

----------	----------	----------
Solicitante del cambio	Lider del proyecto	Informático

< Los siguientes ítems deben ser considerados al evaluar un requerimiento de cambio: >

Tipo de cambio	• *¿El cambio arregla un defecto en implementación y no afecta requerimientos* • *¿El cambio modifica de alguna forma los requerimientos existentes?* • *¿El cambio es un requerimiento de mejora?*
Tamaño del cambio	• *¿Qué tanto de trabajo existente debería ser cambiado?* • *¿Qué tanto de nuevo trabajo se necesitaría hacer?*
Alternativas	• *¿Existe alguna otra alternativa?*
Complejidad	• *¿El cambio propuesto es fácil de realizar?* • *¿Cuáles son las ramificaciones posibles al hacer este cambio?*
Severidad	• *¿Cuál es el impacto de no implementar este requerimiento?* o *¿Existe alguna pérdida de trabajo o datos?* o *¿Es un requerimiento de mejora?* o *¿Es una incomodidad menor?*
Cronograma	• *¿Cuándo se requiere el cambio?* • *¿Qué tan asequible es?*
Impacto	• *¿Cuáles son las consecuencias de hacer los cambios?* • *¿Cuáles son las consecuencias de no hacer los cambios?*
Costo	• *¿Cuál es el costo ahorrado al hacer este cambio?*
Relación con los otros cambios	• *¿Qué otros cambios reemplazan o invalidan a este?* • *¿Qué tanto dependen de este otros cambios?*
Pruebas	• *¿Existe alguna prueba especial que necesite hacer para verificar que el cambio sea un éxito?*

5. Auditoría de Configuración

Se debe validar que el producto este completo y así mantener la consistencia entre los componentes, asegurando que estén en un estado apropiado a través de todo el ciclo de vida del producto y que el mismo sea una colección bien definida de componentes. Cada registro de auditoría debe resultar en un registro especificado con el siguiente formato.

Nomenclatura Item:_____ Grupo:_____

Fecha	Número	Tipo	Introducido	Eliminado	Tiempo elim.	Arreg. def.	Detectado por

Descripción:_____

Fecha	Número	Tipo	Introducido	Eliminado	Tiempo elim.	Arreg. def.	Detectado por

Descripción:_____

6. Plan de Calidad

6.1. Definir el plan

Ciclo de vida del software cubierto

Esta sección debe detallar las etapas más importantes del ciclo de vida del software que cubre el Plan. Y debe contener una lista con todos los productos de proyecto que tendrán revisiones de calidad.

Actividades a realizarse

Las tareas a ser llevadas a cabo deberán reflejar las evaluaciones a realizar, los estándares a seguir, los productos a revisar, los procedimientos a seguir en la elaboración de los distintos productos y los procedimientos para informar de los defectos detectados a sus responsables y realizar el seguimiento de los mismos hasta su corrección.

Las actividades a realizar son detalladas en el siguiente esquema:

Actividad	Observaciones	Revisor	Fecha
Actividad 1	Actividad 1	[Nombre]	[dd/mm/aaaa]

Factores y Métricas para determinar los factores de calidad

Es fundamental definir el atributo de calidad de software que pretendemos estudiar, que debe ser de acuerdo a las necesidades prioritarias del Departamento. Una vez seleccionado el atributo de calidad es necesario verificar su correspondiente métrica.

Los factores a considerar son:

1. *Características Operativas*
– *Corrección* ¿Hace lo que se le pide?
– *Fiabilidad* ¿Lo hace de forma fiable todo el tiempo?
– *Eficiencia* ¿Qué recursos hardware y software necesito?
– *Integridad* ¿Puedo controlar su uso?
– *Facilidad de uso* ¿Es fácil y cómodo de manejar?

2. *Revisión del producto:* capacidad para soportar cambios
– *Facilidad de mantenimiento* ¿Puedo localizar los fallos?
– *Flexibilidad* ¿Puedo añadir nuevas opciones?
– *Facilidad de prueba* ¿Puedo probar todas las opciones?

3. *Transición del producto: adaptabilidad a nuevos entornos*
– *Portabilidad* ¿Podré usarlo en otra máquina?
– *Reusabilidad* ¿Podré utilizar alguna parte del software en otra aplicación?
–*Interoperabilidad* ¿Podrá comunicarse con otras aplicaciones?

Las métricas para determinar los factores de calidad son:

– Exactitud
– Normalización de las comunicaciones
– Completitud
– Concisión
– Consistencia
– Estandarización de los datos
– Tolerancia de errores
– Eficiencia de la ejecución
– Facilidad de expansión
– Generalidad
– Independencia del hardware
– Instrumentación
– Modularidad
– Facilidad de operación
– Seguridad
– Documentación
– Simplicidad
– Independencia del sistema software
– Facilidad de traza
– Formación

6.2 Realización de actividades

Se procede a la ejecución de las actividades planificadas, las mismas que pueden ser:

- Revisar cada producto.- Se revisan los productos que se definieron como claves. Se revisan los productos contra los estándares, utilizando la checklist definida para el producto.
- Revisar el ajuste al proceso.- Se debe recoger la información necesaria de cada producto, buscando hacia atrás los productos previos que deberían haberse generado, para poder establecer los criterios de revisión y evaluar si el producto cumple con las especificaciones.
- *Realizar Revisión Técnica Formal (RTF).-* Es un proceso de revisión riguroso, para llegar a detectar lo antes posible defectos o desviaciones en los productos que se van generando a lo largo del desarrollo. Esta característica se adopta para productos de especial importancia.
- *Documentar desviaciones.-* Las desviaciones encontradas en las actividades y en los productos deben ser documentadas

Si se desea ser más exactos se puede proceder con la medición del software derivando un valor numérico para algún atributo de un producto de software, para lo cual se procede a:

a. Seleccionar los componentes a evaluar.
b. Medir las características de los componentes con las métricas de software.
c. Identificar las mediciones anómalas.
d. Analizar los componentes anómalos.

6.3 Revisión de Requisitos de Software.

Al finalizar cada módulo de software, antes de proceder a la etapa de pruebas, es importante contrastar si los requisitos que inicialmente fueron detallados han sido solucionados en su totalidad.

La evaluación de cumplimiento de los requisitos de software será registrado en la siguiente tabla:

Módulo	**Requisito**	**Actividad**	**Detalle/ Indicación**	**Responsable**	**Fecha**
[Detalle del Módulo o Proceso del Sistema]	[Requisito Evaluado]	[Tarea realizada para la revisión]	[Detalle /Indicación del Resultado de la Evaluación]	[Nombre]	[dd/mm /aaaa]

6.4. Revisión de requisitos implícitos

Existen ciertos requisitos o aspectos que no pueden determinarse cuantitativamente o que en el transcurso del proyecto se han notado ser importantes o para la alta gerencia o para los usuarios tradicionales, es importante que esos requisitos sean explicitados y evaluados a media que se finaliza un módulo o la parte del proyecto en donde los requisitos implícitos pueden originar reacciones

La evaluación de estos requisitos quedará citada en la siguiente tabla:

Módulo	**Requisito**	**Actividad**	**Detalle/ Indicación**	**Responsable**	**Fecha**
[Detalle del Módulo o Proceso del Sistema]	[Requisito Evaluado]	[Tarea realizada para la revisión]	[Detalle / Indicación del Resultado de la Evaluación]	[Nombre]	[dd/mm/ aaaa]

.

ANEXO XIII. PLAN DEL PROYECTO

Fundada en 1867
UNIVERSIDAD DE CUENCA

DEPARTAMENTO DE DESARROLLO INFORMATICO

PLAN DEL PROYECTO

Versión:	[Número]
Fecha creación:	[día] de [mes] de [año]
Última actualización:	[día] de [mes] de [año]
Estado del Documento:	[Propuesta/Borrador/Validado/Aprobado]
Cliente:	[Nombre de la institución para la cual se desarrolla el proyecto]
Elaborado por:	[Nombre de la persona que elaboró el documento]
Revisado por:	[Nombre de la persona que revisó el documento]

Registro de Cambios del Documento

Liste en la matriz los cambios que se han llevado a cabo sobre el documento "Plan del Proyecto", detallando cuándo se realizaron, quién modificó el documento, sobre qué versión se trabajó y el detalle de los cambios efectuados.

Fecha	Autor	Versión	Estado	Cambios realizados

* Estado se refiere al estado del documento y puede ser: Borrador / Propuesta / Validado / Aprobado.

Revisores

Liste en la matriz los nombres de las personas encargadas de revisar el documento de "Plan del Proyecto", indicando la información solicitada.

Nombre	Versión Aprobada	Cargo/Rol en el proyecto	Fecha

Lista de Distribución

Proporcione una lista de las personas a quienes se distribuye el documento y recolecte las firmas respectivas como muestra de su aceptación.

Nombre	Cargo	Firma de Aceptación

Tabla de Contenidos

1. Introducción

1.1. Visión General del Documento

El documento *Plan del Proyecto* tiene por objeto disponer de un documento que tiene la visión del proyecto completo en donde se integra formalmente la documentación referente al proyecto: [Nombre del Proyecto]. En este plan se incluye el propósito, alcance, definiciones, acrónimos, abreviaturas y referencias de este proyecto

1.2 Audiencia

La audiencia hacia la cual está dirigido el documento de *Plan del Proyecto*, incluye:

- [Miembro de la audiencia]
- [Miembro de la audiencia]

Añada viñetas como personas/grupos pertenezcan a la audiencia del documento de Visión de Proyecto.

1.3 Definiciones, Siglas y Abreviaturas

Debe incluir las definiciones de todos los términos, acrónimos y abreviaturas necesarias para interpretar adecuadamente este plan. Se puede referenciar al glosario del proyecto

[***Item**: descripción*]
[***Item**: descripción*]

1.4 Referencias

Esta subsección incluye una lista de todos los documentos referenciados, cada uno de los cuales debería ser identificado por su título, número de informe (si se aplica), fecha y organización o fuente que lo publica.

- [referencia]
- [referencia]

2. Generalidades del Proyecto

2.1 Descripción del proyecto

Propósito, Alcance y Objetivos

<Contiene una breve descripción del propósito y alcance del proyecto, así como una breve descripción de lo que se espera desarrollar en el proyecto.>

2.2 Restricciones y Asunciones

<Contiene una lista de restricciones y asunciones para este plan, por ejemplo: presupuestos, plantilla de personal, equipamiento y horario a aplicar al proyecto. >

2.3 Artículos y Artefactos del Proyecto

<Una tabla con los productos a crear durante el proyecto, incluyendo las fechas previstas para su desarrollo.>

2.4 Evolución del plan de Desarrollo del Software

<Contiene una tabla con las versiones propuestas para este plan y los criterios para la revisión de cada plan y la generación de una nueva versión.>

3. Organización del Proyecto

3.1 Estructura Organizacional

<Describe la estructura de la organización del equipo del proyecto, incluyendo a los gestores y revisores. Haga aquí una descripción de la estructura organizativa del proyecto, apoyado en un organigrama de ser posible>

3.2 Interfaces externas o Canales de contacto

<Describe cómo interacciona el proyecto con grupos externos. Para cada grupo externo, se identifican nombres de contacto internos y externos, así como las responsabilidades de desarrollo y aceptación del producto. Incluir una línea o dos sobre son las personas designadas como canal de comunicación, y los eventos que se esperan reportar por dicho canal.>

3.3 Recursos Humanos y Profesionales

<Describe el personal que laborará en el equipo del proyecto.>

Quien	*Información de Contacto*
[Nombre completo]	*[Correo electrónico, teléfono, etc.]*

3.4 Roles y Responsabilidades

< Identifica a los responsables de cada una de las disciplinas, detalles del flujo de trabajo y procesos de soporte. Los roles son asuntos que deben estar bien definidos, cada uno de ellos ha de tener una clara definición de las tareas, actividades, artefactos y productos de trabajo que se espera que manejen, así como de los recursos que van a utilizar para llevar a cabo su trabajo.

Los roles que se han definido en la metodología RASDUC, son:

ROL	SIMBOLO	Responsabilidades	Miembros del Equipo
Administrador de la Base de Datos			
Modelador			
Administrador de la Configuración			
Revisor			
Desarrollador			
Coordinador del Proyecto			
Documentador			
Involucrado			
Director del Departamento			

Se deberá indicar el nombre de cada rol en su propia columna, indique la descripción breve del mismo en la columna de "Responsabilidades" y finalmente se indique cuales miembros del equipo van a asumir dicho rol. Es decir, que una persona puede aparecer varias veces en la tabla, en distintas líneas de rol

4. Gestión del Proyecto

< El propósito de la Gestión de Proyecto es proveer de control, guía y recursos oportunos sin generar retrasos ni burocracia excesiva que haga más difícil el trabajo técnico. Tener el control sobre algo sin perturbar a este algo es el propósito >

4.1 Estimaciones del Proyecto

<Proporciona los costes y tiempos estimados para el proyecto, así como las bases para esas estimaciones. Se indican aquí los estimados en tiempo y costo del proyecto, así como las circunstancias y momentos en que se ha de revisar este estimado, sin olvidar explicar claramente como se ha llegado a estos estimados. Haga referencia a métricas o experiencias previas que justifiquen y fundamenten lo dicho aquí. >

4.2 Plan del proyecto

4.2.1 Plan de fases

<Indique aquí las fases de RASDUC: MODELO, IMPLEMENTACION, PRUEBAS, DESPLIEGUE, además indique para cada una cual es la meta que se espera lograr y cuales condiciones se han de verificar para considerar que es posible pasar a la siguiente fase >

Fase	*No. de Iteraciones*	*Objetivos del Ciclo de Vida*	*Fecha de Inicio*	*Fecha de Finalización*

4.2.2 Objetivos por iteración

<Para cada una de las iteraciones, se debe adjuntar una lista de objetivos a conseguir>

4.2.3 Incrementos o Versiones

< Aquí se incluye una breve descripción de cada versión, indicando si es demo, beta, etc. Un incremento es una versión funcional aunque parcial del sistema. Detalle aquí cual es la perspectiva sobre este punto y actualice esta sección a medida que los vaya liberando. Se puede indicar por ejemplo, que el proyecto tiene cuatro partes, que una se entregará en la iteración E1 y que las tres restantes deben estar listas para la iteración C2. >

4.2.4 Temporización del Proyecto. Diagrama de Gannt

<Diagramas o tablas mostrando las fechas previstas para completar las iteraciones y fases, la entrega de versiones o demos y otros eventos Un diagrama de tiempos o de Gantt mostrando el tiempo asociado a las fases o iteraciones. >

4.2.5 Diagrama WBD

< Incluye la estructura de división del trabajo, que puede hacerse utilizando un esquema WBS (Work Breakdown Structure) >

4.3. Plan de Gestión por áreas

< De los temas anotados en esta sección, muchos son llevados como documentos independientes en proyectos de cierto tamaño, en cuyo caso hay que referenciarlos en esta sección, pero si las cosas son simples se puede indicar que no aplica, pues los planes de esta sección planes pueden ser considerados como ampliaciones al proceso de gestión del proyecto, siendo ser flexible incluirlos o no >

4.3.1 Plan de adquisición de recursos

< Describa aquí como se manejan las adquisiciones de bienes, servicios y recursos para el proyecto. Indicando quien es el responsable, los criterios que se utilizarán para proceder con las compras y los lapsos y tiempos en que se ha de proceder con las mismas. >

4.3.2 Plan de entrenamiento

<Lista cualquier entrenamiento especial que requieren los miembros del proyecto y las fechas en las que debe hacerse y completarse.>

4.3.2. Plan del personal

<Identificar aquí el número y tipo del personal requerido para cada fase del proyecto o iteración,

incluyendo las habilidades especiales y su experiencia.>

4.3.4 Monitorización y control del Proyecto

<En este punto se describe la técnica seguida para monitorizar el proyecto y contrastarlo con los tiempos previstos, para saber qué acciones correctivas seguir, Describe los reportes y técnicas que se van a utilizar para "pulsar" el avance del proyecto y poder medir si esta en donde debiera. Detalle cuales de estos informes van a ser compartidos con el cliente y cuales son solo para uso interno.

Puede ser algo muy simple; de hecho recomendaría que simplemente fuera la típica reunión de avance semanal o quincenal interna al equipo de desarrollo; quizás con ayuda de alguna técnica de reunión de grupos o de algún que otro documento de reporte de actividades, como pueden ser los cierres de iteración>

4.5 Plan de gestión de Riesgos

<Un plan de riesgos enumera las situaciones negativas que en potencia pueden llegar a ocurrir durante el proyecto. Generalmente en los proyectos se tiene un plan de riesgos estructurado en un propio documento, por lo que en este punto se colocará su referencia. Un Plan de Riesgo debe mantener los riegos enumerados, caracterizados por un marcador que indique si se han concretado, técnica utilizada para mitigarlo (eliminación, cobertura, diversificación, ad-hoc) impacto y probabilidad de ocurrencia.
Se acompaña el plan de riesgos con la llamada "Lista de Riesgos" que contiene la enumeración de cada uno, junto con su probabilidad de ocurrencia y magnitud del daño que causa de ocurrir.>

Lista de Riesgos para el proyecto <"nombre del proyecto">

ID	Descripción	P	I	E	Indicador	Mitigación	Propietario	Fecha de vencimiento
1	*<Liste los mayores riesgos que enfrenta el proyecto. Describa cada riesgo en la forma : condición-consecuencia.* *Ejemplo* *"El personal subcontratado no tiene suficiente experiencia técnica, por lo que su trabajo se retrasó por la capacitación y por la baja curva de aprendizaje." >*	*P	*I	*E	*<Para cada riesgo, describa el indicador o disparador que indica que el riesgo se está convirtiendo en problema>.*	*<Para cada riesgo, indique una o más formas de mitigar el riesgo.* *Aceptar el riesgo también es una opción. .>*	*<Asigne un responsable de cada acción.>*	*<Coloque una fecha hasta la cual cada acción de mitigación deberá ser completa>*

**P = Probabilidad de ocurrencia del riesgo, se expresa como un número 0.1 (poco probabilidad) and 1 (garantizada que ocurra). Alternativamente se puede estimar como Bajo, Medio o Alto.*

**I = Pérdida relativa si el riesgo se convierte en problema, expresado como número entre 1 (impacto mínimo) y 10 (catastrófico). Alternativamente se puede estimar como Bajo, Medio o Alto.*

**E = Exposición. Si fueron asignados valores numéricos a la probabilidad e impacto, entonces la Exposición= P * I. Si fueron usados los valores Bajo, Medio, Alto, la estimación de la exposición usa la siguiente tabla:*

	Impacto		
Probabilidad	*Bajo*	*Medio*	*Alto*
Bajo	Bajo	Bajo	Medio
Medio	Bajo	Medio	Alto
Alto	Medio	Alto	Alto

Una vez calculada la Exposición para cada riesgo, se debe ordenar en orden descendente de exposición. El enfoque de los esfuerzos de mitigación deben estar en los riesgos con mayor exposición.

4.6 Plan de cierre

<Describe las actividades a realizar para poder considerar el proyecto cerrado, incluyendo reasignación del personal, archivo de los materiales del proyecto, informes finales, etc.>

5. Gestión del Proceso Técnico

5.1 Método de desarrollo

< En esta sección se detalla la forma en la que se ha organizado el proyecto en términos de los artefactos y flujos de trabajo de referencia de la metodología utilizada. Si se cuenta con un Caso de Desarrollo estructurado como documento independiente, haga la referencia aquí. >

5.2. Métodos, herramientas y técnicas

<Lista las guías utilizadas para documentar el proyecto y los estándares. Se pueden referenciar las siguientes guías: Modelado del negocio, interfaz de usuario, modelado de casos de uso, diseño, programación, pruebas y estilos de manual.>

5.3 Plan de Aceptación del Producto

<Detalle aquí los pasos a seguir para lograr que el cliente de el visto bueno formal sobre algún aspecto del proyecto, haciendo énfasis en la aprobación del proyecto en sí mismo.>

6. Gestión de los Procesos de Soporte

6.1 Plan de Gestión de la Configuración

<En este contexto, entendemos por configuración a la relación entre las distintas versiones de los artefactos del proyecto. Detalle aquí como se manejará, haciendo énfasis en lo que asegura que nadie va a trabajar con un documento desfasado. Si es una tarea compleja, desarrollarlo con un Plan independiente y haga la referencia aquí. >

6.2 Plan de Evaluación

<Descripción de las técnicas usadas para realizar la evaluación. Detalle aquí como será evaluado el proyecto. Esto incluye reuniones de avance, revisiones formales e informales, auditorias, etc. Si es un punto complejo, desarrolle esto con un Plan independiente y haga la referencia aquí>

6.3 Plan de Documentación

< Desde el mismo Plan de Desarrollo de Software hasta los manuales de usuario, existe mucha documentación con la que se debe trabajar, detalle aquí como pretende manejar todas esas páginas de texto. Si es un punto complejo, desarrolle esto con un Plan independiente y haga la referencia aquí. >

6.4 Plan de Seguro de Calidad

<Descripción del proceso seguido para garantizar la calidad del producto final>

6.5 Plan de Contratación

< Indique aquí que cosas es necesario contratar a terceros y cuáles son las pautas, condiciones y lapsos que estas contrataciones han de cumplir. Si es un punto complejo, desarrolle esto con un Plan independiente y haga la referencia aquí >

7. Planes Adicionales y Anexos

Material adicional, cada uno en una sección independiente o si el asunto es más estructurado como un documento aparte al que se hace referencia aquí. Respecto a los anexos adicionales que sean necesarios, se recomienda que se los incluya cada uno al inicio de una página independiente

ANEXO XIV. MODELO DE SERVICIOS

Fundada en 1867
UNIVERSIDAD DE CUENCA

DEPARTAMENTO DE DESARROLLO INFORMATICO

MODELO DE SERVICIOS

Versión:	[Número]
Fecha creación:	[día] de [mes] de [año]
Última actualización:	[día] de [mes] de [año]
Estado del Documento:	[Propuesta/Borrador/Validado/Aprobado]
Cliente:	[Nombre de la institución para la cual se desarrolla el proyecto]
Elaborado por:	[Nombre de la persona que elaboró el documento]
Revisado por:	[Nombre de la persona que revisó el documento]

Registro de Cambios del Documento

Liste en la matriz los cambios que se han llevado a cabo sobre el documento "Modelo de Servicios", detallando cuándo se realizaron, quién modificó el documento, sobre qué versión se trabajó y el detalle de los cambios efectuados.

Fecha	Autor	Versión	Estado	Cambios realizados

* Estado se refiere al estado del documento y puede ser: Borrador / Propuesta / Validado / Aprobado.

Revisores

Liste en la matriz los nombres de las personas encargadas de revisar el documento de "Modelo de Servicios", indicando la información solicitada.

Nombre	Versión Aprobada	Cargo/Rol en el proyecto	Fecha

Lista de Distribución

Proporcione una lista de las personas a quienes se distribuye el documento y recolecte las firmas respectivas como muestra de su aceptación.

Nombre	Cargo	Firma de Aceptación

Tabla de Contenidos

1. Introducción

1.1. Visión General del Documento

El documento *Modelo de Servicios* tiene por objeto disponer de un documento que tiene la información de los servicios que forman parte del proyecto: [Nombre del Proyecto]. En este plan se incluye el catálogo de los servicios web

1.2 Audiencia

La audiencia hacia la cual está dirigido el documento de *Modelo de Servicios*, incluye:

- [Miembro de la audiencia]
- [Miembro de la audiencia]

Añada viñetas como personas/grupos pertenezcan a la audiencia del documento de Visión de Proyecto.

1.2. Definiciones, Siglas y Abreviaturas

Debe incluir las definiciones de todos los términos, acrónimos y abreviaturas necesarias para interpretar adecuadamente este plan. Se puede referenciar al glosario del proyecto

[***Item**: descripción*]
[***Item**: descripción*]

1.3 Referencias

Esta subsección incluye una lista de todos los documentos referenciados, cada uno de los cuales debería ser identificado por su título, número de informe (si se aplica), fecha y organización o fuente que lo publica.

- [referencia]
- [referencia]

2. Funcionalidades relevantes del proyecto

En esta sección se plantea hacer un esquema de las principales funcionalidades de negocio, que se descubren para el proyecto, puede ser utilizando una representación de objetos o clases desde el punto de vista empresarial. Para cada objeto de negocio hay que encontrar los atributos que representarían los principales repositorios de datos y en cuanto a métodos estos serían las actividades principales que debe realizar ese objeto

2.1 Servicios relacionados al proyecto

En este punto se debe revisar y especificar los servicios que serán necesarios para el proyecto, también se pueden incluir aquellos que ya existen dentro de la organización, y pueden ser utilizados por parte del proyecto, lo que facilitaría la reutilización y ahorro de tiempo.

Número	Nombre	Descripción	Tipo
<Id. del Servicio >	*<Nombre del Servicio >*	*< Detalle descriptivo del servicio>*	*< El tipo del servicio hará referencia a si es nuevo o existente >*

3. Análisis de Servicios

Para cada uno de los servicios descubiertos, analizar en base a los principios de los servicios web descritos en la sección 1.2.3.3 del marco teórico, los cuales están representados por las características que se detallan a continuación

Característica	*< Servicio …>*	*< Servicio …>*	*< Servicio …> …*
Reutilizable S: Si N: No			
Contrato S: Si N: No			
Acoplamiento A: Alto, M: Medio, B: Bajo			
Autónomo A: Alto, M: Medio, B: Bajo			
Sin Estado S: Si N: No			
Autónomo A: Alto, M: Medio, B: Bajo			
Puede ser descubierto S: Si N: No			

* *Para el detalle de las características revisar en el marco teórico el punto 1.2.3.2*.

4. Catálogo de Servicios Web

En este punto se define el interfaz de los servicios, el componente fundamental que define lo que ofrece el Servicio, además se dispondrá de un catálogo con información de entradas, salidas, ubicación y descripción de los mismos. Se trata de compartir esquemas (datos) y contratos (comportamiento) mediante mensajes de entrada y salida. En el contrato se definen políticas que serán parte de las validaciones (toda validación es una política). WSDL (Web Services Description Language) es el estándar utilizado para describir el servicio web, es un contrato entre el proveedor del servicio y el cliente mediante el que el proveedor del servicio indica:

- *Qué funciones se pueden invocar,*
- *Qué tipos de datos utilizan esas funciones,*
- *Qué protocolo de transporte se utilizará para el envío y recepción de los mensajes,*
- *Cómo acceder a los servicios.*
- *URL por el que se utilizan los servicios.*

No es necesario especificar explícitamente como quedarán los documentos WSDL para cada uno de los servicios detallados anteriormente ya que lo construyen automáticamente las herramientas de desarrollo, lo que se necesita es detallar el catálogo de servicios para el DDI junto a la información necesaria para establecer contratos. [Ver ANEXO XV. EJEMPLO DE CATALOGO DE SERVICIOS WEB] en donde se puede observar la descripción de los servicios y procesos que contienen, junto a la especificación de contratos. En este punto se haría referencia a ese catálogo

4.1 Registro global de servicios Web

Incluya aquí una descripción del servidor o sitio web en donde estarán disponibles los servicios creados para el proyecto.

5. Codificaciones Generales

En esta sección se incluyen los códigos que serán reconocidos para los resultados en las que pueden resultar las operaciones efectuadas con los servicios.

5.1 Tablas de códigos de acciones

EstadoExito

Código	Descripción
1	NoExiste
2	Rechazada
4	Autorizada
5	Problemas de comunicación
6	Servicio no autorizado.
7	Entidad Destino no disponible para procesar
8	Problemas en la respuesta del destino
9	Nombre de Usuario Incorrecto
10	Contraseña Incorrecta

TipoAccion

Código	Descripción
1	Inserción
2	Actualización
3	Eliminación
4	Inactivación

5.2 Tablas de códigos de participantes

Seguidamente se detallan los códigos asignados a cada una de las entidades participantes en la Universidad de Cuenca.

Grupos

CODIGO	Descripción
1000	Consejos
1500	Direcciones
2000	Dependencias
2500	Unidades
3000	Facultades
3500	Escuelas

4000	Departamentos
4500	Institutos
5000	Comisiones
5500	Centros
6000	Asociaciones
6500	Personas
7000	Programas

Consejos: 1100

Código	Descripción
1101	Consejo Universitario
1102	Consejo de Planificación
1103	Consejo Ejecutivo
1104	Consejo Directivo de Facultad
1105	Consejo Académico
1106	Consejo de Investigación

Direcciones: 1500

Código	Descripción
1501	Rector
1502	Vicerrector
1503	Dirección Administrativa-Financiera
1504	Jefatura de Recursos Humanos
1505	Decano
1506	Director de Escuela
1507	Dirección de Postgrados
1508	Dirección Investigación

Institutos: 4500

Código	Descripción
4501	Instituto de Educación Física

Dependencias: 2000

Código	Descripción
2001	Rectorado
2002	Vicerrectorado
2003	Archivo
2004	Presupuesto
2005	Centro Documental CDJBV
2006	Coordinación de programas y proyectos de investigación
2007	Departamentos de Investigación y Postgrado de Facultades. DIP

Unidades: 2500

Código	Descripción
2501	Secretaría General Procuraduría
2502	Unidad de Perfeccionamiento Académico

Departamentos: 4000

Código	Descripción
4001	Departamento de Desarrollo Informático
4002	Departamento de Admisión y Becas
4003	Departamento de Bienestar Universitario
4004	Departamento de Cultura
4005	Departamento de Relaciones Públicas
4006	Auditoría
4007	Departamento de Idiomas

Comisiones: 5000

Código	Descripción
5001	Comisión de Escalafón
5002	Comisión de Evaluación Interna y Acreditación
5003	Vinculación con la colectividad

Facultades: 3000

Código	Descripción
3001	Arquitectura
3002	Artes
3003	Jurisprudencia
3004	Filosofía
3005	Ciencias Médicas
3006	Ingeniería
3007	Arquitectura
3008	Ciencias Económicas
3009	Ciencias Agropecuarias
3010	Ciencias Químicas
3011	Odontología
3012	Artes
3013	Ciencias de la Hospitalidad
3014	Psicología

Asociaciones: 6000

Código	Descripción
6001	AETUC
6002	APUC

Programas: 7000

Código	Descripción
7001	PROMAS
7002	CESPLA
7003	CECEMIN
7004	PYDLOS
7005	ACORDES
7006	CEDIUC
7007	VLIR
7008	FONDO PROVIDA

Personas: 6500

Código	Descripción
6500	Persona Natural
6501	Persona Jurídica
6502	Decano
6503	SubDecano
6504	Responsable de Informática de Dependencias
6505	Docente
6506	Investigador
6507	Empleado
6508	Trabajador
6509	Directivo
6511	Estudiante
6510	Jubilado

5.3. Manejo de Excepciones

El manejo de excepciones provocadas en los mecanismos de validación o el procesamiento de las operaciones por parte del DDI pueden generar los códigos de error:

Código decimal	Código Hexadecimal	Descripción
-520421371	E0FB0005	La aplicación origen no se reconoce.
-520421371	E0FB0005	La aplicación destino no existe.
-520421370	E0FB0006	La entidad destino se encuentra inactiva
-520421369	E0FB0007	El motivo utilizado no existe
-520421366	E0FB000A	El código de referencia es inválido
-519372799	E10B0001	El mensaje no cumple con el formato establecido.
-520421361	E0FB000F	La entidad origen es inválida
-523501515	E0CC0035	Operación fuera de Horario
-523501514	E0CC0036	El código de referencia se encuentra repetido
-523501512	E0CC0038	El servicio no se encuentra disponible

ANEXO XVI. EJEMPLO DE CATALOGO DE SERVICIOS WEB

Introducción

El contenido de este documento describe el estándar electrónico para el procesamiento de operaciones de los servicios Web a cargo del Departamento de Desarrollo informático de la Universidad de Cuenca, provisto para las diferentes aplicaciones que se desarrollen en la Universidad de Cuenca y necesiten utilizar alguno de los procesos que a continuación se describen.

Este servicio permite el procesamiento de operaciones en tiempo real, es decir, mediante una interacción entre las aplicaciones universitarias mediante procesos expuestos en servicios web publicados en el servidor de aplicaciones del DDI, de tal forma que no exista un proceso que accede directamente al código de las aplicaciones ni directamente a las bases de datos de las mismas. Para lograr esta comunicación se utiliza cualquier tecnología que permita un paso de mensajes entre las aplicaciones y los servicios web para la comunicación entre servidores.

En particular, el objetivo de este documento es permitir a los departamentos de informática de toda la Universidad de Cuenca verificar el estado de sus sistemas internos e identificar los ajustes necesarios para invocar a los procesos expuestos de los servicios disponibles y así evitar contratiempos y demoras en la invocación de operaciones en los sistemas a cargo del DDI

Alcance

Este documento define el estándar electrónico para los servicios expuestos por el Departamento de Desarrollo Informático y es aplicable a todas las aplicaciones participantes en los diferentes procesos que intervienen para la utilización de los servicios que el DDI brinda.

Esta documentación será proporcionada al personal de desarrollo de software de los Departamentos, Facultades, Escuelas, Institutos y Organismos que necesiten consumir o actualizar información común a todas las entidades universitarias mediante aplicaciones informáticas que tienen acceso a los servicios que en el presente documento se describen.

Consideraciones

Los siguientes servicios descritos se los publicará en el servidor denominado aplicaciones, con el URL: 172.16.0.38, como casi la totalidad de servicios planteados serán internos, por el diseño de red interna, los programadores del Departamento de Desarrollo Informático podrán publicar transparentemente los servicios, ya que se ubicarán en la misma red donde desarrollan aplicaciones. Las aplicaciones consumidoras de servicios podrán acceder a un servicio web de la misma manera en que acceden al servidor de aplicaciones, las aplicaciones internamente se conectarán al servidor, el servicio de descubrimiento otorgará dentro de todos los servicios publicados, aquel proceso cuyo nombre y argumentos concuerden y ya estén disponibles en ese servidor. La seguridad a nivel de aplicación, se lo conseguirá considerando dentro de la invocación de un proceso que es parte de un servicio con los argumentos de nombre de usuario y contraseña, los cuales serán parámetros obligatorios, siempre necesarios de enviar entre los argumentos que requieren todos los procesos de los servicios a continuación descritos. El archivo con la definición del WebService (WSDL) debe ser utilizado por las herramientas de desarrollo de las entidades para generar el WebService descrito.

Términos empleados [10]

Para los fines del presente documento, se entenderá por:

DDI: Departamento de Desarrollo Informático.

UC: Universidad de Cuenca.

XML: Formato de intercambio de datos extendido, estándar para el WEB.

Validaciones en sistemas Origen y Destino

Las entidades destino deben efectuar una serie de validaciones al recibir los datos proporcionados por los servicios antes de aplicarlos a sus sistemas internos. Estas validaciones deberán ser realizadas en el orden que el departamento de desarrollo informático indique para todos los sistemas. En cuanto a los argumentos de entrada, estos deben ser proporcionados en el formato estándar que señale el DDI.

[10] *Para consultar algún otro término que aparezca en este documento, remítase al Glosario de la Tesis "Metodología de Desarrollo de Software hacia una Arquitectura SOA", desarrollada por la Ing. Claudia Espinoza León, 2010*

A continuación se detallan los dos primeros servicios, para ser considerados como una plantilla que permita en base a esta describir la manera en que se establecerían los contratos para el resto y futuros servicios que se necesitaran.

SERVICIO SR1 –ESTRUCTURA

Introducción	Procesos que describen y actualizan la estructura y organización general de la Universidad de Cuenca
Funcionalidad	Este servicio permitirá obtener la información general sobre las localidades, dependencias y como se encuentra estructurada de forma general la Universidad de Cuenca.
Versión	1.0
Servicios Utilizados	
Fecha Efectiva de Servicio	Septiembre 2009
Fecha de Terminación de Servicio	Indefinida
Propietario de Servicio	Departamento de Desarrollo Informático
Tiempo de revisión de Servicio	Anualmente
Contacto de Soporte de Servicio	Ing. Carmita Rojas
Contacto para Modificaciones	Ing. Marcelo Olivo
Disponibilidad del Servicio	Tiempo parcial
Horario de Operación	De Lunes a domingo Hora de Inicio: 04:00 Hora de Fin: 06:00
Carga de Transacciones Esperada	14 diarias
Número de usuarios concurrentes	3
Servicios y Procesos de Negocio relacionados	La mayoría de Procesos harán referencia a las dependencias existentes y la estructura universitaria.
Responsabilidades del Consumidor	• Realizar un Control de Errores de acuerdo al mensaje de salida de la operación realizada • El uso de este servicio no puede ser usado para fines diferentes a los expresados en el presente contrato • Informar en un tiempo menor a 48 horas cualquier anomalía presentada en el uso del presente servicio
Tiempo de Desarrollo de mejoras	2 semanas
Tiempo de Desarrollo de nuevas características	4 semanas
Tiempo de Respuesta de Operación.	• 5 segundos en responder el webservice • 10 minutos de procesamiento de transacción

Operaciones del SERVICIO SR1 –ESTRUCTURA

REVISAR_ESTRUCTURA

Descripción: Presenta la organización de la universidad de cuenca la misma que contiene la estructura de todas las dependencias universitarias, obteniendo información de todas las dependencias y subdependencias universitarias, en forma de un árbol recursivo.

Parámetros

Nombre	Tipo de Dato	Entrada/Salida
Usuario	String	Entrada
Contraseña	String	Entrada
ArbolDependencias	XML	Salida

REVISAR_UNADEPENDENCIA

Descripción: Extrae la estructura de la organización de las dependencias universitarias, obteniendo información de dependencias y subdependencias en forma de un árbol recursivo a partir de cierto código que puede ser de una Facultad, Escuela, Departamento, etc.. es decir a partir de cierta dependencia hacia dentro de esta.

Parámetros

Nombre	Tipo de Dato	Entrada/Salida
Usuario	String	Entrada
Contraseña	String	Entrada
Dependencia	Entero	Entrada
ArbolDependencias	XML	Salida

EDITAR_LOCALIDADES

Descripción: Permite actualización o eliminación de los datos de cierta localidad partir del código de la dependencia y localidad, devolviendo el estado de la transacción en el campo EstadoExito

Parámetros

Nombre	Tipo de Dato	Entrada/Salida
Usuario	String	Entrada
Contraseña	String	Entrada
Dependencia	Entero	Entrada
TipoAccion	Entero	Entrada
Localidad	Entero	Entrada
DatosLocalidad	XML	Entrada
EstadoExito	Entero	Salida

SERVICIO SR2 – PERSONA

Introducción	Procesos que se relacionan a datos generales de las personas que forman parte y se relacionan con la Universidad de Cuenca.
Funcionalidad	Permitirá obtener la información de una persona que labora en la universidad desde las distintas dependencias y carreras universitarias. Extrae y Verifica datos registrados de las personas que tienen relación de dependencia en la universidad, ya sean alumnos, docentes, empleados y trabajadores.
Versión	1.0
Servicios Utilizados	
Fecha Efectiva de Servicio	Septiembre 2009
Fecha de Terminación de Servicio	Indefinida
Propietario de Servicio	Departamento de Desarrollo Informático
Tiempo de revisión de Servicio	Cada 3 meses
Contacto de Soporte de Servicio	Ing. Carmita Rojas
Contacto para Modificaciones	Ing. Marcelo Olivo
Disponibilidad del Servicio	Tiempo parcial
Horario de Operación	De Lunes a domingo Hora de Inicio: 06:30 Hora de Fin: 10:00
Carga de Transacciones Esperada	60 diarias
Número de usuarios concurrentes	5
Servicios y Procesos de Negocio relacionados	• Servicio Estructura Varias aplicaciones revisan y actualizan especialmente datos generales de empleados y estudiantes de la Universidad de Cuenca
Responsabilidades del Consumidor	• Realizar un Control de Errores de acuerdo al mensaje de salida de la operación realizada • El uso de este servicio no puede ser usado para fines diferentes a los expresados en el presente contrato • Informar en un tiempo menor a 48 horas cualquier anomalía presentada en el uso del presente servicio
Tiempo de Desarrollo de mejoras	1 semana
Tiempo de Desarrollo de nuevas características	2 semanas
Tiempo de Respuesta de Operación.	• 5 segundos en responder • 8 minutos de procesamiento de transacción

Operaciones del SERVICIO SR2 – PERSONA

VERIFICA_PERSONA

Descripción: Verifica si una persona al proporcionar su número de cédula y/o sus nombres, consta en el registro de la información de las personas que se relacionan con la Universidad de Cuenca ya sean personas naturales como estudiantes, docentes, o personal administrativo ó personas jurídicas, en caso de estar registrada se devuelve un código que identifica un tipo de persona (ver tabla codificaciones persona:6500) y si el usuario tiene los permisos necesarios, se envía como retorno la información detallada en el campo DatosPersona

Parámetros

Nombre	Tipo de Dato	Entrada/Salida
Usuario	String	Entrada
Contraseña	String	Entrada
Cedula	String	Entrada
Nombre*	String	Entrada[11]
CodigoPersona	Entero	Salida
DatosPersona	XML	Salida
EstadoExito	Entero	Salida

EDITA_PERSONA

Descripción: Posibilita realizar tareas de edición de una persona como inserción, modificación e inactivación de los datos de cierta persona que tiene un registro dentro de la Universidad de Cuenca.

Nombre	Tipo de Dato	Entrada/Salida
Usuario	String	Entrada
Contraseña	String	Entrada
TipoAccion	Entero	Entrada
Cédula	String	Entrada
Nombre*	String	Entrada[1]
DatosPersona	XML	Entrada
EstadoExito	Entero	Salida

[11] Campo Opcional

REVISAR_ESTUDIANTES

Descripción: Proceso para revisar si una persona es un estudiante y obtiene su información detallada, se proporciona una cédula y opcionalmente el nombre; y el método devolverá si es o no un estudiante. Si se tiene los permisos necesarios se retorna en formato XML la información detallada del mismo.

Nombre	Tipo de Dato	Entrada/Salida
Usuario	String	Entrada
Contraseña	String	Entrada
Cédula	String	Entrada
Localidad	Entero	Entrada
DatosEstudiante	XML	Salida
EstadoExito	Entero	Salida

REVISAR_ADMINISTRATIVOS

Descripción: Proceso para revisar si una persona forma parte del personal administrativo y obtiene su información detallada según el campo cédula y opcionalmente el nombre; el método devolverá en EstadoExito si puedo encontrarse como administrativo, si es así y si se tiene los permisos necesarios se retorna en formato XML la información detallada del mismo indicando dentro de estos datos, que tipo de personal administrativo es.

Nombre	Tipo de Dato	Entrada/Salida
Usuario	String	Entrada
Contraseña	String	Entrada
Cédula	String	Entrada
Localidad	Entero	Entrada
DatosAdministrativo	XML	Salida
EstadoExito	Entero	Salida

REVISAR_DOCENTES

Descripción: Proceso para revisar si una persona es un docente de la universidad, en caso de que así sea y si el usuario es el adecuado, se devolverá su información detallada en el campo DatosDocente, el proceso necesita como datos una cédula y opcionalmente el nombre.

Nombre	Tipo de Dato	Entrada/Salida
Usuario	String	Entrada
Contraseña	String	Entrada
Cédula	String	Entrada
Localidad	Entero	Entrada
DatosDocente	XML	Salida

EstadoExito	Entero	Salida

5. Priorización y Escalamiento

En caso de existir inconvenientes se debe seguir el siguiente esquema de escalamiento

Nivel 1. Contacto de Servicio de Soporte: Secretaría del Departamento de Desarrollo Informático

Se le proveerá la información:

- Información sobre uso de servicio
- Información sobre parámetros

Nivel 2. Contacto con proveedor de Servicio: Ing. de Sistemas responsable de utilización de servicios en el Departamento de Desarrollo Informático

Se le proveerá la información:

- Información detallada de error presentado
- Datos incoherentes

Nivel 3. Propietario de Servicio: Programador responsable del servicio

Se le proveerá la información:

- Servicio inalcanzable
- Resultados erróneos
- Servidor no disponible