UNIVERSIDAD CENTROCCIDENTAL “LISANDRO ALVARADO”
DECANATO DE CIENCIAS Y TECNOLOGIA
MAESTRIA EN CIENCIAS DE LA COMPUTACION

PROPUESTA DE UNA ARQUITECTURA ORIENTADA A SERVICIO PARA LA DIRECCION DE ADMISIÓN Y CONTROL DE ESTUDIOS DE LA UCLA

Trabajo presentado para optar al grado de Magíster Scientiarum

Por: ALY GONZALEZ RUIZ

Barquisimeto, 2008

ÍNDICE

INDICE DE TABLAS

ÍNDICE DE FIGURAS

UNIVERSIDAD CENTROCCIDENTAL "LISANDRO ALVARADO"
DECANATO DE CIENCIAS Y TECNOLOGIA
MAESTRIA EN CIENCIAS DE LA COMPUTACION

PROPUESTA DE UNA ARQUITECTURA ORIENTADA A SERVICIO PARA LA DIRECCION DE ADMISIÓN Y CONTROL DE ESTUDIOS DE LA UCLA

Autor: Aly González

Tutor: Ramón Valera

RESUMEN

En los últimos tiempos producto del desarrollo que ha tenido la Ingeniería de Software, se ha evidenciado que uno de los problemas que aún afectan el desempeño de la tecnología de los sistemas y por ende al propio negocio, es el vacío existente entre los negocios y la tecnología que los apoya. Así pues, para afrontar esta problemática han emergido nuevos paradigmas en el análisis y el diseño de sistemas, dentro de los cuales se encuentra el abordado en esta investigación, que es la orientación a servicio y su correspondiente producto, identificado como arquitectura orientada a servicio (SOA, por sus siglas en ingles). El contexto de aplicación de este paradigma y la tecnología asociada es la Dirección de Admisión y Control de Estudios (DACE) de la Universidad Centroccidental "Lisandro Alvarado" (UCLA), donde a través de instrumentos de recolección de datos, se pudo confirmar la existencia de problemas comunes que se pueden abordar con el paradigma en cuestión. En la actualidad la tecnología de facto en la implementación de SOA, es la conocida como los servicios Web que a su vez utilizan principalmente estándares tales como XML, WSDL y SOAP, que son utilizados para ayudar a cumplir con promesas hechas por los componentes de una SOA, que son la reusabilidad, la capacidad de composición, la propiedad de no tener estado y la capacidad de ser descubiertos y por consiguiente dotar a las organización de mayor agilidad antes los cambios en las reglas del negocio.

Palabras clave: Orientación a servicio, SOA, DACE, UCLA, servicios Web, XML, WSDL, metodología de análisis y diseño.

INTRODUCCIÓN

En los últimos tiempos el desarrollo de las llamadas tecnologías de la información (TI) y la ingeniería de software (IS), ha producido herramientas de para mejorar la producción de software, así pues se pueden observar nuevos paradigmas metodológicos para llevar a cabo y controlar el ciclo de vida de los sistemas, aplicaciones para documentar los procesos, herramientas para realizar el análisis y diseño, lenguajes de modelado, programas gestores de proyectos, etc. Pero al parecer todavía sigue existiendo una demanda por llenar el vacío existente entre el modelo de negocios y el software que lo implementa, por lo que han surgido modelos para tratar de llevar a cabo este propósito, dentro de los cuales la orientación a servicio surge como candidato, cuyo fin es la entrega de una arquitectura orientada a servicio (Software Oriented Architecture, SOA), para llenar el vacío existente entre el model,o de negocios y la tecnología, a través de un nuevo enfoque para la separación de intereses y la aplicación de criterios, tales como reusabilidad, capacidad de composición y capacidad de descubrimiento en los componentes de las arquitecturas entregadas.

Tomando en cuenta este panorama, surge la presente investigación, que tiene como propósito ofrecer una SOA, para la Dirección de Admisión y Control de Estudios de la UCLA (DACE), cuyo objetivo sea, en primera instancia dotar a la unidad de capacidad para ofrecer información estandarizada a los demás entes de la institución y disponer de componentes de software en forma de servicios que sirvan para que otras unidades organizativas los incluyan en los desarrollos de que emprendan. Para su construcción se aplicará la metodología enunciada en Earl (2005), la cual consiste en llevar a cabo el análisis y diseño, con un nuevo enfoque en la separación de intereses y así obtener componentes de una arquitectura que refleje el modelo de negocios y reaccionen hábilmente a los cambios producidos en este.

Con la finalidad de organizar la documentación de la investigación y cumplir con una metodología que permita un orden lógico en las fases, el trabajo se ha dividido en cinco capítulos, cuyo contenido se explica a continuación:

Capítulo I, en el que se plantea el problema, se define el objetivo principal y los particulares o específicos, se plantea el alcance y las limitaciones.

Capítulo II, en el cual se reseñan investigaciones anteriores que sirvieron de antecedentes a la investigación, comentando su contenido y el aporte realizado. También se realiza una documentación de la revisión bibliográfica realizada, de donde se obtendrá el basamento teórico de la investigación.

Capítulo III, se realiza una ubicación del tipo y modalidad de la investigación a partir de criterios aceptados y se establecen los instrumentos de recolección de datos. Este capitulo se complementa con el planteamiento de un estudio de factibilidad desde los puntos de vista técnico, operativo, económico y psicosocial.

Capítulo IV, contempla el desarrollo de la propuesta, en donde el contenido está definido por el orden de los objetivos específicos planteados en el planteamiento del problema. Esta secuencia incluye un razonamiento para la selección de la estrategia de entrega de SOA, una aplicación de métodos para realizar el análisis y diseño orientado a servicio.

Capitulo V, compuesto por las conclusiones y recomendaciones emanadas del trabajo de investigación realizado.

CAPITULO I

EL PROBLEMA

Planteamiento del problema

En los últimos años la mayoría de los procesos de negocio han cambiado en cuanto a flexibilidad, interconectividad y autonomía debido a las condiciones del mercado, a los nuevos modelos organizacionales y a los escenarios de uso de los sistemas de información, Pelechado (2005). En ese contexto, Internet y la Web están cambiando la forma en la que se ofrecen los negocios y los servicios a la sociedad global.

Para enfrentar esta realidad, en el campo de la Ingeniería de Software se han propuesto acortar la brecha existente entre el llamado negocio o dominio y el software que le sirve de base para su operación. Es así como han surgido nuevos enfoques para emprender desarrollos de software, donde el modelado del software sea controlado por el modelado de los negocios, para lo cual proponen la aplicación de una serie de principios a la hora de llevar a cabo las actividades del proceso de desarrollo y de esta manera dotar a las organizaciones y por ende a sus sistemas de de mayor agilidad para reaccionar a los cambios suscitados en el negocio.

Uno de estos enfoques mencionados, es lo que actualmente se conoce como orientación a servicio, el cual introduce principios generalmente aceptados para controlar la disposición y el diseño de los componentes dentro de la arquitectura del software. En este sentido Earl (2005) sostiene que la aplicación de tales principios en la construcción de dichos componentes, es lo que lleva al fin último en el desarrollo de sistemas bajo este modelo, que es entregar una arquitectura orientada a servicio (Software Oriented Architecture, SOA).

Adicionalmente Sommerville(2005) afirma que la Ingeniería de Software debe superar en el siglo XXI, los retos de heterogeneidad, el de la entrega y el de la confianza, lo cuales no son independientes. Este autor cita:

> "Por ejemplo, es necesario hacer cambios a los sistemas heredados para proveerlos de una interfaz de servicio Web. Para tratar estos retos necesitaremos nuevas herramientas y técnicas, así como formas innovadoras de combinación y uso de métodos de ingeniería de software existentes".

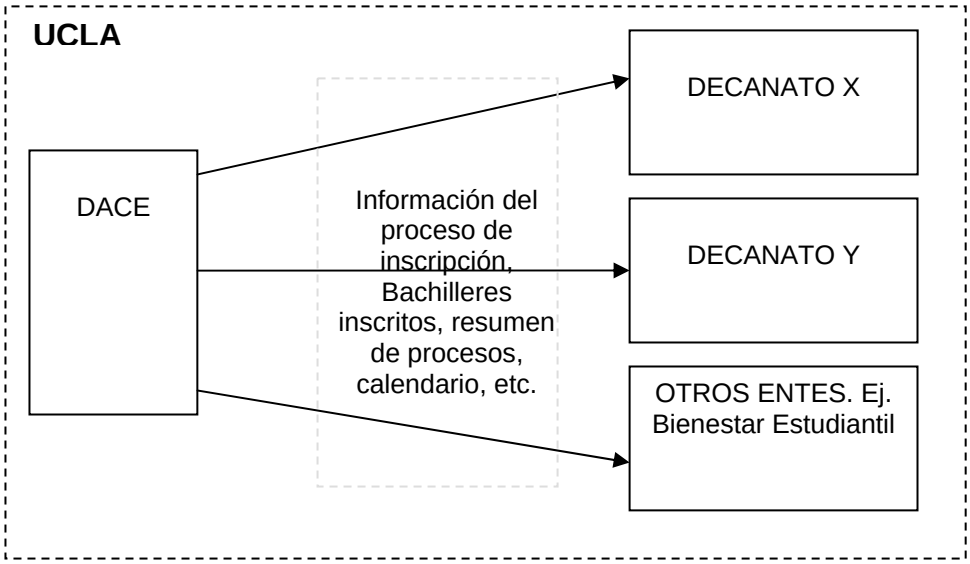
Para enfrentar dichos desafíos, también es necesarios implementaciones tecnológicas, ahora disponibles, como es el caso de los Servicios Web o Web Services (WS), que a su vez dan un auge importante a las arquitecturas de sistemas basadas en servicios, las cuales se definen según W3C (2003), como un tipo de sistema distribuido donde los agentes son "servicios" .

Adicionalmente, para la implementación de estos servicios, hoy en día se cuentan con tecnologías como XML "eXtensible Markup Language", un lenguaje entendido tanto por humanos como por las maquinas (Skonnard y Gudgin, 2002), para permitir la uniformidad en la información intercambiada; el WSDL "Web Service Description Language", lenguaje que hace posible la descripción de los servicios ofrecidos, esto es, lo que realizan los servicios y sus propiedades publicables; UDDI "Universal Description, Discovery and Integration, directorio que contiene un registro/repositorio de descripciones de servicios Web; SOAP "Simple Object Acces Protocol" protocolo de comunicación que establece las reglas básicas para el envío de mensajes en XML.

Una vez descrita la situación desde el punto de vista de la tecnología disponible y de los modelos de metodología para analizar y diseñar sistemas, también es conveniente abordar el contexto organizacional donde se prevé aplicar los recursos tecnológicos citados anteriormente, como lo es la Dirección de Admisión y Control de Estudios (DACE) de la Universidad Centroccidental "Lisandro Alvarado" (UCLA). Esta Dirección, entre otras funciones, tiene a su cargo llevar la gestión del proceso de inscripción de los Bachilleres asignados a la UCLA, para lo cual debe administrar y distribuir la información que se produce

durante todo proceso de gestión. La información administrada se provee a los demás unidades de la organización, en especial a las oficinas de registro académico de los Decanatos de la UCLA. De manera muy general se describe gráficamente en la Figura 1.

Figura 1. Funcionamiento general y distribución de información en la DACE.



Fuente: El autor de la investigación.

Al revisar este escenario y tomando en cuenta que la Universidad Centroccidental "Lisandro Alvarado", UCLA, es una organización que desarrolla y mantiene sistemas de software, se pueden identificar algunos síntomas de problema que pueden ser enfrentados con las herramientas tecnológicas y los modelos identificados. Estas situaciones son:

- Aplicaciones desarrolladas de manera independiente y aislada, en cada uno de los entes que conforman el ambiente tecnológico de trabajo. Por ejemplo en una unidad de registro académico existen modelos de trabajo apoyados en herramientas (motores de base de datos y lenguajes) totalmente dispares con los demás unidades de registro y con la Dirección de Admisión y Control de Estudios, lo que dificulta la consolidación de una base de datos centralizada y en el caso más crítico entorpece el transito de dicha información a través de la organización. Informe técnico UCLA (2005). Actualmente se realiza una centralización de la información por medio de algoritmos de traducción de datos desde los

registros académicos hacia la Dirección de Admisión y Control de Estudios y a su vez se comparte información desde la Dirección hacia los registros. Estos algoritmos son específicamente desarrollados para cada registro o sea para cada modelo de datos.

- Se presentan obstáculos a la hora de emprender nuevos desarrollos y reaccionar eficientemente a cualquier cambio en el ambiente de negocios, ya que no se tiene un modelo de negocio bien simulado en lo tecnológico, en otras palabras la información ni los procesos están estandarizados. En cada ente de la organización a nivel de aplicación se habla un lenguaje totalmente diferente.
- Los nuevos desarrollos de software se han realizado y se realizan desde cero, no se hacen de manera oportunista, o sea utilizando los componentes o servicios que otros entes (decanatos y direcciones) han desarrollado, aunque estos se encuentran ubicados en el mismo negocio.

Síntomas parecidos a estos que se describen, han sido abordados mediante el empleo de SOA, tal como lo señala Krafzig et.al(2004). Dentro de los casos que se pueden citar se encuentra la aplicación de este modelo en empresas tales como Deutsche Post AG, Winterthur Group, Credit Suisse Group y Halifax Bank of Scotland (HBoS), organizaciones de alto rango el concierto empresarial mundial. En estas experiencias, al enfrentarse el desarrollo de software desde el enfoque SOA, se ha logrado analizar el dominio y diseñar componentes "servicios" de arquitecturas que reflejen el negocio, que permitan bajo acoplamiento y que ofrezcan una alta disponibilidad, para que puedan ser utilizados por otras aplicaciones y donde la interconexión entre estas sea simplemente un beneficio más dentro de la aplicación de la orientación a servicio.

Así pues luego de revisar la situación tecnológica actual, los modelos teóricos, contexto del negocio y experiencias previas en problemas similares, se plantea el objetivo de proporcionar mecanismos a la DACE, a través de una SOA, los datos generados dentro de los procesos que administra y la lógica empleada para llevarlos a cabo, sean asequibles para las distintas unidades o dependencias

de la UCLA, en el momento oportuno y de forma automática. Dentro de los procesos que administrar se abordaran los principales procesos que se llevan a cabo en la Dirección como son los relativos a la preinscripción de bachilleres, inscripción de Bachilleres y emisión de informes de la información administrada.

Objetivos

Objetivo general

Diseñar las bases de una arquitectura orientada a servicios (SOA), para dotar a la Dirección de Admisión y Control de Estudios de la UCLA de capacidad para ofrecer información de manera estándar a diferentes entes demandantes de servicios de información y prioritariamente a las oficinas de registro académico de los diferentes Decanatos de la UCLA.

Objetivos específicos

1. Realizar un diagnóstico en el entorno de trabajo de la DACE de la situación actual, en cuanto al modelado de los sistemas, las maneras de intercambio de información y las expectativas del personal.
2. Analizar los resultados del diagnóstico para proponer la utilización del paradigma de orientación a servicio en la resolución del problema.
3. Analizar los procesos que se llevan a cabo en la DACE.
4. Seleccionar una estrategia para realizar un análisis y diseño orientado a servicio dentro de la DACE.
5. Realizar el análisis orientado a servicios, determinado la lógica del negocio, para identificar las operaciones y servicios candidatos que conformaran la arquitectura orientada a servicio dentro de la DACE.
6. Realizar el diseño de los servicios dentro de la DACE.

Justificación e Importancia

La presente investigación, aunque cubrirá solo hasta la etapa de diseño, deja un aporte en lo que respecta a la aplicación de técnicas establecidas en la Ingeniería de Software en el campo práctico, al demostrar que dichas técnicas mejoran la calidad de parte del proceso de producción de software. Para la organización donde se desarrollará la investigación, en este caso la UCLA, significará una contribución a su patrimonio, el cual puede ser utilizado como referencia para futuros desarrollos de software, en lo que respecta a las fases de análisis y diseño.

Al realizarse este proyecto bajo el esquema de SOA y utilizando técnicas de la Ingeniería de Software, se establece una base para el desarrollo sencillo y rápido de otras aplicaciones, que sean necesarias en otras entidades organizativas dentro de la UCLA, por medio del aprovechamiento los servicios de información distribuidos en una arquitectura orientada a servicios funcionando en la DACE.

La puesta en funcionamiento del conjunto de servicios detectados, posibilita el acceso a la información requerida por entes externos, gobierno y sociedad, usando estándares aceptados universalmente para el intercambio de datos.

Con el desarrollo de la propuesta se sientan las bases para llevar a cabo el aprovechamiento de una infraestructura de red existente, la cual comunica todos los decanatos con el rectorado de la UCLA, y da soporte al despliegue de aplicaciones distribuidas utilizando la Web. Es importante recordar que la utilización del recurso se manifiesta es cuando se realiza la puesta en funcionamiento de un producto tecnológico, objetivo este que no es el fin del presente trabajo.

Es conveniente señalar que autores como Sommerville (2005) citan una serie de ventajas de los sistemas que trabajan bajo esta modalidad de SOA, dentro de las cuales se encuentran las siguientes:

- Los servicios pueden ofertarse por cualquier proveedor de servicios dentro o fuera de una organización. Suponiendo que éstos cumplen con ciertos estándares, las organizaciones pueden crear aplicaciones integrando servicios desde varios proveedores. Por ejemplo, una compañía de fabricación puede enlazar directamente a los servicios proporcionados por sus proveedores. Al aplicar este enunciado en contexto de la presente investigación se puede prever que los servicios de la DACE se ofrezcan a otras entidades de la UCLA, a entidades gubernamentales y cualquier organización que lo requiera y que el marco legal y operativo lo permita.
- El proveedor de servicios hace pública la información sobre el servicio para que cualquier usuario pueda usarlo. El proveedor de servicios y el usuario de los servicios no necesitan negociar sobre lo que el servicio hace antes de ser incorporado en un programa de aplicación.
- Las aplicaciones pueden retrasar el enlace de los servicios hasta que éstas sean desplegadas o hasta que estén en ejecución. Por lo tanto, una aplicación que usa un servicio de precios de productos, por ejemplo podría cambiar dinámicamente los proveedores de los servicios mientras el sistema se está ejecutando.
- Es posible la construcción oportunista de nuevos servicios. Un proveedor puede reconocer nuevos servicios que se crean, enlazando servicios existente de formas innovadoras. En el contexto de la DACE, un ejemplo de esta ventaja podría ser que la Dirección de Bienestar Estudiantil construya un servicio de estudiantes con beneficios e incluya como parte de este a los servicios que se ofrecen en la DACE.

Adicionalmente Earl (2005), plantea que una solución orientada a servicio, dentro de tantos beneficios, puede reaccionar de forma ágil a los cambios sucedidos en el ambiente de negocios y dar servicio a nuevos clientes sin la necesidad de desarrollar nuevos servicios, lo que se denomina agilidad organizacional.

Alcances y Limitaciones

El desarrollo de la presente investigación permitirá implementar un modelo de aplicación distribuida, compuesto por unidades de lógica de aplicación y de negocios, en la forma de una arquitectura orientada a servicio (SOA), lo que dará a la institución la capacidad de ofrecer información a las diferentes unidades organizativas de la UCLA y adicionalmente establecer una base tecnológica y de negocio, para que los servicios creados sean utilizados por entidades externas, esto de acuerdo a las políticas establecidas por la organización. Así también, en la medida en que los componentes desarrollados cumplan con los principios de orientación a servicio, estos pueden ser utilizados como parte de otras soluciones dentro de la Institución.

Sólo se diseñará un grupo significativo de servicios que cumplan con SOA, lo que constituirá una base para construcciones de servicios adicionales, la cual debe apuntar hacia niveles superiores de estandarización enunciados en la teoría de lo que es un ambiente orientado a servicio. Estos servicios que se abordaran estarán conformados por el corazón operativo de la DACE que es la preinscripción e inscripción de estudiantes y la emisión de informes, exceptuando en primera instancia el proceso de asignación ya que este en su mayoría es responsabilidad del CNU-OPSU y escapa al control organizacional de la DACE.

Para el desarrollo de la propuesta se utilizaran técnicas de Ingeniería de Software, cuyo uso puede servir de referencia para aplicaciones posteriores y coadyuvar a la mejora de procesos y productos de software en proyectos similares dentro de la Institución. Dentro de estas técnicas se puede nombrar Ingeniería de Requisitos y aplicación de lenguajes de descripción arquitectónicos o ADLs “Architecture Description Languages”, como por ejemplo UML. Es importante señalar que existen estudios donde se considera que UML no cumple con todas las características que debe tener un ADL, Reynoso C. y Kicillof N. (2004). Así también emergen otras opiniones como las de Booch G. (1999), donde se expresa que efectivamente UML es un ADL, esto por nombrar algunas

de las más relevantes y enfrentadas. Por otro lado se puede traer a colación la opinión mediadora de Garlan D. (1999), que expresa que UML no es un buen ADL, sin embargo se pueden describir arquitecturas con el. Para efectos de la presente investigación se considera el UML como un implementación incompleta de un ADL, pero suficiente para utilizarla como herramienta en este estudio.

Una de las limitantes de SOA, citada por algunos autores como Sommerville (2005), es la madurez de las tecnologías que soportan dicho modelo, lo que podría traducirse en poca escalabilidad del modelo. Aunque debe tomarse en cuenta la velocidad con que se desarrolla la tecnología asociada al software y el apoyo que esta recibiendo por parte de empresas y de comunidades de software para dar soporte a SOA. También para argumentar la selección de SOA y la correspondiente metodología de su construcción, hay que revisar los números de crecimiento de su utilización en las organizaciones empresariales; al respecto Chen (2007), comunica datos como que Gartner en 2003 reporta que SOA se convertirá en la solución de arquitectura empresarial dominante en el futuro, basándose en el amplio crecimiento que ha adoptado en los últimos cuatro años. De forma similar Ovum surveyed 333 US IT, en un estudio de 2006 encontró que 27% de las grandes empresas y 17% de medianas empresas, han desplegado SOA en forma operativa en al menos algunas áreas de su infraestructura de TI y que comparado con CORBA (que nunca fue adoptado ampliamente) y EAI/BPM (el cual ha ganado terreno en los pasados siete años), SOA luce como un claro ganador en cuanto a paradigmas empresariales.

La investigación no tratará sobre los requerimientos de seguridad en ninguna etapa del desarrollo, lo cual puede ser tratado como tema aparte o complementario en futuras investigaciones.

Una de las limitaciones a tomar en cuenta es que la investigación va a estar desarrollada en el contexto de solo algunas unidades organizativas de la UCLA, lo que dejará sin demostrar la posible y potencial intermediación con otros entes internos y/o externos a la UCLA.

Un aspecto deseable de la investigación, es que el producto tecnológico que se obtenga apoye el funcionamiento de la Dirección de Admisión y Control de Estudios, pero hay que tomar en cuenta que otros factores que no son abordados, como por ejemplo decisiones administrativas y políticas de estado, pudieran influir en la utilidad del producto para este contexto.

CAPITULO II

MARCO TEÓRICO

Antecedentes

SOA por ser un nuevo paradigma en el ámbito del desarrollo de software, ha despertado el interés de muchos investigadores del área, así también las organizaciones han migrado sus plataformas tecnológicas o en el peor de los casos van en la ruta de adoptar SOA como un nuevo enfoque a la hora de producir software. A continuación se hace mención a algunas de estas investigaciones y experiencias de implantación.

Perepletchikov M, et al (2005), investigaron sobre el impacto de las estrategias de desarrollo de software en el proyecto y en los atributos estructurales del software en SOA, donde persiguen la identificación de tareas que conlleven a un diseño e implantación exitosos de los servicios e investigar los efectos de dichas tareas en el proyecto y en los atributos estructurales del software en SOA, disminuyendo así el esfuerzo y los costos de un proyecto SOA. Las tareas identificadas de realizan para cada una de las tres estrategias de desarrollo conocidas, top-down, bottom-up y met-in-the-middle o estrategia ágil. Este trabajo de investigación servirá de apoyo a la selección de las tareas y las estrategias a utilizar en la presente investigación.

Papazoglou M, et al (2005), realizaron un estudio en el cual revisan tecnologías y enfoques para unificar los principios y conceptos de SOA con aquellos de la programación basada en eventos. La investigación también se enfoca en lo que se conoce como el ESB “Enterprise Service Bus”, para ofrecer un “backbone” manejable, basado en estándares, que ofrezca la posibilidad de conectar componentes y sistemas heterogéneos y provea servicios de integración. Así también, este trabajo propone un enfoque para extender la SOA convencional

para que los requerimientos del ESB incluyan capacidades tales como servicios de orquestación, enrutamiento "inteligente", aprovisionamiento, integridad y seguridad de los mensajes, así como también el manejo de servicios.

Pelechado V, et al (2002), proponen un método OO de producción de software para el desarrollo de "aplicaciones Web" basadas en Servicios WEB XML, en el que proporcionan herramientas de modelado conceptual necesarias para especificar requisitos funcionales y navegacionales de aplicaciones Web dinámicas, realizan un estudio de la naturaleza de los servicios Web y de sus arquitecturas middleware asociadas, proponen una arquitectura software multinivel y definen una serie de correspondencias entre las abstracciones conceptuales y los elementos software que implementan cada uno de los niveles de la arquitectura.

Tsenov M (2006), realiza una investigación donde explica la decisión de utilizar servicios Web entre sistemas de red distribuidos, el método utilizado lo basa en estándares de fuente abierta SOAP y WSDL. La solución se basa en una arquitectura propuesta, desarrollada utilizando una extensión de SOAP para PHP, la cual se presenta y se explica. En este trabajo se puede apreciar un ejemplo en el campo práctico de las soluciones que promueve SOA.

Stojanovic Z, et al (2003), en su investigación platean como el modelado basado en componentes y principios de diseño, se pueden utilizar como base para modelar una SOA. El enfoque de diseño propuesto básicamente en manejado por modelo "model driven", pero incorporando algunos principios y prácticas de desarrollo ágil que proveen flexibilidad y agilidad para afrontar cualquier cambio en el negocio o en el ambiente de IT.

Martin S y Mike D. (2003), examinan el uso de servicios Web en la grid europea de observación solar "The European Grid of Solar Observations (EGSO)", como un significado de comunicación entre los varios roles del sistema. Esto en el contexto en donde la EGSO concibe una grid computacional para archivos de data solar heterogénea y federada que confluyan en un archivo simple

"virtual", que le permita a los científicos localizar y recuperar fácilmente conjuntos de datos particulares de múltiples fuentes.

Ambroszkiewicz S. (2003), propone una tecnología para la composición y descripción de servicios en un ambiente abierto y distribuido. La tecnología consiste en un lenguaje de descripción, llamado Entish, y un protocolo de composición denominado Entish 1.0, los cuales están basados en el paradigma de agentes de software. El lenguaje descriptivo es el lenguaje del contenido de los mensajes se intercambian (entre agentes y servicios) de acuerdo al protocolo de composición. Tanto la sintaxis del lenguaje como los mensajes son expresados en XML y son solo especificaciones. Para mostrar que la tecnología es operativa dispone de una dirección Web para el uso y evaluación vía interfaces Web.

D'andrea V. y Aiello M. (2003), realizan un comparación entre el paradigma orientados a objetos y el orientado a servicio, estableciendo similitudes y diferencias, y buscando dentro de otras finalidades, el aprovechamiento de la suficiente cantidad de metodologías y técnicas de orientación a objetos para el desarrollo de servicios.

Bases Teóricas

El basamento teórico de la presente investigación se centra en dejar claro lo que significa SOA, los beneficios y las maneras de construirla, para cuyo propósito se abordan texto de autores como Earl (2005), Krafzig et al. (2004), Pelechano (2005) y grupos de discusión o foros especializados o que se encuentren relacionados de alguna u otra forma con el tema. Se expone la temática de la utilización del lenguaje UML como un lenguaje de descripción de arquitecturas "Architecture Description Language (ADL)", el cual es un lenguaje que puede apoyar la documentación del proceso de análisis y diseño que se lleva a cabo para la construcción de una SOA. Adicionalmente se aborda el tema de los servicios Web y la correspondiente tecnología subyacente (SOAP, WSDL, UDDI).

1. Fundamentos de SOA.

Debido a que el término "orientado a servicio" existe desde hace algún tiempo, este ha sido usado en diferentes contextos y para diferentes propósitos. Una constante a través de su existencia es que representa un enfoque distinto para la separación de aspectos o de intereses; lo que significa que la lógica requerida en la solución de un problema grande puede construir de mejor manera, llevarse a cabo y manejarse, si se descompone en una colección de pequeñas piezas relacionadas, en donde cada una de esta trate un aspecto o una parte específica del problema.

Este enfoque trasciende las soluciones de la tecnología y de la automatización. Es una teoría establecida y genérica que se puede utilizar para tratar una variedad de problemas. Lo que distingue el enfoque orientado a servicio, en lo que respecta a la separación de intereses, es la manera en la cual este alcanza dicha separación.

1.1. Una analogía de orientado a servicio

Tomemos una ciudad cosmopolita promedio. Esta está llena de negocios orientados a servicio. Las compañías individuales están orientadas a servicio en la que cada una provee un servicio distinto que puede ser usado por múltiples consumidores. Colectivamente estos negocios abarcan una comunidad de negocios. No tiene sentido para una comunidad de negocios ser servida solo por una sola tienda de distribución que preste todos los servicios. Descomponiendo la comunidad en tiendas especializadas e individuales, se logra un ambiente en el cual dichas tiendas pueden ser distribuidas.

Cuando se juntan "arquitectura" y orientación a servicio toma una connotación técnica. "arquitectura orientada a servicios" es un termino que representa un modelo en el cual la lógica de automatización es descompuesta en pequeñas unidades distintas de lógica. Colectivamente, estas unidades conforman un gran componente de lógica de automatización de negocios. Dichas unidades pueden ser distribuidas de forma individual.

Distribuir la lógica de automatización en unidades separadas no es nada nuevo. ¿Qué es entonces lo que hace la separación orientada a servicios tan diferente? Se pueden hacer algunas distinciones notables en una primera revisión.

Incluso en una comunidad de negocios distribuidos, si imponemos desagradables dependencias, podemos inhibir el potencial de los negocios individuales. Aunque se quiera permitir la interacción entra las tiendas y que demanden servicios de las demás, se debe evitar un modelo en el cual las tiendas formen fuertes conexiones que se traduzcan en interdependencias constrictivas. Darle el poder a los negocios para controlar sus servicios individuales, permite que estos crezcan y se desarrollen de forma relativamente independiente unos de los otros.

Aunque se aliente la independencia de las tiendas de distribución, se debe asegurar que estas acepten adherirse a cierta base de convenciones, por ejemplo, una moneda común para el intercambio de bienes y servicios, un código local que

requiera símbolos para cumplir con ciertos parámetros, o quizás se requiera que todos los empleados hablen el mismo lenguaje que los consumidores nativos. Estas convenciones estandarizan aspectos claves de cada negocio para el beneficio de los consumidores, sin influir significativamente en la facultad de los negocios individuales de ejercer su auto gobierno.

Similarmente, la arquitectura orientada a servicio (SOA) impulsa las unidades de lógica individuales, pero que no estén aisladas unas de las otras. A las unidades de lógica se les requiere conformar un conjunto de principios que les permita desarrollar independencia, mientras aun mantienen una suficiente cantidad de coherencia y estandarización. En SOA, estas unidades de lógica son conocidas como servicios.

1.2. Encapsulación de la lógica en los servicios

Para mantener su independencia, los servicios encapsulan dentro de un contexto distinto. Este contexto puede ser una tarea de negocio específica, una entidad de negocio o alguna otra agrupación lógica.

El aspecto tratado por un servicio puede ser pequeño o grande. Por consiguiente, el tamaño y el alcance de la lógica representada por el servicio pueden variar. Adicionalmente, la lógica del servicio puede abarcar la lógica que otros servicios ofrezcan. En este caso, uno o más servicios están incluidos en un colectivo.

Por ejemplo, las soluciones de automatización de negocios normalmente son una implementación de un proceso de negocio. Dicho proceso está compuesto por lógica que fija las acciones realizadas por la solución. La lógica es descompuesta en una serie de pasos que se ejecuten en una secuencia predefinida de acuerdo a las reglas del negocio y condiciones de ejecución.

Como se muestra en la figura 1.1, cuando se construye una solución de automatización compuesta por servicios, cada servicio puede encapsular una tarea realizada por un paso individual o un subproceso compuesto de un conjunto de

pasos. Un servicio puede incluso encapsular la lógica entera del proceso. En los últimos dos casos, el largo alcance representado por los servicio puede envolver la lógica encapsulada por otros servicios.

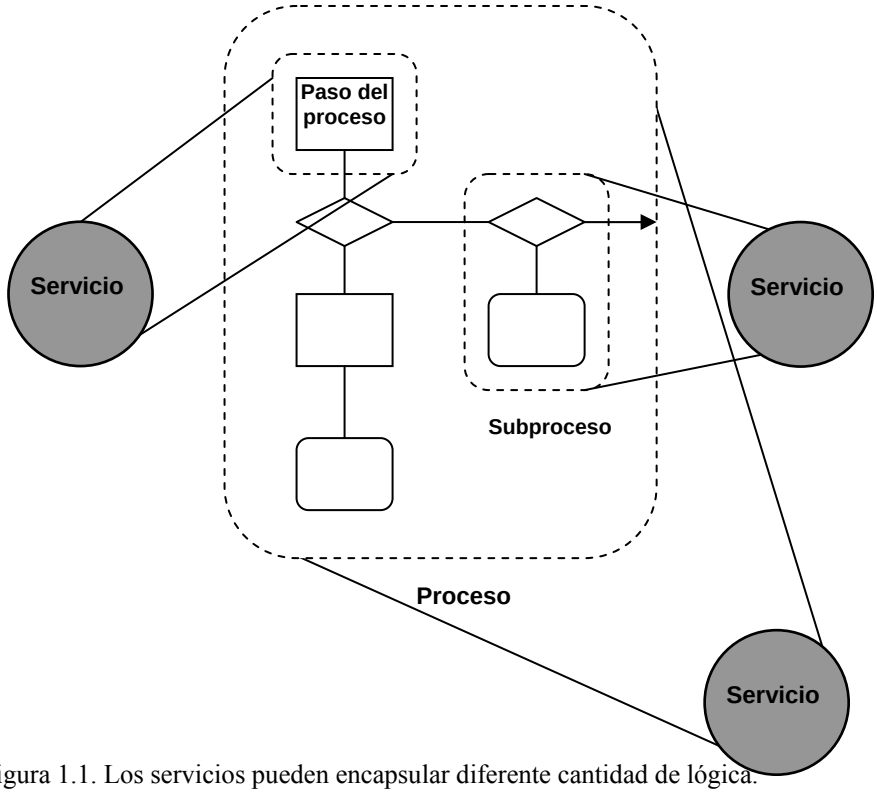


Figura 1.1. Los servicios pueden encapsular diferente cantidad de lógica.
Fuente: Earl (2005).

Para que los servicios utilicen la lógica que encapsulan pueden participar en la ejecución de las actividades del negocio. Para hacer esto, deben formar relaciones distintas con los que deseen utilizarlos.

1.3. La relación entre los servicios

Dentro de SOA, los servicios pueden ser usados por otros servicios u otros programas. No obstante, la relación entre servicios esta basada en un acuerdo que los servicios subscriben para interactuar., estos deben conocer a los demás. Este conocimiento es logrado a través del uso de la descripción del servicio.

Una descripción de servicio en el formato más básico, establece el nombre del servicio y la data esperada y retornada por este. La manera en la cual el servicio usa descripciones de servicio resulta en una relación clasificada como débilmente acoplada. Por ejemplo en la figura 1.2 se muestra a un servicio A que

conoce de un servicio B porque el servicio A está en posesión de la descripción del servicio B.

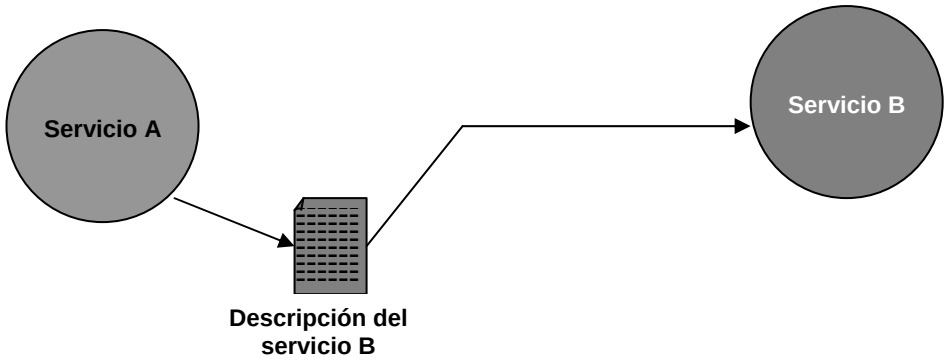


Figura 1.2. Porque este tiene acceso a la descripción del servicio B, el servicio A tiene toda la información necesaria para comunicarse con el servicio B.
Fuente: Earl (2005).

Para que los servicios interactúen y logren algo significativo, estos deben intercambiar información. Por lo tanto se requiere un framework de comunicación capaz de preservar esta relación débilmente acoplada. Un framework así es la mensajería.

1.4. La comunicación entre los servicios

Después de que un servicio envía un mensaje, este pierde el control de lo que pasa con el mensaje después del envío. Es por esto que se requiere que los mensajes sean "unidades comunicación independientes". Esto significa que los mensajes, así como los servicios, deben ser autónomos. Para estos efectos, los mensajes deben estar equipados con suficiente inteligencia para autocontrolar sus partes del procesamiento de la lógica (Figura 1.3).

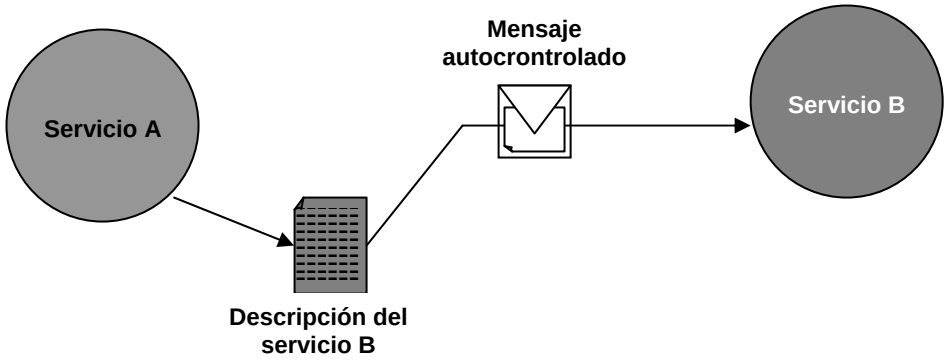


Figura 1.3. Un mensaje existe como una unidad de comunicación independiente.
Fuente: Earl (2005).

Los servicios que proveen descripción y se comunican vía mensajes forman una arquitectura básica, hasta ahora, esta arquitectura parece similar a las arquitecturas distribuidas del pasado que soportan mensajería y una separación de interfaz desde la lógica del proceso. Lo que distingue esta es como estos tres componentes (servicios, descripciones y mensajes) se diseñan. Esto es lo que trae la orientación a servicio.

1.5. El diseño de los servicios

Así como la orientación a objeto, la orientación a servicio se ha convertido en un enfoque de diseño distinto que introduce principios generalmente aceptados que controlan el posicionamiento y el diseño de los componentes. (Figura 1.4).

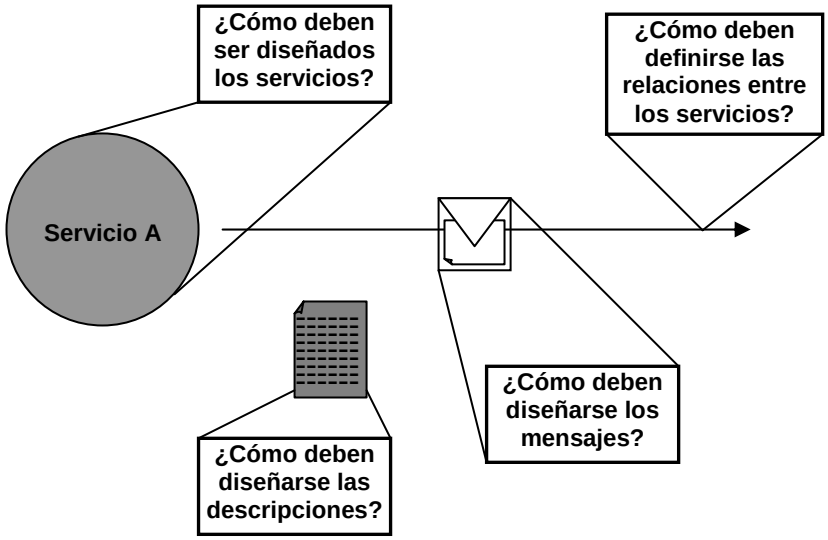


Figura 1.4. Los principios de orientación a servicio rigen las cuestiones de diseño.
Fuente: Earl (2005).

La aplicación de principios de orientación a servicios se explicara luego. Para el propósito de proveer una introducción preliminar, se resaltaran algunos de los aspectos claves de estos principios:

Débil acoplamiento, los servicios mantienen una relación que minimiza las dependencias y solo requieren que estos tengan conocimiento de los otros.

Contrato de servicio, los servicios se adhieren a un contrato de servicio para un acuerdo de comunicación, como se define colectivamente por una o más descripciones de servicios y documentos relacionados.

Autonomía, los servicios tienen el control de la lógica que encapsulan.

Abstracción, más allá de lo que se describe en el contrato de servicio, los servicios esconden lógica para el mundo exterior.

Reusabilidad, la lógica es dividida en servicios con la intención de promover la reutilización.

Composability, las colecciones de servicios pueden ser coordinadas y ensambladas para formar un conglomerado de servicios.

Discoverability, los servicios son diseñados para ser descritos exteriormente de tal forma que puedan ser encontrados y evaluados a través de los mecanismos de descubrimiento disponibles.

Con un conocimiento de los componentes que forman la arquitectura básica y un conjunto de principios de diseño que se pueden usar para modelar y estandarizar estos componentes, todo lo que falta es una plataforma que permita utilizar estas piezas juntas para construir solución de automatización orientada a servicio. La tecnología de los servicios Web ofrece esta posibilidad.

1.6. La construcción de los servicios

Como se ha mencionando recientemente, el termino "orientado a servicio" y varios modelos abstractos de SOA existían antes de arribar los servicios Web. Sin embargo, ningún avance tecnológico ha sido tan adecuado y exitoso para el manifiesto de SOA que los servicios Web.

Todas las plataformas importantes de los principales proveedores soportan la creación de soluciones orientadas a servicio, y la mayoría da por entendido que el soporte de SOA que se provee está basado en el uso de servicios Web. Por lo tanto, mientras que se está en total conocimiento que para lograr una SOA no se requieren servicios Web, la mayoría de los textos se centran en como una SOA puede y debe realizarse a través del uso de la plataforma tecnológica de los Servicios Web.

2. Definición de SOA.

Ahora que se ha finalizado con las características de SOA, se puede culminar con una definición formal.

SOA contemporánea representa una arquitectura abierta, ágil, extensible, federada, con capacidad de composición que está formada por servicios autónomos, con capacidad de QoS, de proveedores diversos, ínter operables, descubribles y potencialmente reusables implementados como servicios Web.

SOA puede establecer una abstracción de la lógica del negocio y la tecnología que puede introducir cambios en el modelado de procesos de negocios y en la arquitectura técnica, resultando un bajo acoplamiento entre estos modelos.

SOA es una evolución de plataformas pasadas, preservando características exitosas de las arquitecturas tradicionales, y que trae distintos principios que fomentan la orientación a servicio para dar soporte a la empresa orientada a servicio.

En toda la empresa se puede colocar SOA como un estándar, pero para lograr este estado se requiere una transición planeada y el soporte por parte de un conjunto de herramientas tecnológicas aún en desarrollo.

Aunque precisa, esta definición de SOA contemporánea en un poco detallada. Para propósitos prácticos, se provee una definición suplementaria que puede ser aplicada tanto a SOA Primitiva como a SOA contemporánea.

SOA es una forma de arquitectura tecnológica que se adhiere a los principios de orientación a servicio. Cuando se realiza a través dela plataforma de los servicios Web, SOA adquiere el potencial de soportar y promover estos principios en todos los procesos de negocios y el dominio de automatización de una empresa.

3. Beneficios de SOA.

Krafzig et al. (2004) establece que el fin primordial de SOA es dotar de agilidad a los sistemas de TI de las organizaciones, y en adición SOA ofrece beneficios a varios niveles, que van desde la reducción de la dependencia tecnológica hasta la simplificación del proceso de desarrollo para incrementar la flexibilidad y la reusabilidad de la infraestructura de negocios.

El objetivo principal de la reusabilidad y la flexibilidad adicional provista por una SOA es una organización ágil, en la cual todos los procesos y servicios son completamente flexibles y puedes ser rápidamente creados, configurados y

reacomodados, según sea requerido por los expertos de negocios sin la necesidad de un staff técnico (ver Figura 3.1).

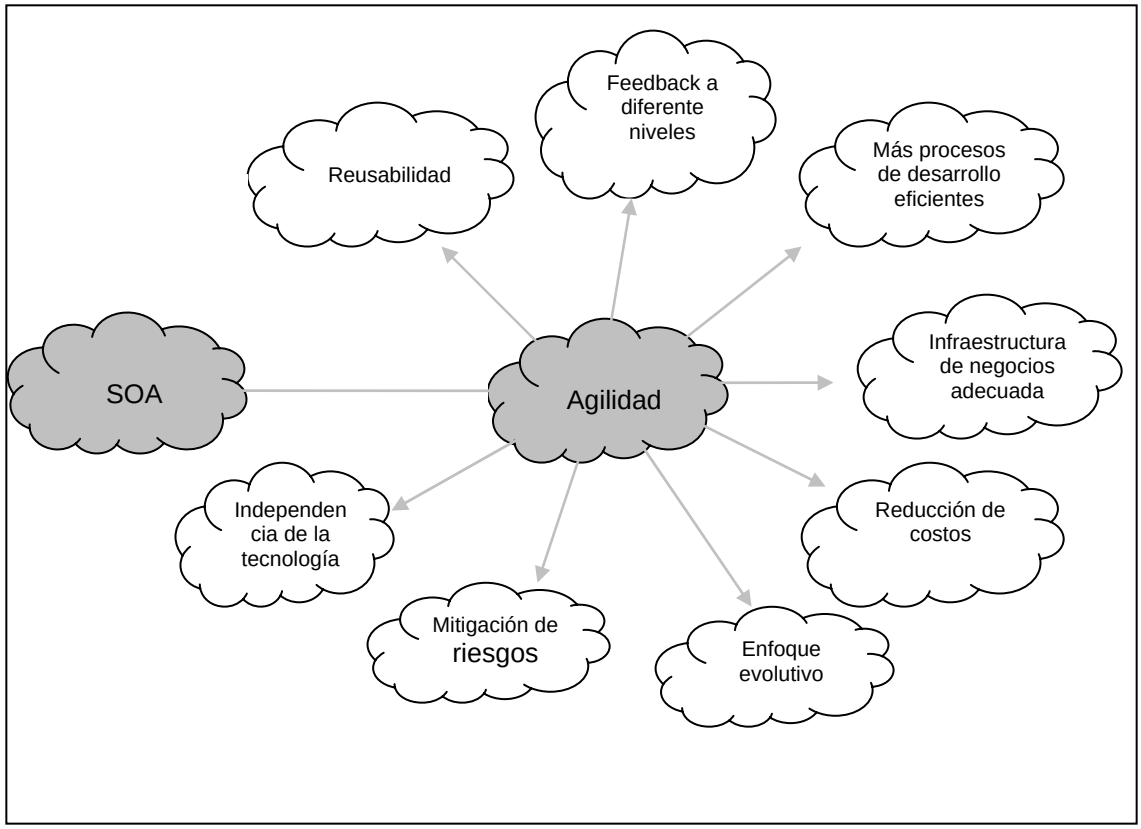


Figura 3.1. Los beneficios de SOA.
Fuente. Krafzig et al. (2004).

4. Construcción de SOA (Planificación y Análisis)

La separación del análisis del negocio, el diseño de la solución y la construcción son bastante distintos en la mayoría de los desarrollos personalizados. Los analistas normalmente obtienen y documentan los requerimientos del negocio que son entregados a los arquitectos y desarrolladores, quienes diseñan y construyen la lógica de automatización correspondiente.

Un proyecto que entrega una solución orientada a servicio contemporánea cambia algo este enfoque al desdibujar las líneas que dividen las fases iniciales del proceso. SOA hace énfasis en una relación directa entre la inteligencia de análisis de negocios y los servicios que terminan representando e implementado la lógica del negocio.

El resultado es el requerimiento de una única forma de análisis que necesita ser completada antes del diseño de servicios individuales. Antes de comenzar lo que se tratará de explicar, ayuda a tomar algunas decisiones acerca de cómo debe ser organizado un proyecto que entrega servicios prometidos por SOA.

4.1. Fases en el ciclo de vida de SOA

El ciclo de vida de un proyecto de desarrollo de SOA simplemente comprende una serie de pasos que se necesitan completar para construir los servicios de una solución orientada a servicio dada.

4.1.1. Fases principales de un ciclo de vida de SOA.

Los proyectos de desarrollo para soluciones orientadas a servicio son, superficialmente, muy parecidos a proyectos de desarrollo para aplicaciones distribuidas. Los servicios Web son diseñados, desarrollados y desplegados junto a los componentes estándar y la selección de soporte usual de las tecnologías backend y frontend. Cuando se indaga un poco más a fondo bajo la capas de la orientación a servicio, sin embargo, se encontrará que para construir y ubicar correctamente los servicios como parte de SOA, los ciclos de los proyectos tradicionales requieren algunos ajustes.

Veamos la figura 4.1, y se puede sorprender como el nombre de las primeras dos fases comienzan con “orientado a servicio” y donde las fases siguientes tienen nombres que comienzan simplemente con servicio. La principal razón porque se hace esta distinción, es porque esto es durante las etapas de análisis y diseño que las características de SOA y los principios de orientación a servicio que se han discutido se incorporan en la solución cuando esta comienza.

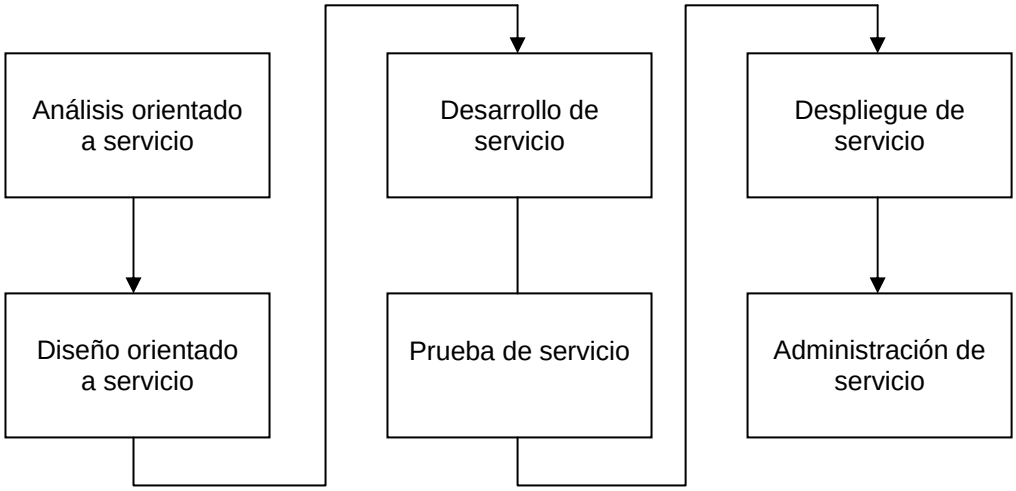


Figura 4.1. Fases comunes en el ciclo de vida en la entrega de SOA.
Fuente: Earl (2005).

Ahora se explicará cada una dichas fases.

4.1.1.1. Análisis orientado a servicio.

Esta es la etapa inicial en donde se determina el alcance potencial de la SOA en cuestión. Las capas de servicio delineadas y los servicios individuales son modelados como servicios candidatos para componer una SOA preliminar.

Un proceso formal de modelado de servicio paso a paso se provee más adelante como parte, de la fase de análisis orientado a servicio.

4.1.1.2 Diseño orientado a servicio.

Cuando se conoce que es lo que se quiere construir, se necesita determinar como será construido. El diseño orientado a servicio es una fase fuertemente

controlada por estándares que incorporan convenciones industriales y principios de orientación a servicio en el proceso de diseño de servicio.

Esta fase, por lo tanto, confronta a los diseñadores de servicio con las decisiones clave que establecen los límites de la lógica dura encapsulada por los servicios. Las capas de servicio diseñadas durantes esta fase incluyen la capa de orquestación, la cual resulta en una definición formal del proceso del negocio.

4.1.1.3. Desarrollo del servicio.

Lo próximo, por supuesto, es la fase de construcción actual. Aquí entran en juego los aspectos de la plataforma de desarrollo. Específicamente, la escogencia del lenguaje y el ambiente de desarrollo determinarán la forma física de los servicios y el proceso de negocios orquestado que toman, de acuerdo con su diseño.

4.1.1.4. Prueba del servicio.

Dada su naturaleza genérica y el potencial para ser reutilizados y compuestos en situaciones imprevisibles, los servicios son exigidos bajo pruebas rigurosas previas al desarrollo en un ambiente de producción.

A continuación una muestra de algunos aspectos claves a tomar en cuenta por los probadores del servicio:

- ¿Qué tipo de demandantes de servicio pueden potencialmente acceder al servicio?
- ¿Se pueden reunir satisfactoriamente todas las afirmaciones de la política del servicio?
- ¿A qué tipos condiciones de excepción estará potencialmente sujeto el servicio?
- ¿Las descripciones revisadas del servicio alteran o amplían versiones anteriores?
- ¿Cómo pueden ser compuestos los servicios de forma fácil?

- ¿Cómo pueden ser descubiertas de forma fácil las descripciones del servicio?
- ¿Se cumple con los perfiles WS-I requeridos?
- ¿Qué aspectos relacionados con los tipos de datos se pueden mostrar?
- ¿Han sido exteriorizadas todas las posibles actividades y composiciones del servicio?
- ¿Han sido probadas completamente todos los procesos de compensación?
- ¿Qué pasa si las excepciones ocurren dentro de los procesos de compensación?
- ¿Todos los servicios nuevos cumplen con los estándares de diseño existente?
- ¿Los nuevos servicios introducen encabezados SOAP configurables? Y, si es así, ¿Son capaces todos los demandantes potenciales (incluyendo los intermediarios) de comprender y procesar estos encabezados?
- ¿Los nuevos servicios introducen requerimientos funcionales o de QoS que la arquitectura actual no soporta?

4.1.1.5. Despliegue del servicio.

La etapa de implementación trae consigo la alegría de instalar y configurar los componentes distribuidos, interfaces de servicio, y cualquiera de los productos middleware asociados dentro de servidores de producción.

- Los aspectos comunes que surgen durante está fase incluyen:
- ¿Cómo se pueden distribuir los servicios?
- ¿Es la infraestructura adecuada para cumplir los requerimientos de procesamiento de todos los servicios?
- ¿Cómo afectará la introducción de nuevos servicios los existentes y las aplicaciones?
- ¿Cómo deben ser posicionados y desplegados los servicios utilizados por múltiples soluciones?
- ¿Cómo afectará la introducción de cualquier middleware requerido al ambiente existente?
- ¿Introducen estos nuevos servicios nuevas versiones de descripciones que necesitarán ser desplegadas a través de versiones existentes?
- ¿Qué configuraciones de seguridad y cuentas se requieren?
- ¿Cómo se mantendrán los conjuntos de servicios para adecuarse a los requerimientos planeados o imprevistos?
- ¿Cómo se hará mantenimiento y monitoreo a los sistemas legados con las limitaciones de desempeño y confiabilidad?

Nota:

El despliegue del servicio es específico a la plataforma de tecnología para la cual los servicios se desarrollan.

4.1.1.6. Administración de servicio.

Después que los servicios se desarrollan, los aspectos del manejo de aplicaciones estándar se convierten en el primer plano. Estos son similares en su naturaleza a los aspectos de administración para las aplicaciones distribuidas basadas en componentes, excepto que estos también se pueden aplicar a los servicios de manera general (en comparación con los servicios que pertenecen a un ambiente de aplicación específico).

Los aspectos que se incluyen frecuentemente son:

- ¿Cómo será monitoreado el uso del servicio?
- ¿Qué forma de control de versiones será usada para manejar los documentos de descripción del servicio?
- ¿Cómo serán rastreados y manejados los mensajes?
- ¿Cómo serán detectados los cuellos de botella de desempeño?

Nota:

La prueba y la administración de los servicios están más allá del alcance de este contenido.

Las etapas del ciclo de vida identificadas previamente representan una vía simple y secuencial para construir servicios individuales.

Ahora se necesita organizar estas etapas en un proceso que pueda:

- Ajustar nuestras preferencias en relación a que tipo de capas de servicio se quieren entregar.
- Coordinar la entrega de servicios de aplicación, negocio y proceso.
- Dar soporte a una transición hacia una SOA estandarizada mientras que nos ayuda a cumplir los requerimientos específicos del proyecto de forma inmediata.

El último ítem en está lista presenta un gran desafío. El éxito de SOA dentro de una empresa generalmente es dependiente del grado de estandarización de esta, cuando esta ha sido planificada en fases dentro de dominios de negocio y de aplicación. Sin embargo, el éxito de un proyecto para entregar una solución orientada a servicio generalmente se mide por el grado en que la solución cumple con los requerimientos esperados dentro de un contrato y un tiempo estimado.

Para tratar este problema, se necesita una estrategia. Esta estrategia debe estar basada en las prioridades de la organización, para establecer el balance correcto entre la entrega de las metas de migración a largo plazo y el cumplimiento de requerimientos a corto plazo. Estas estrategias comunes han emergido, cada una tratando este problema de diferente manera.

- De arriba hacia abajo (top-bottom).
- De abajo hacia arriba (bottom-up).
- Ágil (meet-in-the-middle)

Estas vías difieren en consideraciones de prioridad y prácticas. Las siguientes tres secciones proveen descripciones de procesos y exploran los pros y los contras de cada enfoque.

La manera como se enfoque la creación de un ambiente orientado a servicio determinará en última instancia lo que se obtendrá al final. Las estrategias aquí discutidas, por consiguiente, presentaran una confrontación en la toma de algunas decisiones importantes. Escoger el enfoque correcto determinará el grado para el cual el modelo orientado a servicio y los esfuerzos de diseño pueden aprovechar todo el potencial de SOA.

4.2. Estrategias para el desarrollo de SOA.

Las fases comunes de los proyectos asociadas con la construcción y la entrega de servicios se organizan en diferentes secuencias basadas en prioridades organizacionales, lo que constituye las diferentes estrategias para el desarrollo de SOA.

4.2.1 La estrategia de arriba hacia abajo (top-down).

Esta estrategia es un principalmente un enfoque "analizar primero" que no solo requiere que los procesos del negocio se conviertan en orientados a servicio, sino que también promueven la creación (o reorganización) de todo el modelo de negocios de una organización. Por consiguiente este proceso está atado o se deriva de una lógica de negocios existente en la organización.

4.2.1.1 El proceso

El enfoque de arriba hacia abajo normalmente contendrá algunos o todos los pasos ilustrados en la figura 4.2. Note que este proceso asume que los requerimientos del negocio ya han sido recolectados y definidos.

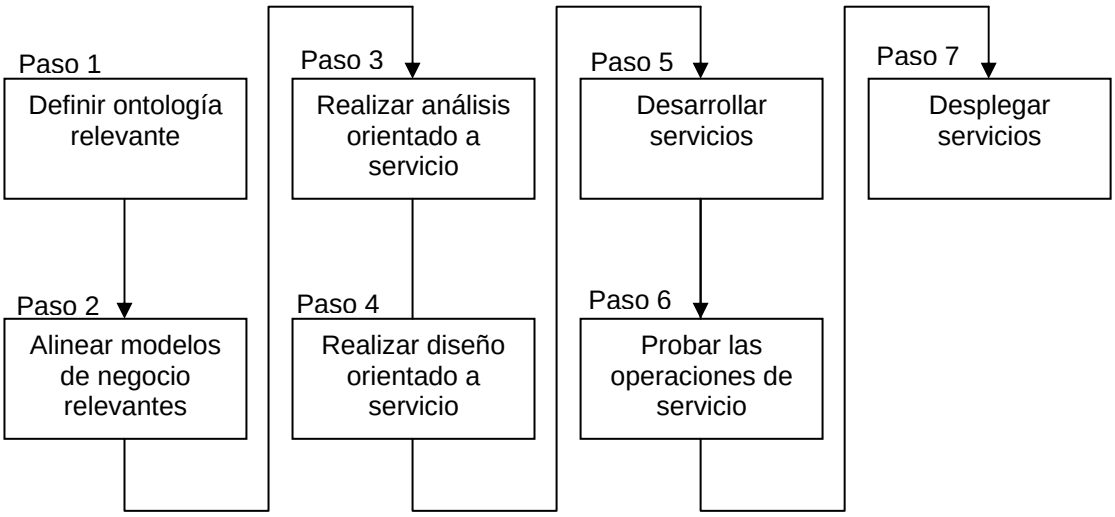


Figura 4.2. Pasos comunes en el proceso de la estrategia top-down.
Fuente: Earl (2005).

Paso 1: Definir ontología relevante en toda la empresa.

Parte de lo que establece una ontología es una clasificación de sistemas de información procesados por una organización. Esto resulta en un vocabulario común, como también una definición de cómo estos sistemas de información se relacionan unos con otros. Las organizaciones grandes con múltiples áreas de negocio pueden tener varias ontologías, cada una rigiendo en una división de

negocios específica. Se espera que estas ontologías especializadas se alineen para dar soporte a una ontología en toda la empresa.

Si tal vocabulario de negocios no existe aun para cualquier sistema de información se requiere una solución para trabajar con este, entonces este paso requiere que esto este definido. Por consiguiente se puede requerir una cantidad significativa de recolección de información por adelantado y esfuerzo en el análisis de negocios a alto nivel.

Paso 2: Alinear los modelos de negocios relevantes (incluyendo modelos de entidad) con ontologías nuevas o existentes.

Después que la ontología se establece, los modelos de negocio existentes necesitan ser ajustados (o incluso creados) para representar propiamente el vocabulario provisto por la ontología en términos de modelado de negocios. Los modelos de entidad particularmente son de importancia, pues pueden ser utilizados más adelante como la base para los servicios de negocio centrados en entidad.

Nota:

Aunque se relacionan con análisis, los pasos 1 y 2 se colocan aquí más como un prerrequisito para la fase de análisis orientado a servicio que se ha definido.

Paso 3. Realizar análisis orientado a servicio.

Una fase de análisis orientado a servicio, tal como se describe de forma detallada más adelante.

Paso 4. Realizar un diseño orientado a servicio.

Las capas de servicio se definen formalmente como parte del proceso del diseño orientado a servicio, tal como se describe posteriormente como un tema aparte.

Paso 5. Desarrollar los servicios requeridos.

Los servicios se desarrollan de acuerdo a sus respectivas especificaciones de diseño y las descripciones de servicios creadas en el paso 4.

Paso 6. Probar los servicios y todas las operaciones de los servicios.

La fase de prueba requiere que todas las operaciones de los servicios experimentan necesariamente chequeos de aseguramiento de calidad. Esto normalmente excede la cantidad de pruebas requeridas para la lógica de automatización que se implementa debido a que los servicios reutilizables probablemente estén sujetos a pruebas más allá del alcance inmediato de la solución.

Paso 7. Desplegar los servicios.

La solución finalmente es desplegada en la producción. Una consideración de implementación más allá de esas que fueron identificadas originalmente, como parte de este paso, es el futuro potencial de reutilización del servicio. Para facilitar múltiples solicitantes de servicio, los servicios altamente reutilizables pueden requerir un poder de procesamiento extra y pueden tener requerimientos especiales de seguridad y accesibilidad que necesitaran ser resueltos.

4.2.1.2. Los pros y los contras.

El enfoque top-down para construir SOA generalmente resulta en una arquitectura de servicio de alta calidad. El diseño y los parámetros alrededor de cada servicio analizados a fondo, maximizan el potencial de reutilización y las oportunidades para una composición adecuada. Todo esto establece una base para una empresa estandarizada y federada donde los servicios mantengan un estado de adaptabilidad, mientras de continua unificando la heterogeneidad existente.

Los obstáculos a superar en un enfoque top-down usualmente están asociados con tiempo y dinero. A las organizaciones se les requiere una inversión significativa en proyectos de análisis por adelantado que pueden tomar una gran cantidad de tiempo (proporcional al tamaño de la organización y la solución inmediata), sin mostrar un resultado inmediato.

4.2.2. La estrategia de abajo hacia arriba (bottom-up).

Este enfoque esencialmente impulsa la creación de servicios como una manera de cumplir los requisitos centrados en la aplicación. Los servicios Web se construyen en una base "como se necesitan" y se modelan para encapsular lógica de aplicación y de esta manera cumplir con los requerimientos de la solución. La integración es el principal motivador para el diseño bottom-up, donde la necesidad de tomar ventaja de un framework de comunicaciones abierto SOAP puede encontrarse simplemente agregando servicios como envoltorios para sistemas legados.

4.2.2.1. El proceso.

Un enfoque bottom-up típico sigue un proceso similar al explicado en la figura 4.3. Note que este proceso asume los requerimientos del negocios ya han sido recolectados y definidos.

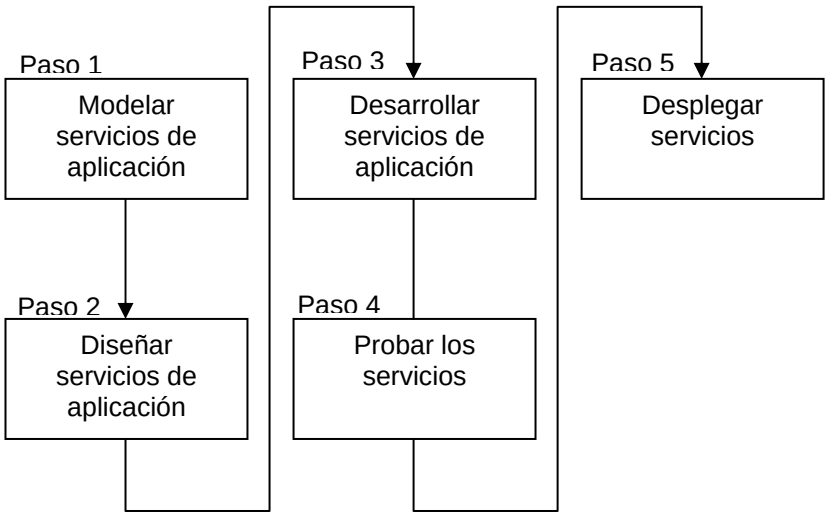


Figura 4.3. Pasos en el proceso de la estrategia bottom-up.
Fuente: Earl (2005).

Paso 1. Modelar los servicios de aplicación requerida.

Este paso resulta en la definición de los requerimientos de aplicación que pueden ser cumplidos a través del uso de servicios Web. Los requerimientos típicos requieren incluir la necesidad de establecer canales de integración punto a

punto entre los sistemas legados o en soluciones B2B. Otros requerimientos comunes surgen del deseo de reemplazar la tecnología de comunicaciones remota con el framework de comunicaciones de mensajería SOAP.

Para las soluciones que emplean la estrategia bottom-up, para entregar soluciones altamente centradas en servicio, los servicios de aplicación también deben ser modelados para incluir reglas y lógica de negocios específicas. En este caso, es probable que dos capas de servicios de aplicaciones emerjan, compuestas de servicios híbridos y de utilidad (utility). Estos servicios clasificados como reutilizables pueden actuar como terminales de aplicaciones genéricas para propósitos de integración, o estos pueden estar compuestos por sistemas híbridos padres.

Paso 2. Diseñar servicios de aplicación requeridos.

Algunos de los servicios de aplicación modelados en el paso 1 pueden ser entregados a través de compras o alquiler de servicios de envoltorio de terceros o quizás por medio de la creación de servicios Proxy autogenerado. Estos servicios pueden proveer una pequeña oportunidad para el diseño adicional. Los servicios de aplicación configurables, si bien, necesitarán experimentar un proceso de diseño en donde los estándares de diseño existentes se aplican para asegurar un nivel de consistencia.

Paso 3. Desarrollar los servicios de aplicación requeridos.

Los servicios de aplicación se desarrollan de acuerdo a sus respectivas descripciones de servicio y especificaciones de diseño aplicables.

Paso 4. Probar los servicios.

Los servicios, sus ambientes de solución asociados, y la lógica legada subyacente son probados para asegurar que los requerimientos de procesamiento puedan ser cubiertos. Las medidas de desempeño y las pruebas de stress a menudo son usadas para establecer los parámetros del proceso de sistemas legados expuestos vía servicios de envoltorio. La prueba de seguridad también es una parte importante en esta etapa.

Paso 5. Desplegar el servicio.

La solución y sus servicios de aplicación son desplegados en producción. Las consideraciones de implementación para los servicios de aplicación frecuentemente incluyen requerimientos de desempeño y seguridad.

4.2.2.2. Los pros y los contras.

La mayoría de las organizaciones que actualmente construyen servicios Web aplican el enfoque bottom-up. La razón primaria detrás de esto es que las organizaciones simplemente agregan servicios Web a sus ambientes de aplicación existentes para aprovechar el conjunto de tecnología de los servicios Web. La arquitectura dentro de la cual los servicios Web se agregan queda sin cambios, y por consiguiente los principios de orientación a servicio son poco considerados.

Cómo resultado, el termino que se usa para referirse a este enfoque "estrategia bottom-up" es algo mal nombrado. La estrategia de bottom-up en realidad no es estrategia del todo. Ni tampoco es un enfoque válido para lograr una SOA contemporánea. Esta es una práctica con la que muchas organizaciones chocaran cuando comiencen a tomar la orientación a servicio, como un modelo arquitectónico, de forma más seria. Aunque el diseño botton-up permite la creación eficiente de los servicios Web como se requieren por las aplicaciones, implementar una SOA correcta en ultima instancia puede resultar en una gran cantidad de reajustes o incluso la introducción de nuevas capas de servicio estandarizadas colocadas sobre servicios no estandarizados producidos por este enfoque.

4.2.3. La estrategia ágil.

El desafío que queda es encontrar un balance aceptable en la incorporación de principios de diseño orientado a servicio en ambientes de análisis de negocios, sin tener que esperar a la integración de la tecnología de servicios dentro de los ambientes técnicos. Para muchas organizaciones, por consiguiente, es útil ver estos dos enfoques como extremos y encontrar un término medio aceptable.

Esto es posible definiendo un nuevo proceso que permita para el análisis a nivel de negocio que ocurra concurrentemente con el diseño y desarrollo del servicio. También conocido como en el enfoque “meet in the middle”, la estrategia ágil es más compleja que las dos anteriores porqué esta necesita cumplir con dos conjuntos de requerimientos que chocan.

4.2.3.1. El proceso.

Los pasos del proceso mostrados en la figura 4.4 demuestran un ejemplo de cómo una estrategia ágil puede ser utilizada para alcanzar las metas respectivas de los enfoques top-down y bottom-up.

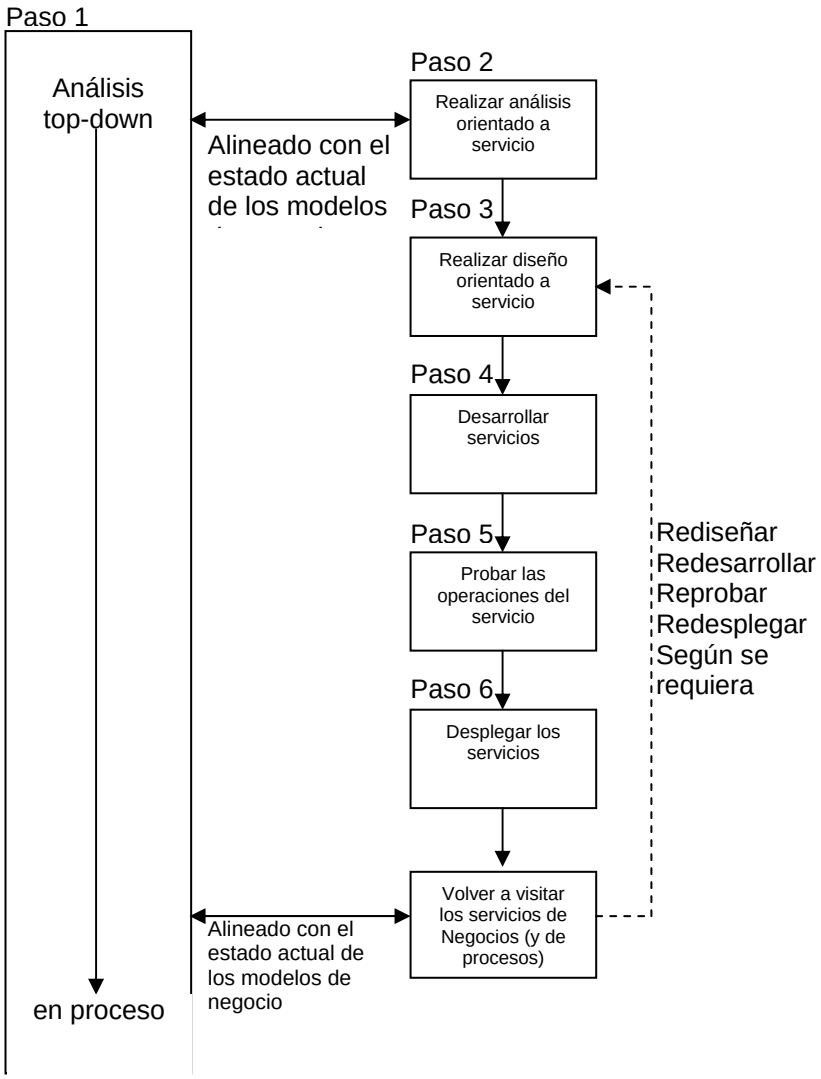


Figura 4.4. Un ejemplo del proceso de estrategia ágil
Fuente: Earl (2005)

Paso 1: Comenzar el análisis top-down, centrándose primero en las partes claves de ontología y entidades de negocio relacionadas.

El análisis top-down estándar comienza pero con un pequeño foco. Las partes de los modelos de negocio, relacionadas directamente a la lógica del negocio comienzan automáticamente a recibir prioridad inmediata.

Paso 2: Cuando el análisis top-down ha progresado suficientemente, se realizar el análisis orientado a servicio.

Mientras que el paso1 está aún en progreso, este paso inicia una fase de análisis orientado a servicio. Dependiendo de la magnitud del análisis requerido para completar el paso 1, es conveniente dar una buena ventaja a ese paso. Mientras mas se profundice en estos procesos, más cantidad de diseños de servicio se beneficiarán.

Después que el análisis top-down ha progresado suficientemente, los servicios del modelo de negocio para representar mejor el modelo de negocio con cualquier resultado del análisis, ya está disponible. Este es un punto clave de decisión en este proceso. Esto puede requerir de un juicio equilibrado para determinar, cuando el análisis top-down en proceso está suficientemente maduro para proceder con la creación de los modelos de servicios de negocio. Esta consideración debe pesar entonces, contra la importancia y urgencia de los requerimientos pendientes del proyecto.

Paso 3: Realizar el diseño orientado a servicio.

Se definen las capas de servicio seleccionadas, y son diseñados los servicios individuales como parte del proceso de diseño orientado a servicio, como se describe en la sección dedicada al diseño.

Paso 4, 5 y 6: Desarrollar, probar y desplegar los servicios.

Desarrollar los servicios y dejarlos listos para los procedimientos de pruebas y desarrollo.

Paso 7: A medida que continua el progreso del análisis top-down, revisitar los servicios de negocio.

Realizar revisiones periodicas de todos los servicios de negocio para comparar su diseño contra el estado actual de los modelos de negocio. Realizar una nota de discrepancia y programar un rediseño para estos servicios fuera de alineación. Esto normalmente requerirá una extensión a un servicio existente, para que este provea la mejor gama de capacidades requeridas. Cuando un servicio se rediseña, este necesitará de nuevos pasos de desarrollo, pruebas y despliegue estándar

Para preservar la integridad de los servicios producidos bajo este enfoque, el concepto de *contratos de servicios inmutables* necesita ser implementado de forma estricta. Después que se publica un contrato, este no puede ser alterado. A menos que las revisiones hechas a los servicios, den lugar a extensiones que no impongan ninguna restricción a un contrato existente (tales como la adición de nuevas operaciones para una definición WSDL), el paso 7 de este proceso probablemente resultará en la necesidad de publicar nuevas versiones del contrato y el requerimiento de un sistema de manejo de versiones.

4.2.3.2. Los pros y los contras.

Esta estrategia toma lo mejor de ambos mundos y combina esto dentro de un enfoque para realizar SOA, que cumpla los requerimientos inmediatos, sin arriesgar la integridad del modelo de negocio de la organización y las cualidades de orientación a servicio de la arquitectura.

Mientras esta reúne ambas, pequeñas y grandes necesidades, el resultado neto de emplear esta estrategia, incrementa el esfuerzo asociado con la entrega de cada servicio. El hecho que los servicios puedan necesitar ser revisitados,

rediseñados, redesarrollados y redesplegados aumentará proporcionalmente a la cantidad de servicios sujetos a esta paso de retrabajo.

Adicionalmente, este enfoque impone tareas de mantenimiento, que se requieren para asegurar que los servicios existentes se mantengan en concordancia con los modelos de negocio revisados. Incluso, con un proceso de manteniendo in situ, los servicios aún corren el riesgo de desviación, con respecto a cambios constantes del modelo de negocio.

5. Análisis orientado a servicio (introducción).

El primer y quizás más importante paso a través del ciclo de vida de SOA, es un análisis de lo que consistirán exactamente los servicios de nuestra SOA. En lo subsiguiente se realiza una introducción de los aspectos fundamentales de esta fase del proyecto, para establecer un proceso de modelado de servicio formal y resaltar una serie de aspectos, que necesitan ser tratados antes de proceder a la etapa de diseño.

5.1. Introducción al análisis orientado a servicio.

El proceso de determinar ¿cómo se pueden representar los requerimientos de automatización de negocio a través de orientación a servicio?, es el dominio del análisis orientado a servicio.

5.1.1. Objetivos del análisis orientado a servicio.

Las preguntas principales hechas durante esta fase son:

- ¿Qué servicios necesitan ser construidos?
- ¿Qué lógica necesita ser encapsulada por cada servicio?

El grado al cual se respondan estas preguntas está directamente relacionado con la magnitud del esfuerzo invertido en el análisis. Mucho de los aspectos que se han discutido en las secciones anteriores puede ser parte de esta etapa. Específicamente, la determinación de que capas de servicio construir y como enfocar su entrega, son puntos de decisiones críticos que terminarán formando la estructura de todo el ambiente orientado a servicio.

Nota:

Mientras la selección de la estrategia de entrega de SOA pueda ser agregada como un paso separado dentro de la fase de análisis orientado a servicio, nuestra suposición en esta sección es que esta decisión ya ha sido tomada.

Todas las metas de realizar análisis orientado a servicio son las siguientes:

- Definir un conjunto preliminar de operaciones de los servicios correspondientes.
- Agrupar operaciones de servicio candidatas en contextos lógicos. Estos contextos representan los servicios candidatos.
- Definir los límites preliminares de los servicios, de tal manera que estos no se solapen con cualquier servicio existente o planeado.
- Identificar la lógica encapsulada con reutilización potencial.
- Asegurar que el contexto de lógica encapsulada es apropiado para su uso pretendido.
- Definir cualquiera de los modelos de composición preliminares conocidos.

5.1.2. El proceso de análisis orientado a servicio.

Introducir un nuevo proceso de análisis en un ambiente de TI existente puede ser una cuestión delicada. Cada organización ha desarrollado su propio enfoque para analizar problemas y soluciones de automatización de negocios, años de esfuerzo y documentación ya han sido invertidos en procesos bien establecidos y entregables de modelado. El proceso descrito en esta sección no tiene la intención de suplantar procesos existentes. En lugar de eso, se propone una secuencia de pasos complementarios, específicamente para la entrega de una solución orientada a servicio.

El análisis orientado a servicio puede ser aplicado a diferentes niveles, dependiendo en cuales de las estrategias de entrega de SOA se utilizan para producir servicios. Como se explicaba anteriormente, la escogencia de una estrategia determinará las capas de abstracción que comprenden las capas de servicio de un ambiente se solución.

Desde una perspectiva de análisis, cada capa tiene diferentes requerimientos de modelado. Por ejemplo, la naturaleza del análisis requerido, para definir servicios de aplicación es diferente de la que se necesita para modelar las capas de servicio de negocios.

Por lo tanto, como se mencionó previamente, un prerrequisito clave de este proceso es la selección de una estrategia de entrega de SOA. Otras preguntas que deben ser respondidas previamente para proceder con el análisis orientado a servicio incluyen:

- ¿Qué trabajo excepcional se necesita para establecer el modelo o los modelos de negocio requeridos y la ontología?
- ¿Que herramientas de modelado serán utilizadas para llevar a cabo el análisis?
- ¿El análisis será parte de un plan de transición de SOA?

Nótese que la respuesta a la última pregunta, a menudo dependerá del alcance del proyecto. El análisis puede ser una fase programada dentro de un plan más grande que guíe la transición de toda la organización hacia SOA. O en proyectos más pequeños, el propio análisis orientado a servicio, puede incorporar un paso separado para el plan de transición. Crear un plan de transición es un asunto que está, por lo tanto, fuera del alcance de este material.

El proceso de análisis orientado a servicio es un subproceso dentro de todo el ciclo de vida de la entrega de SOA. Los pasos en la figura 5.1 son tareas comunes asociadas con esta fase y son descritas en las siguientes secciones.

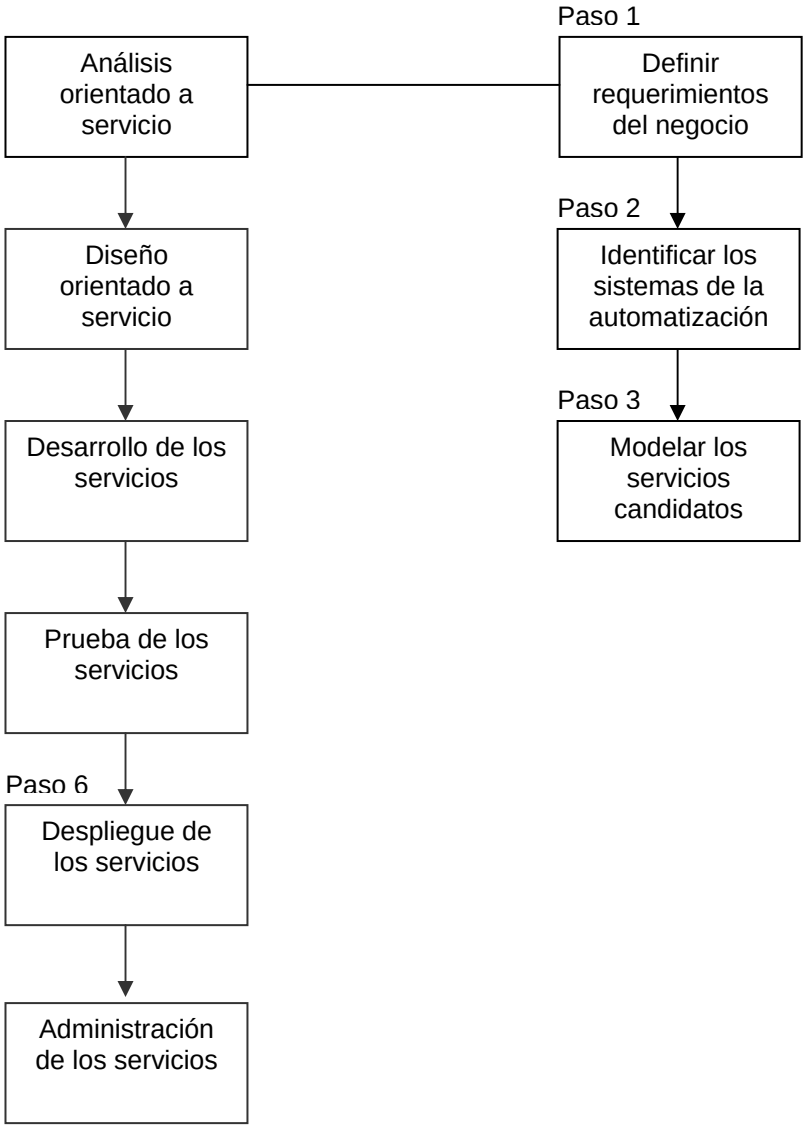


Figura 5.1. Un proceso de alto nivel de análisis orientado a objeto.
Fuente: Earl (2005).

Note que los pasos 1 y 2 esencialmente representan tareas de recolección de información que son llevadas a cabo en preparación para el proceso de modelado descrito en el paso 3.

Paso 1: Definir requerimientos de automatización del negocio.

Los requerimientos del negocio son recolectados de cualquier manera, su documentación se requiere para comenzar el proceso de análisis. Dado que el

alcance del análisis se centra en la creación de servicios como soporte a una solución orientada a servicio, solo los requerimientos relacionados al alcance de la solución, son los que deben ser considerados.

Los requerimientos deben estar suficientemente maduros de manera tal, que se pueda definir un proceso de automatización de alto nivel.

Paso 2: Identificar los sistemas de automatización existentes.

Se necesita identificar la lógica de aplicación que ya está, en cualquier magnitud, para automatizar cualquiera de los requerimientos identificados en el paso 1. Mientras que el análisis orientado a servicio no determina como los servicios Web encapsularán o reemplazarán la lógica de aplicación legada, esto nos ayuda a determinar los potenciales sistemas afectados.

Los detalles de cómo se relacionan los servicios Web con los sistemas existentes se esclarecen en la fase de diseño. Por ahora, esta información será usada para ayudar a identificar los servicios de aplicación candidatos durante el proceso de modelado de servicio descritos en el paso 3.

Nótese que este paso está más orientado a soportar los esfuerzos de modelado de una solución orientada a servicio a gran escala. Una comprensión de ambientes legados afectados también es útil, cuando se modela una cantidad más pequeña de servicios, pero no se requerirá una gran cantidad de esfuerzo de investigación.

Paso 3. Modelar servicios candidatos

Un análisis orientado a servicio introduce el concepto de *proceso de modelado de servicio,* para lo cual las operaciones de servicio candidatas son identificadas y entonces agrupadas en un contexto lógico. Estos grupos eventualmente toman forma de servicios candidatos que posteriormente son ensamblados en un modelo de composición tentativo, representando la lógica combinada de la aplicación orientada a servicio planeada.

Este proceso se explica en detalle en la sección de modelado de servicio (proceso paso a paso).

5.2. Beneficios de SOA centrada en negocios.

La llegada de los servicios Web ha publicitado la importancia de un framework de comunicaciones abierto. Como resultado, muchos de los profesionales de IT están en una etapa donde conocen la relevancia de la tecnología de los servicios Web.

Como se ha establecido previamente, la mayoría de los servicios Web que se construyen actualmente, son más o menos una mezcla de servicios de aplicación y de negocio. Estos tipos de servicios híbridos son atractivos porque, con mínimo esfuerzo, cumplen los requisitos inmediatos con beneficios claros del retorno de inversiones. La proliferación de servicios híbridos es el resultado de el enfoque bottom-up que se vuelven muy común. Estos proveen una manera inmediata para que todas las formas pasadas de arquitectura de aplicación tomen parte en framework de comunicaciones de los servicios Web.

Los servicios de negocio, por otro lado, a menudo no necesitan justificación. A pesar de todo, muchos adversarios aun no estan concientes de los beneficios de introducir los principios de orientación a servicio en el dominio del análisis de negocios. Es fácil ignorar el modelado de negocios orientado a servicios y simplemente centrarse en la orientación a servicios de como se aplica la tecnología y la arquitectura técnica. La razón común para este enfoque es que cualquier proceso de negocio necesita ser automatizado y dividido en servicios Web físicos según se requiera.

Muchas de las características de SOA contemporánea que se nombraron anteriormente, pueden lograrse aún sin el uso de servicios de negocio. Puede parecer que usted mismo se está ahorrando una carga de trabajo por tomar esta vía, sin embargo los beneficios aparentes son superficiales. Hay muy buenas razones para tomar el tiempo de modelar y construir servicios de negocios. En

esta sección se listan una serie de beneficios de incorporar orientación a servicio, dentro del nivel de los procesos del negocio.

5.2.1 Los servicios de negocio construyen agilidad en modelos del negocio.

La orientación a servicio le da a los modelos del proceso de negocio una estructura que pueden mejorar significativamente la flexibilidad y la agilidad, con lo cual los proceso pueden ser remodelados en respuesta a los cambios. Cuando se diseñan correctamente, los servicios de negocio pueden establecer un ambiente tecnología de información sensible, donde los cambios en las áreas de negocio de la organización puedan ser resueltos eficientemente a través de la recomposición, tanto del proceso de negocio como la arquitectura de tecnología que lo soporta (como se expresa en la capa de servicios de aplicación).

Así como las SOA son manejadas por los negocios, hay restricciones del mundo real (infraestructura, restricciones de seguridad, restricciones de presupuesto) que requieren que la tecnología sea colocada detrás. Esto puede cambiar la carga de adaptación sobre los modelos del proceso de negocio. Este tipo de requerimiento de agilidad puede ser cubierto por la capa de servicio de negocios, como permite a los servicios de negocio adecuarse a los cambios de requerimiento, originados de los ambientes técnicos de la organización.

En otras palabras, aplicando abstracción a ambos lados, en la tecnología y en los negocios, se establece el potencial para que una empresa logre una forma de agilidad de dos vías (figura 5.2).

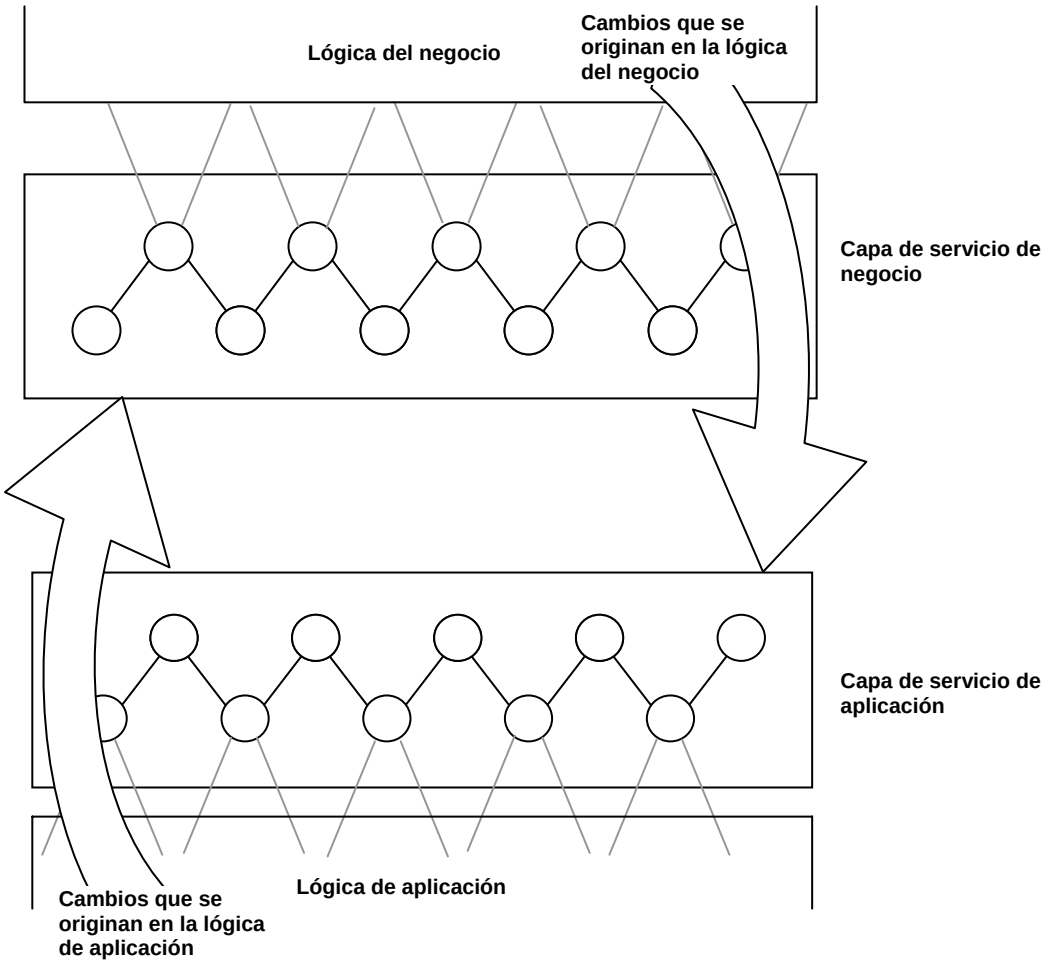


Figura 5.2. Cambios originados en el domino de la lógica de negocios son acomodados por el dominio de la lógica de aplicación y viceversa.
Fuente: Earl (2005).

5.2.2. Los servicios de proceso preparan un proceso para la orquestación.

Si pronto se moverá o no, hacia un ambiente orientado a servicio basado en orquestación, se estará incrementando la importancia de estar listo para esta transición. La orquestación trae consigo conceptos, que cuando se implementan, descansan en la base de SOA. Por lo tanto, modelar los procesos actuales a fin de que estos puedan eventualmente ser más fácilmente migrados hacia un ambiente manejado por orquestación, es lo que se recomienda.

5.2.3. Los servicios de negocio activan la reutilización.

La creación de una capa de servicios de negocio promueve la reutilización en ambos servicios, de aplicación y de negocios, como sigue:

Al modelar la lógica de negocio como servicios distintos con límites explícitos, la reutilización al nivel de los procesos de negocio se puede lograr. La lógica atómica de subprocesos o incluso los procesos enteros pueden se reusados como parte de otra lógica de proceso o como parte de un proceso de composición.

Al tomarse el tiempo para alinear adecuadamente los modelos de negocio con la representación de los servicios de negocio, la capa de servicio de negocio resultante termina liberando toda la capa de servicio de aplicación, de asumir tareas específicas o funciones de procesamiento de actividades. Esto permite a los servicios de aplicación ser posicionados y evolucionar hacia servicios puros, servicios de utilidad reusables que faciliten los servicios de negocio a través de los límites de la solución.

5.2.4. Sólo los servicios de negocio pueden realizar la empresa orientada a servicio.

El modelado de los servicios de negocios casa los principios de orientación a servicio con el modelo de negocios de la organización. La perspectiva resultante puede expresar claramente como los servicios se relacionan con e incorporan el cumplimiento de los requisitos del negocio.

Aplicar servicios de negocio fuerza a una organización a ver e interpretar el conocimiento del negocio en una manera orientada a servicio. Cambiar la perspectiva de cómo los procesos de negocio puede ser estructurada, dividida y modelada es un paso esencial para lograr un ambiente en el cual una orientación a servicio es estandarizada, exitosamente consistente y naturalmente común.

Aunque la capa de servicios de negocio puede representar acertadamente un modelo de negocio corporativo sobre la implementación, esto puede estar obsoleto una vez que emergen requerimientos nuevos y revisados. Siempre y

cuando esta se mantenga relativamente alineada con el estado actual de los modelos de negocio, esta continuará sirviendo como una vista valiosa de la empresa, valiosa porque esta no existe en abstracto sino en una forma implementada y operacional.

5.3. Derivar servicios de negocio.

Así como no existe una definición estándar a nivel industrial de SOA y así como no han sido totalmente estandarizados los principios de orientación a servicio, hay también significados no estandarizados de modelado de servicios de negocio. Como con todos los aspectos de SOA, existen opiniones de sobra, y sin embargo aunque muchos tienen ideas, unas pocas metodologías han emergido. En lugar de eso, hay un selecto grupo de enfoques, algunos de los cuales han sido más aceptados que otros.

Quizás no debe haber un solo enfoque para derivar servicios. No es inusual para el modelo de negocio detrás de una empresa típica, tener que experimentar cientos de revisiones, conformado a través de años de adaptación al clima de negocios circundante de la organización. Las organizaciones emplean diferentes metodologías, relaciones de entidades de negocio, y vocabularios, resultando en estructuras del modelo de negocio ampliamente divergentes. Por otro lado, hay preferencias e influencias culturales de la plataforma del proveedor que resultan en expresión de los modelos de negocios a través de diferentes conjuntos herramientas y lenguajes de modelado.

Al fin de cuentas es que cada modelo de negocios es único, por consiguiente, el análisis por adelantado no se puede evitar, para derivar servicios de negocio que representan lo mejor posible una organización como una entidad de negocio cohesiva.

5.3.1. Fuentes de las cuales los servicios de negocio se pueden derivar.

El funcionamiento interno de una organización, independientemente de su estructura o talla, puede se descompuesta en una colección de servicios de negocio. Esto es porque un servicio de negocio simplemente representa una unidad lógica de trabajo, y cualquier cosa que haga cualquier organización consiste de unidades de trabajo.

La diferencia, aun, es como la organización estructura y documenta el trabajo realizado. Al comienzo de esta sección se insistía en el hecho que cada ambiente corporativo es único en la forma y tamaño de sus modelos de negocios y como los implementa y mantiene. Esto por lo tanto advierte al analista conocedor de orientación a servicio, para determinar la mejor manera de traducir la lógica existente a servicios.

Debajo hay algunos ejemplos de enfoques de análisis de negocio comunes usados por muchas organizaciones. Para cada uno se realiza una breve discusión de cómo pueden ser derivados los servicios.

Modelos de manejo de procesos de negocio. *Bussines Process Management* (BPM).

La llegada de BMP ha resultado en una conmoción en todo el ámbito empresarial de la actividad de modelado y remodelado de procesos. Por lo tanto, los modelos de proceso se han convertido en una forma central de la documentación del análisis de negocios en muchas organizaciones. Los servicios de negocio se pueden derivar de la lógica del proceso del workflow.

Anteriormente se estableció como el alcance de servicio de negocio puede variar. Específicamente, se discutió como un servicio puede representar un paso dentro de un proceso, un subproceso parte de un proceso más grande o incluso un proceso entero (figura 5.3).

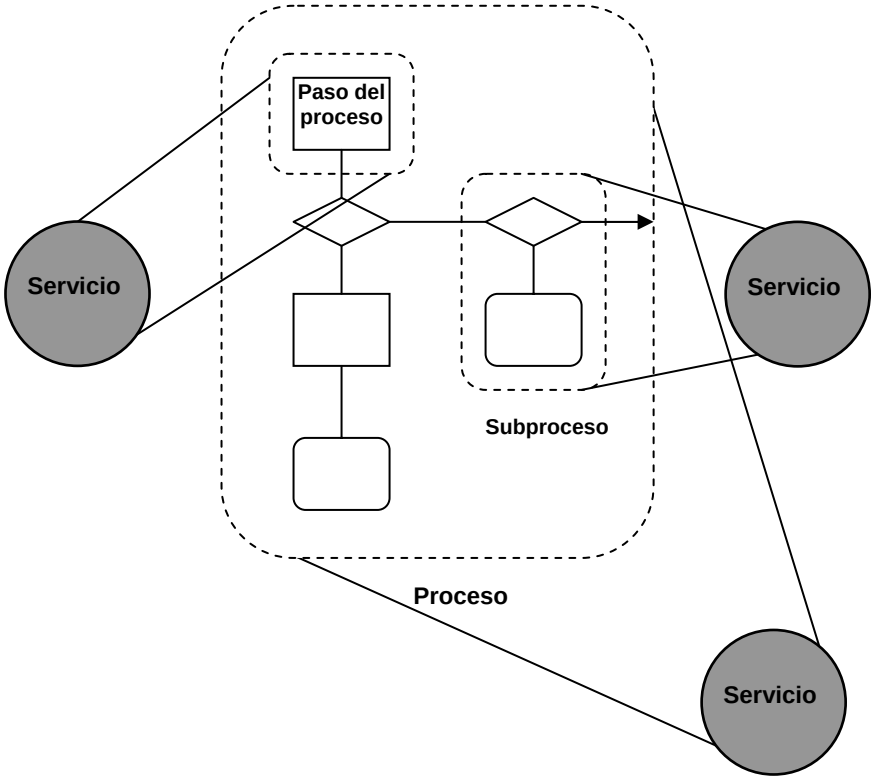


Figura 5.3. Partes de un proceso que pueden ser encapsuladas por un servicio de negocio. Fuente: Earl (2005).

Derivar un servicio de negocio de un proceso de negocio requiere un conocimiento a fondo de la lógica de workflow subyacente. Esto es porque al definir el alcance de la lógica de negocio a ser representada es una llamada de juicio que puede tener implicaciones importantes, cuando se implementan los servicios de negocio como parte de los ambientes de solución; por lo tanto, a mejor juicio, mejor la calidad de servicio. Y por supuesto, los servicios que se obtienen con mayor calidad redundarán en mejor calidad del ambiente de servicio obtenido.

Nota:

Vale la pena recordar que aun cuando la encapsulación de la lógica de negocio normalmente se ilustra usando servicios, son las operaciones de servicio que representan y ejecutan la lógica en la capa de servicio. Por lo tanto, es crítico asegurar que cuando se identifica la lógica conveniente para la encapsulación del servicio, esta debe ser descompuesta en operaciones individuales. Los servicios entonces se determinan por el agrupamiento propuesto de estas operaciones.

Modelos de entidad.

Las entidades primarias representan los principales documentos de negocio y áreas de transacciones de una empresa. Por ejemplo, factura, orden de compra, clientes y reclamo son entidades hito dentro de los tipos de negocios. Adicionalmente, las organizaciones modelan las entidades de acuerdo a reglas y políticas de negocio propias. Esto resulta en entidades que tienen relaciones unas con las otras.

Los servicios centrados en entidad (explicados brevemente) reflejan el modelo de la entidad conteniendo un conjunto de operaciones genéricas que facilitan los diferentes tipos de funciones asociadas al proceso de la entidad. La comunicación entre los diferentes servicios centrados en entidad también puede ser controlada por restricciones asociadas a la relación inherente entre entidades.

Nota.

Los tipos de servicios a menudo siguen la convención de nombres de POO, donde un nombre es usado para etiquetar un objeto y los verbos son usados para llamar los métodos. Debido a que los servicios centrados en entidades representan entidades de data, los servicios pueden ser usados con nombres y las operaciones con verbos.

5.3.2. Tipos de servicios de negocio derivados.

Al derivar servicios de las dos fuentes simplemente se identifican resultados en la creación de distintos tipo de servicios de negocio.

Servicios de negocios centrados en tarea.

Estos son servicios Web que han sido modelados para adecuar un proceso de negocio específico. Las operaciones son agrupadas de acuerdo a su relevancia para la ejecución de una tarea en soporte de un proceso.

Ejemplos típicos de servicios centrados en tarea son:

- VerificarFactura
- ObtenerReporteHistorico

Cada uno de estos servicios contiene operaciones que relacionan una tarea particular con el contexto del proceso. Los servicios centrados en proceso normalmente resultan de ejercicios de modelado, que están enfocados en reunir requerimientos de negocio inmediatos. Las fuentes típicas incluyen modelos de caso de uso y definiciones de proceso BMP.

Mientras que estos requieren menos esfuerzo de análisis para producir, estos tipos de servicios de negocio han limitado el potencial de reusabilidad. Modelar servicios de negocio centrados en tarea y reusables, a menudo requiere que múltiples casos de uso y modelos de procesos de negocio sean analizados primero para identificar puntos comunes, previo al modelado actual de los servicios.

Servicios de negocio centrados en entidad.

Los servicios centrados en entidad generalmente se producen como parte de un esfuerzo de análisis a largo plazo o un análisis en proceso para alinear los servicios de negocio, con los modelos corporativos de negocios que existen. Su

naturaleza genérica inherente hace que estos sean altamente reusables por numerosos procesos de negocio. Aun cuando los servicios de negocio centrados en entidad, a menudo se construyen como parte del proyecto de desarrollo de aplicación centrado en torno a un proceso de negocio particular, ellos difieren de los servicios centrados en tarea, en que estos no proveen una interfaz específica de ese proceso. En lugar de eso, la fuente de inspiración para este tipo de servicios son los modelos de entidad.

Cuando se comparan los servicios centrados en tarea, los servicios centrados en entidad, incrementan significativamente la agilidad con la cual los procesos orientados a servicio pueden ser remodelados. Esto es porque los servicios centrados en tarea por lo general son construidos para ayudar a automatizar un proceso de negocio y entonces puede estar atado a este proceso. Cuando cambia la lógica de proceso, el contexto bajo el cual se usan y se componen los servicios, también pueden cambiar. Esto puede invalidar el agrupamiento original de las operaciones de servicio y puede resultar en el requerimiento de esfuerzo para un rediseño y un redesarrollo.

Los servicios centrados en entidad requieren más análisis por adelantado, incrementando el costo de cada servicio y el tiempo requerido para producirlos. Adicionalmente, estos pueden ser tan genéricos en su naturaleza que pueden ser entregados sin el concepto de la lógica de proceso de negocio. Por lo tanto su uso puede llegar a ser dependiente de los principales controladores del negocio, así como los servicios de proceso o los servicios controladores centrados en tarea. Como se sabe, construir una capa de servicio de negocio compuesta por una serie servicios centrados en entidad controlados en su composición, por una capa mayor de orquestación establece una SOA deseada, promoviendo un alto grado de agilidad y una representación acertada del modelo de negocios.

5.3.3. Servicios de negocio y orquestación.

El servicio de proceso, una implementación de la orquestación, también puede ser clasificado como una forma de servicio de negocio. Este es mucho más

“centrado en negocio”, porque este reside en el tope de la jerarquía en la capa de servicios y es responsable de componer servicios de negocio de acuerdo a las reglas especificadas en la lógica de workflow de orquestación.

Una orquestación puede organizar una combinación de servicios de negocio centrados en tarea y centrados en entidad. El modelo de negocio central es representado por los servicios centrados en entidad, mientras que las tareas de negocio relacionadas con la lógica pueden ser implementadas en servicios centrados en tarea, que son diseñados específicamente para complementar el servicio de proceso.

Esencialmente, el uso de la orquestación establece la siguiente estructura en la capa de servicio:

- La lógica de workflow y las reglas de negocio específicas del proceso, son incrustados en una definición de proceso. La orquestación compone servicios de negocio (y posiblemente servicios de aplicación) de acuerdo a esta definición.
- Los servicios de negocio usan en composición servicios de aplicación para ejecutar la lógica del negocio.
- Los servicios de aplicación interactúan con sistemas subyacentes para procesar las funciones requeridas.

La orquestación abstrae la lógica de workflow, colocando esta fuera de las fronteras del servicio. Esto incrementa la agilidad para permitir cambios a las reglas del negocio sin afectar los servicios de negocio o aplicación. Esto es un aspecto crítico de orquestación, ya que la lógica de proceso del negocio está sujeto a muchos factores que pueden resultar en cambios. Esto incluye intervención humana, cambios a políticas corporativas y reglas de negocio y condiciones de excepción imprevisibles.

6. Análisis orientado a servicio (modelado de servicio).

Esta sección continúa, una vez estudiada la sección anterior de introducción al análisis orientado a servicio, proveyendo un proceso de modelado de servicio que nos lleva a través de pasos individuales requeridos para producir servicios y operaciones candidatos.

6.1. Modelado de servicio (un proceso paso a paso).

Un proceso de modelado de servicio es esencialmente un ejercicio en organizar la información que se ha obtenido en el paso 1 y 2 del proceso principal de análisis orientado a servicio. Las fuentes de información requeridas pueden ser diversas, que van desde varios documentos de modelo de negocio existente hasta entrevistas verbales con personal clave que puede tener el conocimiento clave un área clave de negocio. De esta manera, este proceso puede ser estructurado de diferentes formas. El proceso descrito en esta sección se considera mejor un punto de partida desde el cual se pueden diseñar a la medida dentro de las plataforma y procedimientos de análisis de negocio existentes en al organización.

6.1.1- “Servicios” versus “Servicios Candidatos”

Antes de comenzar, se va a introducir un término importante de modelado: candidato. La meta principal de la etapa de análisis orientado a servicio es entender que es lo que se necesita diseñar y construir más adelante, en las fases subsiguientes del proyecto. Por consiguiente esto es útil para recordar continuamente que por los momentos no se está implementando un diseño en esta etapa. Solo se esta realizando un análisis que resulta en un separación de lógica propuesta usada como entrada para la consideración durante la fase de diseño orientado a servicio. En otras palabras, se esta produciendo candidatos abstractos que pueden o no, ser realizados como parte del eventual diseño concreto.

La razón de esta distinción es tan relevante porque una vez que los candidatos son confirmados para el proceso de diseño, estos están sujetos a las realidades de la arquitectura técnica en la cual estos esperan residir. Una vez que

las restricciones, los requerimientos y las limitaciones específicas del ambiente de implementación se descomponen en factores, el diseño final de un servicio, puede ser una partida significativa desde el correspondiente candidato original.

Así, en esta etapa, no se producen servicios; se crean servicios candidatos. Similarmente, no se definen operaciones de servicio; se proponen operaciones de servicio candidatas. Finalmente, los servicios candidatos y las operaciones de servicio candidatas son el fin último del proceso denominado modelado de servicio.

6.1.2. Descripción del proceso.

Lo siguiente es una serie de 12 pasos que componen un proceso de modelado de servicio propuesto (Figura 6.1). Específicamente, este proceso particular provee pasos para el modelado de una SOA consistente de capas de servicio de aplicación, de negocio y orquestación.

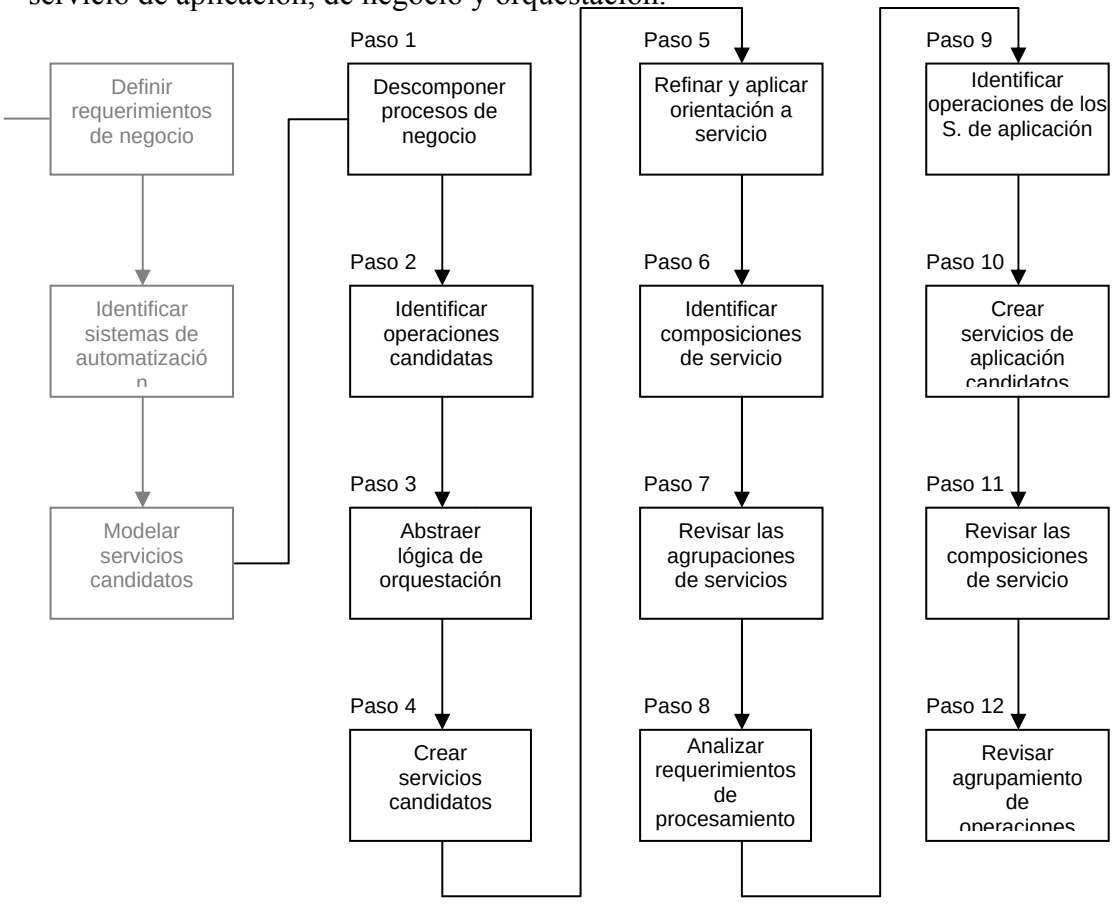


Figura 6.1.Un ejemplo de proceso de modelado de servicio.
Fuente: Earl (2005).

Paso 1: Descomponer el proceso de negocio.

Tomar el proceso de negocio documentado y descomponer este en una serie de pasos granulares de proceso. Es importante que una lógica de workflow de proceso sea descompuesta en la representación más granular de pasos de proceso, lo cual puede diferir del nivel de granularidad al cual fueron originalmente documentados los pasos del proceso. (La sección de clasificar lógica del modelo de servicio introduce nuevos términos que ayudan a distinguir el alcance de los pasos individuales del proceso.)

Paso 2. Identificar operaciones de servicio de negocio candidatas.

Algunos pasos dentro de un proceso de negocios pueden ser fácilmente identificados y no pertenecer a la lógica potencial que será encapsulada por un servicio candidato.

Los ejemplos incluyen:

- Pasos de proceso manual que no pueden ni deben ser automatizados.
- Pasos de proceso realizados por la lógica existente legada, para la cual la encapsulación de un servicio candidato no es una opción.

Al filtrar estas partes estamos dejando los pasos del proceso más relevantes a nuestro proceso de modelado de servicio.

Paso 3. Abstraer la lógica de orquestación.

Si se ha decidido construir una capa de orquestación como parte de SOA, entonces se debe identificar las partes de la lógica de proceso a la que abstraerá esta capa de manera potencial. (Si no se está incorporando una capa de servicio de orquestación, entonces se salta este paso.)

Los tipos potencial de lógica adecuada para esta capa son:

- Reglas de negocio
- Lógica condicional

- Lógica de excepción
- Lógica de secuencia.

Nótese que estas formas de lógica de orquestación pueden o no ser representada exactamente por una descripción de paso. Por ejemplo, algunas descripciones de paso de proceso que consiste de una condición y una acción (si ocurre la condición x, entonces realizar la acción y). En este caso, solo se remueve la condición y se mantiene la acción.

Nótese también, que alguna de la lógica de workflow identificada probablemente se borre de forma eventual. Esto es porque no todos los pasos del proceso necesariamente vayan a ser operaciones de servicio. Recuerde que en este punto, solo se están creando candidatos. Cuando se entre a la fase de diseño de servicio, las consideraciones prácticas entran en juego. Esto puede resultar en la necesidad de remover alguna de las operaciones candidatas, lo cual también puede requerir que la lógica de workflow correspondiente también sea removida de la capa de orquestación

Paso 4. Crear servicios de negocio candidatos.

Revisar los pasos del procesamiento que se mantienen y determinar uno o más contextos lógicos con los cuales estos pasos pueden ser agrupados. Cada contexto representa un servicio candidato. Los contextos con los que se termina, dependerán de los tipos servicios de negocio que se han seleccionado para construir.

Por ejemplo, los servicios de negocio centrados en tarea requerirán un contexto específico para el proceso, mientras que los servicios de negocio centrados en entidad introducirán la necesidad de agrupar los pasos del proceso de acuerdo a su relación con las entidades previamente definidas. (Cómo se estableció en la sección de derivar servicios de negocios, una SOA también puede consistir de una combinación de estos tipos de servicios de negocio).

Es importante no preocuparse de cuantos pasos pertenecen a cada grupo. El propósito primario de este ejercicio es establecer el conjunto de contextos requerido.

También es alentador que los servicios de negocio candidatos centrados en entidad estén equipados con operaciones candidatas que faciliten la reutilización futura. Por lo tanto, el alcance de este paso puede ser expandido para incluir un análisis de operaciones de servicio candidatas adicionales no requeridas por el proceso de negocio actual, pero agregadas para dotar los servicios de entidad con un conjunto completo de operaciones reusables.

Paso 5. Refinar y aplicar principios de orientación a servicio.

Hasta ahora simplemente se han agrupado pasos de proceso derivados de un proceso de negocio existente. Para hacer nuestros servicios candidatos verdaderamente dignos de una SOA, se debe hacer una revisión a fondo a la lógica subyacente de cada operación de servicio candidata propuesta.

Este paso da una oportunidad de hacer ajustes y aplicar principios de orientación a servicio. Esto es donde un estudio de soporte de servicio Web nativo para principios de orientación a servicio" viene a ser útil. Cómo se puede recordar, se han identificado cuatro principios claves como que no se proveen intrínsecamente a través del uso de servicios Web:

- Reusabilidad.
- Autonomía.
- Sin estado.
- Con capacidad de ser descubiertos (discoverability).

De estos cuatro, solo los dos primeros son importantes en la etapa de modelado de servicio. Los dos siguientes de la lista son tratados en el proceso de diseño orientado a servicio. Por lo tanto, el mayor interés en este paso es también asegurar que cada operación de servicio candidata sea potencialmente reusable y tan autónoma como sea posible.

Paso 6: Identificar las composiciones de servicios candidatos.

Identificar un conjunto de los escenarios más comunes que pueden tener lugar dentro de las fronteras del proceso de negocios. Para cada escenario, seguir los pasos del proceso requeridos, como existen hasta ahora.

Este ejercicio lleva a cabo lo siguiente:

- Da una buena idea en cuanto a cuan apropiado es agrupar los pasos del proceso.
- Este demuestra la relación potencial entre las capas de orquestación y de servicio de negocio.
- Este identifica composiciones de servicios potenciales.
- Este destaca cualquier lógica de workflow o pasos del proceso perdidos.

Hay que asegurarse que dentro de los escenarios seleccionados se incluyan condiciones de falla que envuelvan lógica de manejo de excepciones. Nótese también que cualquier capa de servicio que se establece en este punto, aun esta en fase preliminar y sujeta a revisiones durante el proceso de diseño.

Paso 7: Revisar agrupamiento de operaciones de servicios de negocios.

Basándose en los resultados del ejercicio de composición en el paso 6, revisitar el agrupamiento de los pasos del proceso de negocio y revisar la organización de operaciones de servicio candidatas cuanto sea necesario. No es inusual consolidar o crear nuevos grupos (servicios candidatos) en este punto.

Paso 8. Analizar los requerimientos de procesamiento de la aplicación.

Al final del paso 6, se debe haber creado una vista centrada en negocios de las capas de servicios. Esta vista podría muy bien consistir de ambos, tanto de los servicios candidatos de negocio y de aplicación, pero la atención hasta el momento ha estado en representar la lógica del proceso de negocio.

Estas próximas series de pasos son opcionales y más convenientes para los procesos de negocio complejos y grandes ambientes orientados a servicio. Esto

requiere de un estudio más a fondo de los requerimientos de proceso subyacentes de todos los servicios candidatos para abstraer cualquiera de los servicios candidatos adicionales centrados en tecnología desde esta vista, que complementará una capa preliminar de servicios de aplicación. Para lograr esto, cada paso de proceso identificado hasta ahora se requiere de un mini análisis.

Específicamente, lo que se necesita determinar es:

- Que lógica de aplicación subyacente se necesita ejecutar para procesar la acción descrita por la operación candidata.
- Si la lógica de aplicación requerida ya existe o si se necesita desarrollar una nueva.
- Si la lógica de aplicación requerida abarca las fronteras de la aplicación. En otras palabras, ¿Se requiere más de un sistema para completar esta acción?

Nótese que la información recopilada durante el paso 2 del principal proceso de análisis orientado a servicio será referenciado en este punto.

Paso 9. Identificar las operaciones candidatas de los servicios de aplicación.

Dividiendo cada requerimiento de procesamiento de la lógica de aplicación en una serie de pasos. Hay que ser explicito en cuanto a como se etiquetan estos pasos a fin de que estos reflejen la función que realizan.

Paso 10. Crear servicios de aplicación candidatos.

Agrupar estos pasos del proceso de acuerdo al contexto predefinido. Con los servicios de aplicación candidatos, el contexto primario es una relación lógica entre las operaciones candidatas. Esta relación se puede basar en cualquier número de factores, incluyendo:

- Asociación con un sistema legado específico.
- Asociación con uno o más componentes de solución.
- Agrupación lógica de acuerdo al tipo de función

Algunos otros aspectos una vez divididos en servicios candidatos, están sujetos al proceso de diseño orientado a servicio. Por ahora, este agrupamiento establece una capa de servicio de aplicación preliminar.

Paso 11: Revisar las composiciones de servicios candidatos.

Revisitar los escenarios originales que se identificaron en el paso 5 y revisarlos de nuevo. Solamente, esta vez también se incorporan los nuevos servicios de aplicación candidatos. Esto resultará en el mapeo de actividades elaboradas que le dan vida a composiciones de servicio expandidas. Hay que asegurarse de mantener el seguimiento de cómo los servicios de negocio candidatos guían a los servicios de aplicación candidatos durante este ejercicio.

Paso 12: Revisar el agrupamiento de las operaciones de los servicios de aplicación.

Al pasar a través de los movimientos de mapear los escenarios de actividad del paso 11, usualmente resultará en cambios para el agrupamiento y definición de las operaciones de servicio de aplicación candidatas. Esto probablemente también apunte a cualquier omisión los pasos de procesamiento a nivel de aplicación, resultando en la adición de nuevas operaciones de servicio candidatas y a lo mejor inclusive nuevos servicios candidatos.

Paso opcional: Mantener un inventario de servicios candidatos.

Has ahora, este proceso ha asumido que es la primera vez que usted modela servicio candidatos. Idealmente, cuando se practicaron iteraciones subsecuentes de este proceso, se deben tomar en cuenta los servicios candidatos existentes antes de crear nuevos.

Sin embargo, puede ser difícil conseguir oportunidades de reutilización cuando se modelan documentos. Esto es porque un tanto de la verborrea usada para describir las operaciones y los servicios candidatos se pierde cuando esta información se trasladada posteriormente en diseños de servicio concretos. Como resultado, este paso es considera opcional.

7. ADL.

7.1. Criterios de definición de un ADL

Los ADLs se remontan a los lenguajes de interconexión de módulos (MIL) de la década de 1970, pero se han comenzado a desarrollar con su denominación actual a partir de 1992 o 1993, poco después de fundada la propia arquitectura de software como especialidad profesional. La definición más simple es la de Tracz (1997) que define un ADL como una entidad consistente en cuatro "Cs": componentes, conectores, configuraciones y restricciones (constraints). Una de las definiciones más tempranas es la de Vestal (1999) quien sostiene que un ADL debe modelar o soportar los siguientes conceptos:

- Componentes
- Conexiones
- Composición jerárquica, en la que un componente puede contener una sub-arquitectura completa
- Paradigmas de computación, es decir, semánticas, restricciones y propiedades no funcionales
- Paradigmas de comunicación
- Modelos formales subyacentes
- Soporte de herramientas para modelado, análisis, evaluación y verificación
- Composición automática de código aplicativo

Basándose en su experiencia sobre Rapide, Luckham y Vera (1995) establecen como requerimientos:

- Abstracción de componentes
- Abstracción de comunicación
- Integridad de comunicación (sólo los componentes que están conectados pueden comunicarse)

- Capacidad de modelar arquitecturas dinámicas
- Composición jerárquica
- Relatividad (o sea, la capacidad de mapear o relacionar conductas entre arquitecturas)

Tomando como parámetro de referencia a UniCon, Shaw y otros (1995) alegan que un ADL debe exhibir:

- Capacidad para modelar componentes con aserciones de propiedades, interfaces e implementaciones
- Capacidad de modelar conectores con protocolos, aserción de propiedades e implementaciones
- Abstracción y encapsulamiento
- Tipos y verificación de tipos
- Capacidad para integrar herramientas de análisis

Otros autores, como Shaw y Garlan (1994) estipulan que en los ADLs los conectores sean tratados explícitamente como entidades de primera clase (lo que dejaría al margen de la lista a dos de ellos al menos) y han afirmado que un ADL genuino tiene que proporcionar propiedades de composición, abstracción, reusabilidad, configuración, heterogeneidad y análisis, lo que excluiría a todos los lenguajes convencionales de programación y a los MIL.

La especificación más completa y sutil (en tanto que se presenta como provisional, lo que es propio de un campo que recién se está definiendo) es la de Medvidovic (1996):

Componentes

- Interfaz
- Tipos
- Semántica
- Restricciones (constraints)

- Evolución
- Propiedades no funcionales

Conectores

- Interfaz
- Tipos
- Semántica
- Restricciones
- Evolución
- Propiedades no funcionales

Configuraciones arquitectónicas

- Comprensibilidad
- Composicionalidad
- Heterogeneidad
- Restricciones
- Refinamiento y trazabilidad
- Escalabilidad
- Evolución
- Dinamismo
- Propiedades no funcionales

Soporte de herramientas

- Especificación activa
- Múltiples vistas
- Análisis
- Refinamiento
- Generación de código

- Dinamismo

Nótese, en todo caso, que no se trata tanto de aspectos definitorios del concepto de ADL, sino de criterios para la evaluación de los ADLs existentes, o sea de un marco de referencia para la clasificación y comparación de los ADLs.

En base a las propuestas señaladas, definiremos a continuación los elementos constitutivos primarios que, más allá de la diversidad existente, son comunes a la ontología de todos los ADLs y habrán de ser orientadores de su tratamiento en este estudio.

Componentes. Representan los elementos computacionales primarios de un sistema. Intuitivamente, corresponden a las cajas de las descripciones de caja-y-línea de las arquitecturas de software. Ejemplos típicos serían clientes, servidores, filtros, objetos, pizarras y bases de datos. En la mayoría de los ADLs los componentes pueden exponer varias interfaces, las cuales definen puntos de interacción entre un componente y su entorno.

Conectores. Representan interacciones entre componentes. Corresponden a las líneas de las descripciones de caja-y-línea. Ejemplos típicos podrían ser tuberías (pipes), llamadas a procedimientos, broadcast de eventos, protocolos cliente-servidor, o conexiones entre una aplicación y un servidor de base de datos. Los conectores también tienen una especie de interfaz que define los roles entre los componentes participantes en la interacción.

Configuraciones o sistemas. Se constituyen como grafos de componentes y conectores. En los ADLs más avanzados la topología del sistema se define independientemente de los componentes y conectores que lo conforman. Los sistemas también pueden ser jerárquicos: componentes y conectores pueden subsumir la representación de lo que en realidad son complejos subsistemas.

Propiedades. Representan información semántica sobre un sistema más allá de su estructura. Distintos ADLs ponen énfasis en diferentes clases de

propiedades, pero todos tienen alguna forma de definir propiedades no funcionales, o pueden admitir herramientas complementarias para analizarlas y determinar, por ejemplo, el throughput y la latencia probables, o 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.

Estilos. Representan familias de sistemas, un vocabulario de tipos de elementos de diseño y de reglas para componerlos. Ejemplos clásicos serían las arquitecturas de flujo de datos basados en grafos de tuberías (pipes) y filtros, las arquitecturas de pizarras basadas en un espacio de datos compartido, o los sistemas en capas. Algunos estilos prescriben un framework, un estándar de integración de componentes, patrones arquitectónicos o como se lo quiera llamar.

Evolución. Los ADLs deberían soportar procesos de evolución permitiendo derivar subtipos a partir de los componentes e implementando refinamiento de sus rasgos. Sólo unos pocos lo hacen efectivamente, dependiendo para ello de lenguajes que ya no son los de diseño arquitectónico sino los de programación. Propiedades no funcionales. La especificación de estas propiedades es necesaria para simular la conducta de runtime, analizar la conducta de los componentes, imponer restricciones, mapear implementaciones sobre procesadores determinados, etcétera.

7.2. Lista de ADL

A continuación se presenta una lista de los ADL más representativos hasta la fecha:

Tabla 1. Cronología de ADL.

ADL	Fecha	Investigador - Organismo
Acme	1995	Monroe & Garlan (CMU), Wile (USC)
Aesop	1994	Garlan (CMU)
ArTek	1994	Terry, Hayes-Roth, Erman (Teknowledge, DSSA)
Armani	1998	Monroe (CMU)
C2 SADL	1996	Taylor/Medvidovic (UCI)
CHAM	1990	Berry / Boudol
Darwin	1991	Magee, Dulay, Eisenbach, Kramer
Jacal	1997	Kicillof , Yankelevich (Universidad de Buenos Aires)
LILEANNA	1993	Tracz (Loral Federal)
MetaH	1993	Binns, Englehart (Honeywell)
Rapide	1990	Luckham (Stanford)
SADL	1995	Moriconi, Riemenschneider (SRI)
UML	1995	Rumbaugh, Jacobson, Booch (Rational)
UniCon	1995	Shaw (CMU)
Wright	1994	Garlan (CMU)
xADL	2000	Medvidovic, Taylor (UCI, UCLA)

Fuente: Regional Architect Forum 2006.

7.4. UML como ADL

Según Blanco S. (2006) a continuación se describen a continuación los diagramas de UML 2.0 que se utilizarían para documentar las arquitecturas, y de que manera se utilizarían.

Diagrama de estructura compuesta y componentes

Es un diagrama que muestra la estructura interna de un clasificador, incluyendo sus puntos de interacción con otras partes del sistema. Muestra la configuración y las relaciones entre las partes que juntan realizan las tareas del sistema contenedor.

El diagrama de estructura compuesta, esta formado por los siguientes elementos:

- Clase: Es la representación de un objeto (componente), que refleja su estructura y comportamiento dentro del sistema.
- Interfaz: Es una especificación de comportamiento que implementa un contrato a seguir.
- Parte: Son instancias en tiempo de ejecución de una clase o interfaz.
- Puerto: Define la interacción de un componente con su ambiente. Las interfaces expuestas controlan esta interacción.
- Colaboración: Define un conjunto de roles cooperantes y sus conexiones. Se utiliza para mostrar en conjunto una funcionalidad.
- Interfaz expuesta: Es la manera gráfica de demostrar la forma de comunicación del componente (clase, interfaz o parte)
- Conector: Ilustra un enlace de comunicación entre partes para llegar al propósito de la estructura
- Ensamble: oficia de puente entre una interfaz expuesta “requerida” de un componente, con una interfaz expuesta “provista” por otro componente.

- Enlace de rol: Es el mapeo entre los roles internos de una colaboración y las respectivas partes necesarias para implementar una situación específica.
- Representación: indica que una colaboración se usa en un componente
- Ocurrencia: Indica que una colaboración representa a un componente.

El diagrama de componente, esta formado por los siguientes elementos:

- Paquete: Es un contenedor de otros paquetes, componentes, etc. Muy usado para mostrar espacios de nombres o ensamblados.
- Componente: Es una parte modular de un sistema, cuyo comportamiento se define por sus interfaces requeridas y provistas.
- Clase: Es la representación de un objeto (componente), que refleja su estructura y comportamiento dentro del sistema.
- Interfaz: Es una especificación de comportamiento que implementa un contrato a seguir.
- Objeto: Es una instancia de una clase en tiempo de ejecución
- Puerto: Define la interacción de un componente con su ambiente. Las interfaces expuestas controlan esta interacción.
- Interfaz expuesta: Es la manera gráfica de demostrar la forma de comunicación del componente (clase, interfaz o parte)
- Artefacto: es cualquier pieza física de información usada o producida por un sistema
- Ensamble: oficia de puente entre una interfaz expuesta “requerida” de un componente, con una interfaz expuesta “provista” por otro componente.
- Delegado: Define que componente interno de un componente atiende el pedido a un puerto determinado.

Se usarían para modelar el aspecto físico de un sistema. Cada componente puede ser un ejecutable, librerías, archivos de datos o cualquier otro recurso físico que es parte de un sistema.

Ejemplo:

El sistema Negocio .NET tiene 3 interfaces requeridas: TCP, http y MSMQ, con las que se comunica con 3 servicios distintos que atienden los pedidos de dichas interfaces.

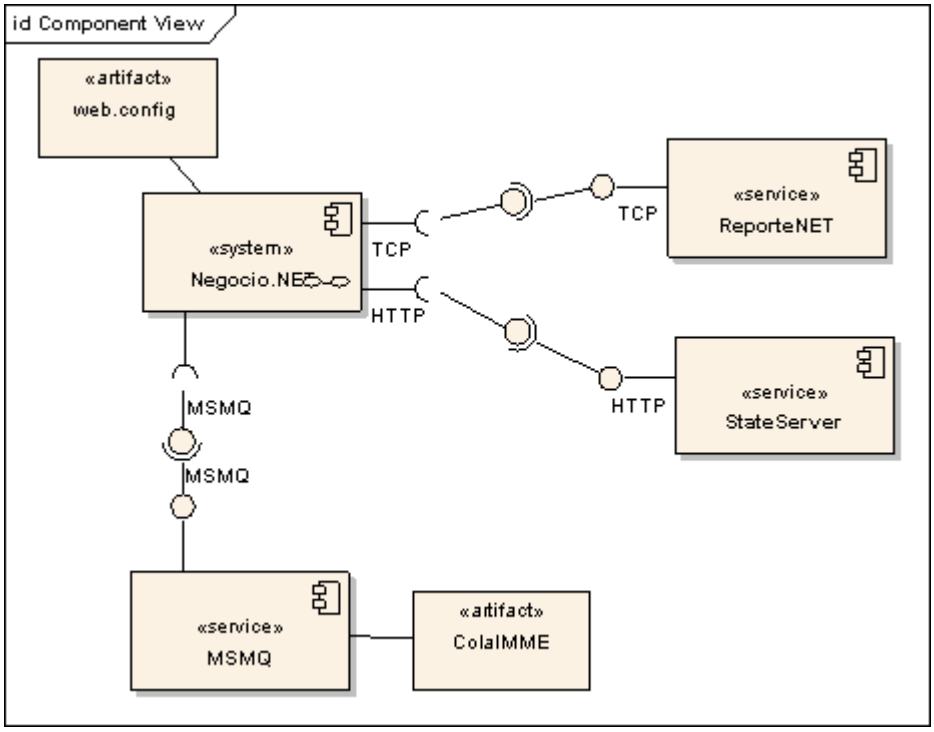


Figura 7.1.Diagrama UML de estructura compuesta y componentes.
Fuente: Regional Architect Forum 2006.

Diagrama de implementación

Modela la arquitectura en tiempo de ejecución de un sistema. Muestra la configuración de los elementos hardware (nodos), y muestra como los elementos y artefactos software están distribuidos en estos nodos.

El diagrama de implementación esta formado por los siguientes elementos:

- Nodo: Representa un equipo sobre el que se implementará el sistema.

- Componente: Es una parte modular de un sistema, cuyo comportamiento se define por sus interfaces requeridas y provistas.
- Interfaz: Es una especificación de comportamiento que implementa un contrato a seguir.
- Artefacto: es cualquier pieza física de información usada o producida por un sistema
- Especificación de implementación: Especifica una guia de parámetros de implementación de un artefacto, como ser archivos de configuración o algún componente hardware.
- Paquete: Es un contenedor de otros paquetes, componentes, etc. Muy usado para mostrar espacios de nombres o ensamblados.
- Asociación: Muestra que dos elementos están relacionados.
- Clase Asociación: Es una asociación que también tiene propiedades de clase. No solo conecta dos componentes, sino que especifica un conjunto de características que llevan a la relación.
- Generalización: Indica herencia
- Implementación: es una relación que indica en que nodo se implementa un componente SW o un artefacto
- Manifiesto: es una relación que indica que el fuente de un artefacto encierra la especificación de un componente.
- Dependencia: Indica que un componente (nodo, SW, artefacto, etc.) depende de otro componente.
- Inclusión: Indica que un componente está empaquetado dentro de otro.

Se usaría para mostrar una vista estática de la configuración en tiempo de ejecución de los nodos de procesamiento y los componentes que corren en dichos nodos. Pueden usarse, también, para mostrar las conexiones entre hardware, software y cualquier middleware que se utilice en un sistema.

Ejemplo:

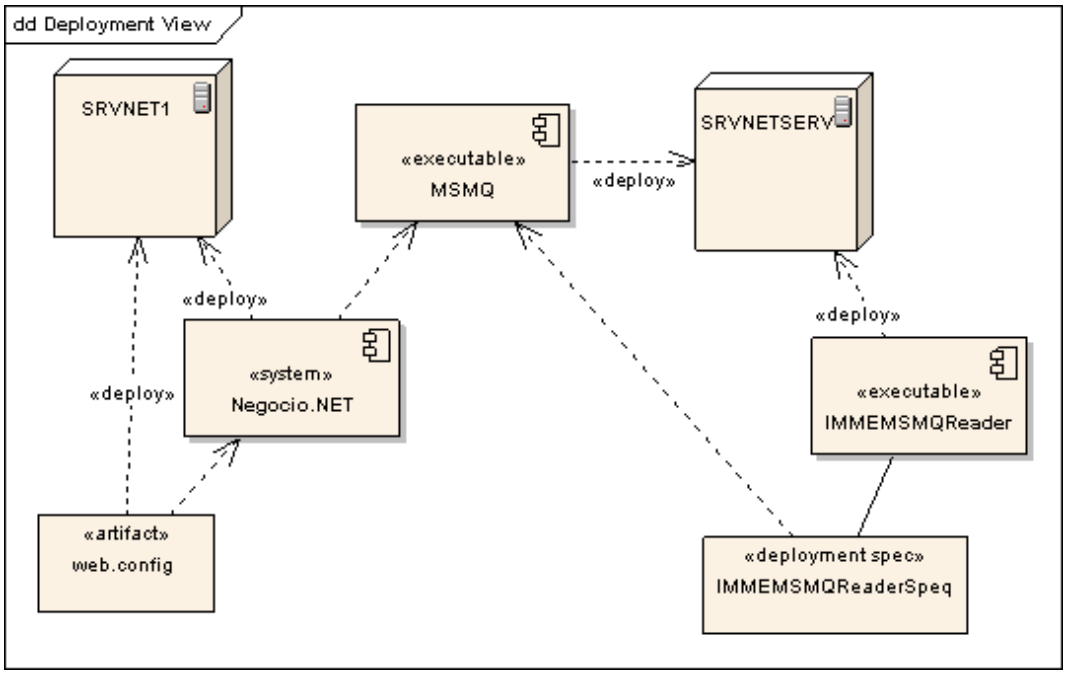


Figura 7.1.Diagrama UML de implementación.
Fuente: Regional Architect Forum 2006.

Diagrama de empaquetamiento

Se usan para reflejar la organización de paquetes y sus elementos.

Está formado por los siguientes elementos:

- Paquete: Es un contenedor de otros paquetes, componentes, etc. Muy usado para mostrar espacios de nombres o ensamblados.
- Clase: Es la representación de un objeto (componente), que refleja su estructura y comportamiento dentro del sistema.
- Interfaz: Es una especificación de comportamiento que implementa un contrato a seguir.
- Objeto: Es una instancia de una clase en tiempo de ejecución
- Tabla: Es una clase estereotipada. Se le puede asignar valores de propiedad de columnas, etc.

- Asociación: Se utiliza para realizar asociaciones entre mas de dos elementos
- Asociado: Asocia dos elementos
- Generalización: Indica herencia de comportamiento o propiedades.
- Composición: Indica que un componente esta formado por otros componentes.
- Agregación: Indica que un componente contiene a otros componentes.
- Clase Asociación: Es una asociación que también tiene propiedades de clase. No solo conecta dos componentes, sino que especifica un conjunto de características que llevan a la relación.
- Ensamble: oficia de puente entre una interfaz expuesta "requerida" de un componente, con una interfaz expuesta "provista" por otro componente.
- Dependencia: Indica que un componente (nodo, SW, artefacto, etc.) depende de otro componente.
- Inclusión: Indica que un componente esta dentro de otro componente.
- Merge: Indica que el contenido de un paquete esta incluido en el de otro paquete, en el momento de implementar
- Importación: Indica que el contenido de un paquete se importa a otro paquete.

Se usaría para mostrar como se relacionan los distintos elementos componentes de un sistema.

Ejemplo:

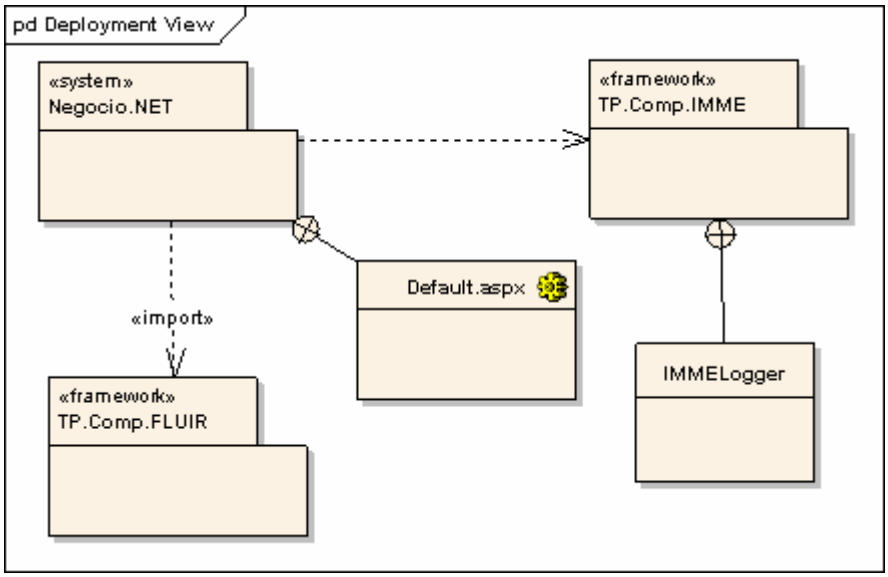


Figura 7.1.Diagrama UML de empaquetamiento.
Fuente: Regional Architect Forum 2006.

Diagrama de estados

Modela el comportamiento de un objeto, especificando la secuencia de estados por los que pasa un objeto como respuesta a eventos ocasionados sobre el mismo, o por el mismo.

El diagrama esta formado por los siguientes elementos:

- Estado: Indica una situación que se mantiene por una condición invariante, que puede ser estática, o dinámica.
- Sub máquina: Es un puntero a un diagrama de estados hijo.
- Inicial: Indica donde comienza el flujo de estados.
- Final: Indica donde termina el flujo de estados
- Historial: Se utiliza para volver a un estado anterior. Hay dos tipos:

Bajo: representa el estado más reciente.

Profundo: Representa la configuración activa mas reciente

- Sincronización: Indica que caminos concurrentes del diagrama corren sincrónicamente.
- Objeto. Es una instancia de una clase en tiempo de ejecución.
- Decisión: Indica un punto de progresión condicionada. Dada una condición, según el valor retornado, se tomará un camino u otro.
- Unión: Se usa para unir en un mismo punto a varios caminos
- Fork / Join: Se usa para separar el flujo en distintos caminos, y / o unir el flujo de varios caminos en uno.
- Transición: Es el movimiento lógico de un estado a otro.
- Flujo de Objeto: Indica un flujo o transición de estado.

Se usaría para mostrar como un componente va cambiando de estados, cuales son los estados esperados que tenga, y mediante que evento (interno o externo), se modifica su estado.

Ejemplo:

El siguiente diagrama muestra el diagrama de estados de un puerto de comunicaciones

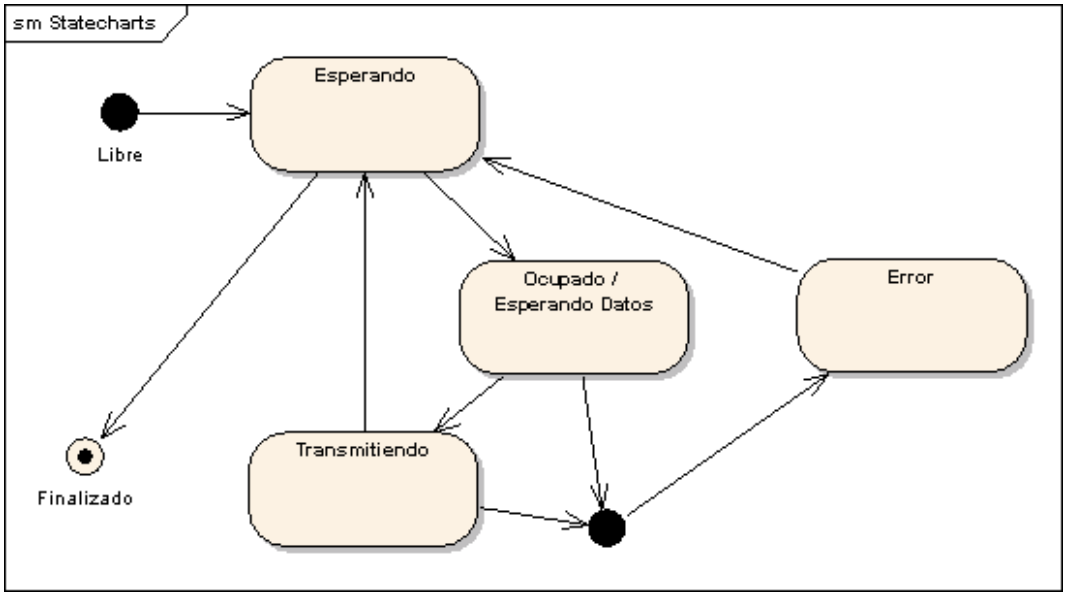


Figura 7.1.Diagrama UML de estados.
Fuente: Regional Architect Forum 2006.

Diagrama Temporal

Se utilizan para mostrar el cambio de estado o valor de uno o más elementos, a través del tiempo. También puede mostrar la interacción entre eventos y el tiempo y duración constantes que los dominan.

El diagrama esta compuesto por los siguientes elementos:

- Línea de estado: Es el camino que un objeto toma a través de una medida de tiempo. Sigue transiciones discretas entre los distintos estados.
- Línea de valor: Es el camino que un objeto toma a través de una medida de tiempo. Sigue transiciones continuas entre los distintos estados.
- Etiqueta de mensaje: Es una forma de denotar mensajes entre líneas de vida.
- Fin de mensaje: Fin de un estado o valor.
- Puerta del diagrama: Punto en el que se transmiten los mensajes
- Mensaje: Links de comunicación entre líneas de vida

Lo usaríamos en el caso de que algún componente de un sistema cambie su estado o valores mediante el factor tiempo (una tarea que tenga que levantarse cada x tiempo).

Ejemplo:

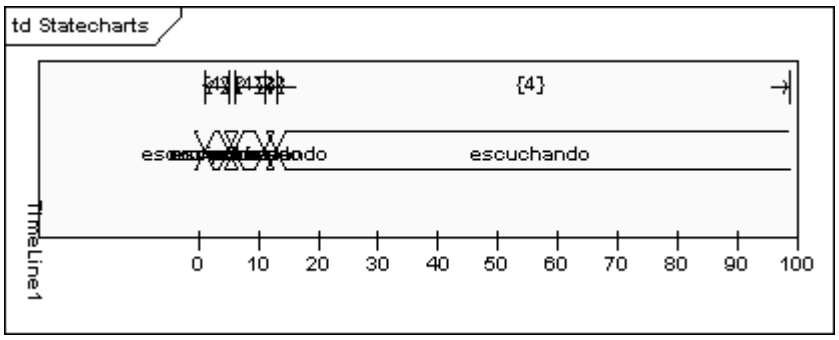


Figura 7.1.Diagrama temporal.
Fuente: Regional Architect Forum 2006.

8. Servicios Web.

La evolución vertiginosa de Internet y el negocio electrónico necesita de tecnologías software que faciliten la "conexión de información, sistemas, dispositivos y personas". Desde la aparición de la Web los procesos de negocio ofrecidos a través de la Web han ido mejorando, en una primera fase (1996 aproximadamente) se inicia el interés por el desarrollo de portales fomentando la presencia de las empresas, en una segunda fase (1997 aproximadamente) se implanta el comercio electrónico y las transacciones a través de Internet (*eCommerce*), en una tercera fase (desde 1999 hasta la actualidad) se inicia lo que se llama la Economía Digital con el desarrollo e implantación de los negocios (*eBusiness*), las empresas (*eEnterprises*) y los mercados (*eMarkets*) electrónicos. Los mercados electrónicos constituyen la nueva generación de negocios digitales proporcionando bienes y servicios de carácter electrónico que sustituyen a los tradicionales. Dichos servicios pueden integrarlos otras empresas en sus modelos de negocio: mensajería, reserva de vuelo, etc. En este contexto se está evolucionando hacia empresas y relaciones virtuales a través de Internet con clientes cada vez más exigentes.

Los "Servicios Web" constituyen un esfuerzo para construir plataformas distribuidas para la Web que den soporte adecuado, como mecanismo de implementación, a la Economía Digital. Los Servicios Web son la última evolución tecnológica desde el clásico modelo cliente/servidor, el middleware distribuido mediante sistemas basados en RPC, monitores TP, brokers de objetos, monitores de objetos a la orientación a mensajes (MOM), pasando por el middleware de integración de aplicaciones (EAI), los sistemas de Workflow y por último la tecnología Web clásica y los sitios Web (o portales). La nueva evolución tecnológica debe: permitir la interoperabilidad universal, la amplia y rápida adopción y un soporte eficiente de entornos abiertos. Los servicios Web para tener éxito y promover su rápida implantación a nivel mundial deben cumplir con los siguientes requisitos:

- Basado en estándares.

- Soporte a la Pervasividad (*Everywhere, Everytime, Everyplace*).
- Permitir la integración de aplicaciones de manera flexible.
- Estar centrado en mensajes y documentos, no en APIs.

8.1. Definición de Servicio Web.

Una definición genérica de Servicio Web es: "Una aplicación accesible a otras aplicaciones a través del Web". El UDDI consortium (www.uddi.org) introduce la siguiente definición "*Self.contained, modular business applicationes that have open, Internet-oriented, Standard-bases interfaces*", la referencia munidal en el mundo Web W3C lo define como "*es un sistema software identificado por una URI, cuyos interfaces públicas y enlaces se definen y describen usando XML. Su definición puede ser descubierta por otros sistemas software. Estos sistemas pueden interactuar con el servicio Web de la forma prescrita por su definición, usando mensajes basados en XML a través de protocolos estándares de Internet*", una definición precisa disponible en Webopedia dice "*a standardized way of integrating web based applications using XML, SOAP, WSDL and UDDI open Standard over an Internet protocol backbone, XML is used to tag the data, SOAP us used to transfer data, WSDL is used for describing the services available, and UDDI is used for listing what services are available*". En definitiva, un Servicio Web expone funcionalidad a un consumidor, es una URL programable y proporciona mecanismos para invocar opraciones de forma remota (a través de Internet), está basado en estándares Web(http,XML,SOAP,WSDL,UDDI, y más que vendrán…), puede implementarse en cualquier lenguaje y en cualquier plataforma, actuando como caja negra (componentes reutilizables y alquilables).

8.2. Estándares y Tecnologías Subyacentes. Infraestructura Básica

La infraestructura mínima que requieren los Servicios Web se puede definir en términos de:

- Lo que va en "la red": Formatos y protocolos de comunicación

- Lo que describe lo que va en la red: Lenguajes de Descripción de Servicios
- Lo que nos permite encontrar y almacenar dichas descripciones: Descubrimiento de Servicios.

8.2.1. La pila de las Tecnologías.

Las especificaciones que se han desarrollado para implementar estos mecanismos (en muchas propuestas, se presentan como una pila de tecnologías donde las especificaciones superiores hacen uso de las inferiores, (ver figura 8.1)) son:

1. HTTP: (*Hypertext Transfer Protocol*) HTTP es un protocolo estándar de W3C para la transferencia de documentos en Internet. Los Servicios Web lo utilizan como mecanismo de comunicación. Es un protocolo genérico y sin estado.
2. XML: (*eXtensible Markup Language*) lenguaje que deriva de SGML, diseñado para representar y transferir datos estructurados, separa datos de formateo y transformación, se utiliza como base para definir los formatos de….
3. SOAP: (Simple *Object Access Protocol*, www.w3.org/2002/ws/), ofrece los mecanismos de comunicación básicos para el envío de mensajes en formato XML, permitiendo la invocación remota de servicios. Normalmente funciona sobre HTTP, pero no siempre. SOAP es el "*sine qua non*" de los servicios web.
4. WSDL: (Web *Services Description Language*, www.w3.org/TR/wsdl), es un formato basado en XML (desarrollado por IBM y Microsoft) para describir de manera formal servicios WEB.
5. UDDI: (*Universal Description, Discovery, and Integration*, www.uddi.org), es un directorio que contiene un registro/repositorio de descripciones de servicios Web.

Figura 8.1. La pila de los Servicios Web.
Fuente: Pelechano (2005).

8.2.1.1. SOAP (Simple Object Access Protocol). El Formato de los Mensajes.

SOAP (recomendado por W3C en 2003) define un protocolo que da soporte a la interacción (datos + funcionalidad) entre aplicaciones en entornos distribuidos y heterogéneos, es interoperable (neutral a plataforma y lenguajes de programación, independiente del Hardware y protocolos. Funciona sobre la infraestructura (Estándares) existente en Internet. SOAP define cómo organizar información usando XML de forma estructurada para intercambiarla entre distintos sistemas.

¿Qué especifica SOAP?

SOAP especifica:

- Un formato de mensaje para una comunicación unidireccional, describiendo cómo se empaqueta la información en documentos XML.
- Un conjunto de convenciones para usar mensajes SOAP para implementar el patrón de interacción RPC, definiendo cómo los clientes pueden invocar un Procedimiento Remoto enviando un mensaje SOAP y cómo los servicios pueden responder enviando otro mensaje al llamador.
- Un conjunto de reglas que una entidad que procesa mensajes SOAP debe seguir, definiendo en particular los elementos XML que una entidad debe leer y entender, así como las acciones que deben toma si no entienden el contenido. Reglas de Codificación de los Datos.

- Una descripción de cómo se debe transportar un mensaje SOAP sobre HTTP y SMTP. Se definirán "*Bindings*" a otros protocolos de transporte en futuras versiones de la especificación.

El Formato del Mensaje

SOAP intercambia información mediante mensajes. Los mensajes se utilizan como envoltorios que la aplicación utiliza para guardar la información que quiere enviar. Cada envoltorio contiene dos partes: Una cabecera (opcional) y un cuerpo (obligatorio). La cabecera y el cuerpo pueden tener múltiples subpartes en forma de bloques de la cabecera y bloques del cuerpo (ver figuras 8.2 y 8.3).

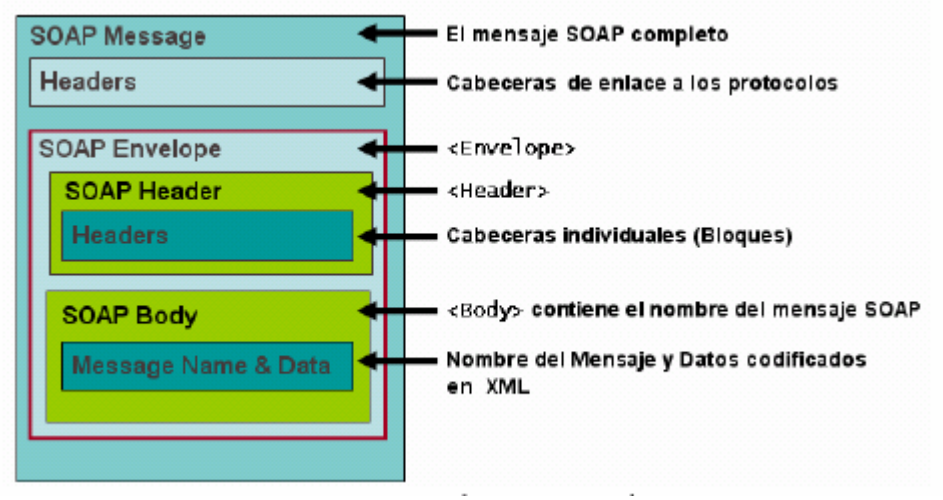


Figura 8.2. Formato de un Mensaje SOAP.
Fuente: Pelechano (2005).

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi=…>
  <soap:Header>
    <WoodgroveAuthInfo xmlns="http://tempuri.org/">
      <Username>string</Username>
      <Password>string</Password>
    </WoodgroveAuthInfo>
  </soap:Header>
  <soap:Body>
    <GetAccount xmlns="http://tempuri.org/">
      <acctID>int</acctID>
    </GetAccount>
  </soap:Body>
</soap:Envelope>
```

Figura 8.3. Mensaje SOAP.
Fuente: Pelechano (2005).

8.2.1.2. WSDL (Web Services Description Language). El Lenguaje de Descripción.

WSDL creado originalmente por IBM, Microsoft y Ariba. Tiene el rol y el propósito similar al de los IDL de las plataformas middleware. Un archivo WSDL es un documento XML que describe los Servicios Web, en particular sus interfaces. Como característica que lo diferencia de los IDL es que WSDL debe definir los mecanismos de acceso (protocolos) a los servicios Web. (IDL no los define en el middleware convencional porque se suponen por defecto). Otra característica diferenciadora es la necesidad de definir (en la especificación) la localización del servicio (puntos finales). La separación de interfaces y enlaces de protocolos, y la necesidad de incluir información de localización permite la definición de especificaciones modulares. WSDL permite definir interfaces más complejos y expresivos permitiendo definiciones de interacciones asíncronas y diferentes paradigmas de interacción, y la posibilidad de combinar o agrupar operaciones.

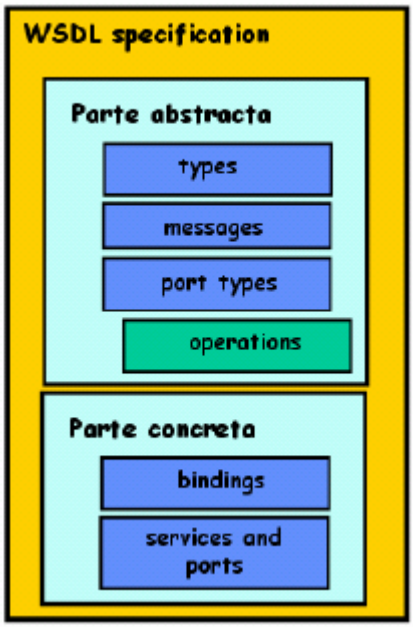


Figura 8.4. Estructura de un Documento WSDL.
Fuente: Pelechano (2005).

8.2.1.2.1. Estructura de un Documento WSDL

El documento WSDL de un servicio proporciona dos piezas de información básicas: (1) una parte o interfaz abstracta (independiente de la

aplicación) y (2) una parte concreta que define los enlaces a protocolos e información de los puntos finales de acceso al servicio (ver figura 8.4).

La parte abstracta esta compuesta por:

- Definiciones de *Port types*: análogos a los interfaces en los IDL. Cada port type es una colección lógica de operations.
- Cada *operation* define un intercambio simple de mensajes.
- Un *message* es una unidad de comunicación representando un intercambio de datos en una única transmisión lógica.
- Un sistema de tipos (*types*) usados por las *operations* (por defecto XML *schema*).

La parte concreta esta compuesta por:

- Definiciones de *Bindings*: se especifica la codificación de los mensajes, y los enlaces a protocolos de todas las operaciones y mensajes definida en un *port type*.
- Definiciones de *Ports*: se especifica en qué dirección (URI) se puede acceder la implementación del *port type*. Definen un punto final (lugar de la red) donde está el servicio.
- Definiciones de *Services*: definen una agrupación de *Ports*.

De esta forma WSDL se utiliza para describir un Servicio Web en términos de los mensajes que acepta y genera, actúa como contrato entre un consumidor (cliente) y dicho Servicio. WSDL puede describir puntos finales y sus operaciones sin especificar el formato de los mensajes o los protocolos de red (*Simple Object Access Protocol* (SOAP) 1.1, HTTP-GET/POST y MIME) a los cuales el punto final esta ligado. En el desarrollo se utiliza como entrada a los compiladores de *Stubs* y *Proxies* (ver ejemplo en figura 8.5).


```
<?xml version="1.0"?>
<definitions name="StockQuote"

    <types>
        <-- XSD for the messages -->
    </types>

    <message name="GetLastTradePriceInput">
        <-- part(s) identifying the messages within the schema -->
    </message>

    <portType name="StockQuotePortType">
        <operation name="GetLastTradePrice">
            <-- grouping messages into logical operations -->
        </operation>
    </portType>

    <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
        <-- specifying how operations map to protocols such as SOAP -->
    </binding>

    <service name="StockQuoteService">
        <port name="StockQuotePort" binding="tns:StockQuoteBinding">
            <soap:address location="http://example.com/stockquote"/>
            <-- providing specific access points for a service -->
        </port>
    </service>

</definitions>
```

Figura 8.5. Ejemplo de archivo WSDL.
Fuente: Pelechano (2005).

8.2.1.3. UDDI (Universal Description, Discovery, and Integration). El Repositorio de Servicios.

UDDI, creado originalmente por IBM, Microsoft y Ariba, desde su versión 3 (en 2002) ha pasado a manos de OASIS (www.oasis-open.org) que a partir de ahora va a determinar su futuro y extensiones.

UDDI se concibió como un Registro de Negocio = Servicio de Directorio y Nombrado sofisticado. Especifica un marco para describir y descubrir Servicios Web. UDDI define estructuras de datos y APIs para publicar descripciones de servicios en el registro y para consultar el registro para buscar descripciones publicadas. Las APIs de UDDI están especificadas con WSDL y con SOAP Binding, lo que permite acceder a ellas como Servicios Web.

La especificación UDDI tiene dos objetivos esenciales: (1) ser un soporte a los desarrolladores para encontrar información sobre Servicios Web y poder construir clientes, (2) facilitar el enlace dinámico de Servicios Web, permitiendo consultar referencias y acceder a servicios de interés.

8.2.1.3.1. Modelo de Información. Estructura de Datos de UDDI

La información en un registro UDDI se almacena en ficheros XML con una estructura jerárquica (ver figura 8.6). Los elementos de esta estructura son:

- *businessEntity*: es el elemento "top-level", describe un negocio o una entidad que ha registrado un servicio en UDDI. Ejemplos: Departamento de Contabilidad, Servidor de Aplicaciones Corporativo. Este elemento soporta información estándar tal como nombre, descripción, e información de contacto, así como información de metadatos (por ejemplo: identificadores y categorías).

- *businessService*: describe un Servicio Web que ha sido expuesto por una entidad de negocio, soporta el nombrado de un Servicio Web y lo asocia con una entidad de negocio y con la información de *binding*. Soporta la asignación de categorías al Servicio Web (industria, productos, códigos geográficos, etc.).

- *bindingTemplate*: describe la información técnica necesaria para enlazar con un Servicio Web en particular. Este elemento soporta el nombrado de un Servicio Web y su asociación con una entidad de negocio e información de *binding*. La información de *binding* se describe como un punto de acceso que posee un atributo llamado *UrlType* utilizado para especificar los siete tipos de puntos de entrada: mailto, http, Https, Ftp, Fax, Phone, Other.

- *tModel: (Technology Model*). Estructura de Metadatos Genérica para representar cualquier concepto o construcción (definiciones de protocolos, ficheros WSDL, XML *schemas*, Espacios de Nombres, esquemas de categorías, etc.).

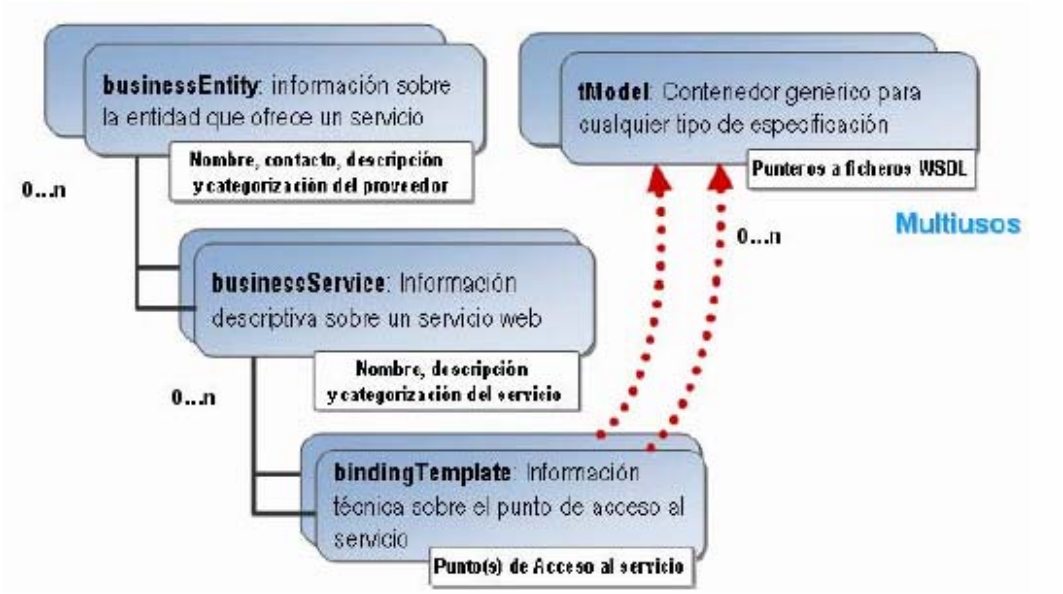


Figura 8.6. Estructura de Datos UDDI.
Fuente: Pelechano (2005).

Operacionalización de las variables

Con la finalidad de establecer los aspectos a medir en la presente investigación, se identifican las variables según el siguiente esquema:

Variables conceptuales

Información estandarizada: Se definió como la información que maneja y ofrece una organización basada en una ontología de negocios, la cual debe tener un mismo significado en todo el ambiente del negocio, estar organizada en servicios y poder ser consumida por cualquier entidad del negocio.

Servicio de información: Se definió como un componente de tecnología que conforma una arquitectura capaz de ofrecer información a través de estándares abiertos y cuyo modelado refleje unidades de lógica (conceptos) del modelo de negocios.

Variables operacionales

Para expresar las variables conceptuales en operacionales se consideraran las siguientes equivalencias:

Información estandarizada, se transformó en operable en las dimensiones organización de la información y distribución.

Servicios de información, se transformó en operable en las siguientes dimensiones, modelado y tecnología-compatibilidad.

La operacionalización de las variables de resume en la siguiente tabla:

Tabla 2. Resumen de la operacionalización de variables.

Variable	Dimensión	Indicadores	Item Cuestionario Nro. 1
Información estandarizada	Organización de la información	Vocabulario del dominio	1
		Organización en servicios	2
		Consistencia de los datos	3
	Distribución	Dificultad para ser consumida por otros entes	4
		Eficiencia en hacerla disponible	5
		Acuerdos para el intercambio	6
Servicio de información	Modelado	Basado en conceptos del dominio	7
		Cumplimiento con enunciados teóricos	8
	Tecnología - Compatibilidad	Basado en estándares abiertos	9
		Cumplimiento con exigencias de la industria	10

Fuente: El autor de la investigación.

CAPÍTULO III

MARCO METODOLÓGICO

Con la finalidad de que esta investigación se realice de una forma sistemática y reúna las características de organización y validez necesarias, se procede a enmarcarla en los protocolos establecidos para la investigación científica.

Modalidad de la investigación

Tomando como base el concepto emitido por la Universidad Pedagógica Experimental Libertador, en el Manual de Trabajos de Grado de Maestrías y Tesis Doctorales (U.P.E.L – 2003), el cual señala que:

El proyecto factible consiste en la investigación, elaboración de una propuesta de un modelo operativo viable para solucionar problemas, requerimientos o necesidades de organizaciones o grupos sociales; puede referirse a la formulación de políticas, programas, tecnologías, métodos o procesos. El proyecto debe tener apoyo en una investigación documental, de campo o un diseño de incluya ambas modalidades (Pág. 13).

Se puede afirmar que la presente investigación está enmarcada en la modalidad de proyecto factible, ya que está destinada a la solución de una necesidad presentada en una unidad organizativa de la UCLA, es este caso la Dirección de Admisión y Control de estudios.

Tipo de investigación

Se puede ubicar el presente proyecto dentro de una investigación de tipo descriptiva, ya que plantea conocer los fenómenos presentes en los proceso del negocio en la Dirección de Admisión y Control de Estudios de la UCLA, determinando sus necesidades, para describir los hechos basándose en un modelo teórico práctico previamente definido, como se sustenta en el capítulo II.

Obtención y tratamiento de los datos

En la presente investigación, es imprescindible el uso de herramientas o instrumentos que permitan una recolección de datos efectiva y eficaz, que garantice una base sólida para el análisis y el éxito del proyecto.

En el grupo de instrumentos y técnicas revisados, se ha encontrado que los más adecuados al caso investigado son la entrevista, la observación en condiciones reales de las actividades de trabajo, la revisión de documentación impresa y digital relacionada con el reglamento de funcionamiento de la UCLA. Así mismo en lo que respecta a la recolección de información referente al tema de la ingeniería de software tratado en este caso, como lo es la orientación a servicio se utiliza la revisión bibliográfica.

La implementación de la entrevista no estructurada estará dirigida a conocer el contexto donde funcionará el sistema propuesto, y las expectativas de los usuarios, o sea qué esperan del nuevo sistema propuesto.

La entrevista estructurada está orientada a conocer cómo funciona el sistema actual (el negocio), como se realizan los procesos, como se trabaja actualmente, cuáles son las fases y los procedimientos, cuáles son los cálculos que se realizan. Este instrumento también se utiliza para conocer la forma y el contenido de la información que expondrán los componentes de la solución planteada.

Los instrumentos citados serán aplicados a una muestra constituida por tres empleados, tanto del nivel operativo como del estratégico, que son los siguientes: un coordinador de admisión y control, un jefe de registro académico y un analista. Está muestra fue seleccionada de la estructura organizativa de la UCLA y es representativa de la población, que vendría a ser todo el personal que laborar en la Dirección de Admisión y Control de Estudios y en los departamentos de Registro Académico de los distintos Decanatos de la UCLA.

Fases de la Investigación

En atención a la modalidad y al tipo de investigación, las fases que se siguieron fueron tres (3): La primera de diagnostico, la cual consistió en la elaboración y aplicación de los instrumentos de recolección de datos y la presentación de los resultados. La segunda, la formulación de la propuesta de acuerdo a los resultados obtenidos en la fase anterior y tomando en cuenta las alternativas ofrecidas por la tecnología y la Ingeniería de Software. La tercera consistió en la elaboración de la propuesta de acuerdo a los resultados de la fase diagnóstica, al paradigma seleccionado para su desarrollo y a la tecnología seleccionada. Esta última fase se detalla más adelante en las etapas de desarrollo de la propuesta, luego de la presentación de los resultados de la fase diagnóstica.

Resultados de la fase de diagnóstico

En lo que respecta a la aplicación del instrumento de la entrevista no estructurada y complementado con otras técnicas como la observación directa y la revisión de documentación impresa y digital se tiene:

1. Los procesos centrales en el funcionamiento de la DACE son los de preinscripción de Bachilleres asignados, la inscripción de los Bachilleres vía Web, la apertura de expediente en la DACE por parte de los Bachilleres y la emisión de informes por cohorte de inscripción.
2. Para que un Bachiller se encuentre formalmente inscrito ante la DACE de la UCLA, debe realizar una serie de pasos posteriores a la condición de asignado, los cuales se inician con un registro de preinscripción a través de la Web para confirmar su interés en el cupo otorgado, luego según la planificación procede a realizar un segundo registro vía Web conocido como inscripción y obtener una cita para consignar los recaudos ante la DACE y realizar la apertura de expediente respectiva.
3. La información que se produce y se administra en la DACE es requerida en otras dependencias de la UCLA, principalmente en los Decanatos, los cuales la utilizan para fines diversos como por ejemplo

validar la inscripción de alumnos de nuevo ingreso. Así también otras unidades como es el caso de la Dirección de Bienestar Estudiantil hacen uso regular de la información administrada por la DACE. Es de hacer notar que unos de los actores más importantes en la universidad son los estudiantes y por condición natural los datos relativos a esta entidad son necesitados en casi todos los ámbitos de la UCLA.

4. Existe la percepción en el personal involucrado en el dominio tratado de la investigación (DACE y Decanatos) que al producirse cambios en la reglas del negocio van afectar significativamente a los sistemas actuales de TI de la organización, pero que se pueden preparar a través de nuevos diseños y tecnologías para que estos doten de mayor agilidad a la institución y que a su vez faciliten la integración con los demás sistemas que conforman la plataforma de TI de la UCLA y cualquier otro ente externo. Así también estos cambios redundarán en un aumento en el grado de desempeño de la DACE.

En relación a la entrevista estructurada se pudieron identificar situaciones específicas que reflejan deficiencias en los sistemas de información de la DACE a la hora de exponer la información y la lógica que maneja a otras aplicaciones y a otros componentes del negocio. Estas situaciones son:

1. Ausencia de un glosario de términos como parte de una ontología del negocio.
2. No existen componentes de software (servicios) que ofrezcan la información manejada por la lógica de negocio de la DACE.
3. Los mecanismos establecidos para compartir la información en la DACE no ofrecen un acceso sencillo a los datos.
4. En algunos casos los mecanismo utilizados para publicar o enviar información a otros entes, no son automatizados, se utilizan la impresión de informes, transferencia de archivos de datos en diferentes formatos vía correo electrónico y publicación de archivos de texto vía Web, que requieren la acción de una persona para la habilitación.

5. No existe un acuerdo definitivo, en cuanto a mecanismo y formato para el intercambio de datos entre la DACE y los Registro académicos.
6. Los componentes de software que existen no se encuentran diseñados de acuerdo al modelado del negocio, lo que dificulta la reacción ágil ante cualquier cambio en las reglas del negocio y la inclusión de su funcionalidad en otras aplicaciones.
7. Los componentes de software incluidos en la plataforma de TI de la DACE no ofrecen interfaces basados en estándares abiertos como por ejemplo el uso del lenguaje XML.

Los resultados obtenidos producto de la aplicación de los instrumentos citados se complementan con la revisión bibliográfica, la cual aporta elementos de decisión, para apoyar la selección de una estrategia que permita abordar las situaciones encontradas. A tal efecto luego investigar el desarrollo de nuevos paradigmas en el desarrollo de software y de tecnologías disponibles se dispuso de la aplicación del modelo de orientación a servicio en el contexto de la DACE, para de esta manera obtener su fin ultimo que es una Arquitectura de Software Orientada a Servicio. Así mismo la tecnología seleccionada para emprender diseño de los componentes es la conocida como los Servicios Web, la cual es hoy en día la que cumple con la mayor cantidad de enunciados incluidos en la teoría de SOA. A continuación se describen la etapas en el desarrollo de la aplicación de la metodología de sistemas.

Etapas en el desarrollo de propuesta

Las etapas que superaron para llevar a cabo el desarrollo de la propuesta, en cuanto a Ingeniería de Software y orientación a servicio, son un desarrollo de metodología delineada por Earl(2005), ya que la aplicación de estas fases garantiza el logro de los objetivos específicos descritos en el capítulo I.

El orden lógico de las fases de la metodología coincide con los objetivos específicos planteados, las cuales se describen a continuación un mayor nivel de especificidad:

1. Seleccionar una estrategia para realizar el proceso de desarrollo (análisis y diseño) orientado a servicio, dentro de las tres conocidas (top-down, bottom-up, y meet in the middle o ágil), propuestas por Earl (2005).
2. Realizar el análisis orientado a servicios, determinado la lógica del negocio y los procesos para identificar las operaciones y servicios candidatos. Se determina el alcance potencial de la SOA. Se modelan las capas de servicio delineadas y los servicios individuales como servicios candidatos para componer una SOA prelimitar.
3. Realizar el diseño de los servicios seleccionados. Determinar como se construirán los servicios basándose en estándares, para incorporar convenciones industriales y principios de orientación a servicio al proceso de diseño.
4. Paralelamente a las fases de análisis y diseño se ira documentando los pasos de la metodología utilizada para llevar a cabo dicha labor.

Estudio de factibilidad

Factibilidad técnica

El software requerido para el desarrollo de la investigación consiste principalmente en frameworks que soporten los principios enunciados en los principios de SOA, así como también herramientas que permitan documentar el modelado. Así también son necesarios productos de software específicos como por ejemplo un servidor Web, un servidor de aplicaciones y ambientes de desarrollo integrados (IDE).

Todos los tipos de software citados están disponibles para su adquisición vía Internet y poseen licencias de tipo "software libre", estos se describen en el siguiente tabla (tabla número 2):

Tabla 3. Disponibilidad del software candidato.

Fuente: El autor de la investigación.

Categoría del Software	Nombre del software	¿Posee modalidad de licencia de software libre?
Framework de soporte a SOA	J2EE	SI
Servidor Web	Apache	SI
Servidor de aplicaciones de soporte a servlets	Apache Tomcat	SI
Herramienta de modelado con UML	Visual Paradigm	Licencia de prueba y académica
IDE de desarrollo	Eclipse	SI

En cuanto al hardware necesario para el desarrollo de esta investigación; las computadoras y la infraestructura de red disponibles para la Dirección de Admisión de Estudios, supera los requerimientos mínimos del software citado.

Las fuentes de información necesarias para llevar a cabo la labor de análisis son accesibles.

Por lo esbozado, se puede concluir que el proyecto es técnicamente viable.

Factibilidad económica

En lo que respecta a la adquisición de licencias de software para el desarrollo del proyecto, el costo se reduce a cero, ya que los productos a utilizar poseen modalidades de licencia que no ameritan realizar ningún pago. Por otro lado el hardware a utilizar ya se encuentra disponible como parte de los bienes del ente donde se desarrollará la investigación. En cuanto al recurso humano necesario, se tiene previsto que dicha necesidad sea cubierta por el mismo investigador, por lo que no se tiene previsto realizar ninguna erogación por este concepto.

Por lo expuesto se concluye que la investigación es económicamente factible.

Factibilidad operativa

Los usuarios de la solución planteada, en primera instancia, van a ser todos los entes que desarrollen aplicaciones que necesiten de la información suministrada por lo componentes de la arquitectura propuesta. El canal de comunicación entre estos, tal como lo establece la propuesta, va a estar basado en estándares, a los cuales cada día más se adhieren las diferentes herramientas de desarrollo. Por la naturaleza del modelo arquitectónico, debe ser posible su implantación dentro de cualquier ambiente tecnológico y convivir con otras soluciones.

Entonces se puede afirmar que el proyecto es operativamente factible.

Factibilidad psicosocial

Existe receptividad en la mayoría de los desarrolladores, con relación a la implementación de servicios que provean información de forma estandarizada y que ofrezcan la posibilidad de ser utilizados como componentes en otras soluciones. Así mismo el nivel estratégico de la organización donde se implementará la propuesta ha manifestado la necesidad de contar con información fiable y oportuna, para lo cual la palabra clave es la interconexión, y es esta propuesta lo que va contribuir con la plataforma para lograr este objetivo.

Por otro lado el acceso a los servicios de red, por parte de la comunidad UCLA es cada vez mayor, por lo que se puede prever que el alcance de los servicios va a ir en aumento y por ende va a llegar a una mayoría de usuarios.

CAPÍTULO IV

PROPUESTA DEL ESTUDIO

Una vez que se han aplicado los instrumentos de recolección de datos para la investigación, se ha fijado un punto de partida para el desarrollo de la propuesta, la cual consiste en la realización de una serie de actividades guiadas por los objetivos específicos enunciados en el planteamiento del problema y que por ende son conducentes al objetivo general. Los pasos que se llevarán a cabo producirán las bases para una arquitectura de software orientada a servicio (SOA) para la DACE de la UCLA, en donde los componentes de dicha estructura son, por supuesto, los servicios que se entregaran.

La primera fase consiste en realizar un análisis que oriente sobre la selección de la estrategia a seguir para la entrega de los servicios o dicho de otra forma, la estrategia utilizada para llevar a cabo la labor de análisis y diseño orientado a servicio.

La segunda fase es realizar el respectivo análisis orientado a servicio, siguiendo la metodología o lineamientos sugeridos por Earl (2005), en donde se explican, se documentan y se obtiene el entregable de cada paso del proceso de análisis. También en esta etapa del desarrollo de la propuesta se hace especial énfasis en el subproceso de modelado de los servicios (ver figura 2), que es la materia prima para el posterior diseño orientado a servicio.

Como parte final de la propuesta y al culminar con las fases anteriores de análisis se procede al diseño de los servicios seleccionados, aplicando una serie de pasos para producir las definiciones de interfaz de los servicios componentes de la SOA de la DACE de la UCLA. Para este diseño se siguen diferentes pautas, establecidas por la metodología, bien sean comunes o especificas de acuerdo al tipo de servicio a diseñar, como por ejemplo los servicios basados en tarea, los

servicios de aplicación y cualquier otro tipo de servicio que puedan surgir en la el desarrollo de las actividades del guión metodológico.

Una vez que se ha establecido un panorama de lo que será el desarrollo de la propuesta, se comienza entonces con ejecutar los pasos respectivos:

Estrategia de entrega de SOA

Dentro las estrategias planteadas para la entrega de SOA o dicho de otra manera las estrategias para realizar el análisis y diseño orientado a servicio, se encuentran la estrategia de arriba hacia abajo o *top-down*, de abajo hacia arriba o *bottom-up* y la ágil o *meet-in-the-middle*. Cada una ofrece un enfoque diferente para realizar el trabajo con sus respectivos pros y contras y que de acuerdo a la naturaleza del proyecto que se va emprender, se debe seleccionar una como guía para la labor de análisis y diseño.

Para tener elementos que ayuden a la decisión de escoger la estrategia, se procede a ilustrar en un cuadro comparativo de las tres opciones, que viene a ser una adaptación del análisis textual realizado por Earl (2005). Es conveniente aclarar que las características enunciadas aquí son solo enunciados teóricos que se cumplen en condiciones regulares, no extraordinarias, en otras palabras expresan los resultados que se obtendrían al realizar bien el trabajo de análisis y diseño en un entorno ideal.

Tabla 4. Comparación de estrategias de entrega de SOA.

Estrategia	Dimensiones en el proceso de análisis y diseño				
	Calidad en la arquitectura	Profundidad del análisis	Utilización de recursos (tiempo y dinero)	Tiempo de entrega	Reusabilidad y composición
Top-down	Alto	Alto	Alto	Alto	Alto
Bottom-up	Bajo	Bajo	Bajo	Bajo	Bajo
Ágil	Alto	Alto	Medio	Medio	Alto

Fuente: Resumen de explicación textual realizada Earl (2005).

En el caso de la DACE lo deseable es obtener el mayor grado de calidad posible, por lo que la opción Bottom-up se descarta en primera ronda. Así también en el escenario planteado en la presente propuesta, se dispone de una cantidad limitada de recursos para el estudio y se requiere un tiempo de entrega lo más reducido posible. La estrategia ágil surge como una posible opción a seguir, viendo que ofrece un alto grado en las características de calidad en la arquitectura, profundidad en el análisis, reusabilidad y composición, con una inversión de recursos menor con respecto a las demás estrategias.

El anterior razonamiento sugiere que la estrategia a seguir es la ágil, sin embargo hay que recordar que la bibliografía, en este caso Earl (2005), describe las iteraciones que se llevan a cabo utilizando la estrategia ágil, en la fase de diseño (diseñar probar y rediseñar), lo que no es aplicable al presente estudio, ya que el alcance cubre solo las etapas tempranas de diseño; por lo que la estrategia a seleccionar es la *top-down*, por lo menos en el contexto planteado.

Análisis orientado a servicio

En el desarrollo de una SOA el análisis constituye la fase de primordial importancia, por lo cual se debe garantizar que se lleve a cabo de manera ordenada, detallada y que garantice la documentación. Para lograr este objetivo se plantea el seguimiento de un guión como se muestra en la figura 2, donde se ilustra como encuadra el detalle del subproceso de análisis dentro del ciclo de vida de SOA. Como se puede observar se expande el paso de análisis orientado a servicio de la primera columna a tres pasos en la segunda columna para detallar las pasos del análisis y luego se vuelve a expandir el paso de modelado de servicios candidatos hacia la columna tercera donde se muestran todos los pasos que hay que realizar para completar el modelado.

La manera de cómo se presentan cada uno de las etapa del proceso de análisis, es nombrando cada una de estas, explicando en que consisten y produciendo el entregable de acuerdo al caso de estudio (propuesta para la DACE).

Las preguntas principales hechas durante esta fase son:

- ¿Qué servicios necesitan ser construidos?
- ¿Qué lógica necesita ser encapsulada por cada servicio?

El grado al cual se respondan estas preguntas está directamente relacionado con la magnitud del esfuerzo invertido en el análisis. Mucho de los aspectos que se han discutido en las secciones anteriores puede ser parte de esta etapa. Específicamente, la determinación de que capas de servicio construir y como enfocar su entrega, son puntos de decisiones críticos que terminarán formando la estructura de todo el ambiente orientado a servicio.

Todas las metas de realizar análisis orientado a servicio son las siguientes:

- Definir un conjunto preliminar de operaciones de los servicios correspondientes.

- Agrupar operaciones de servicio candidatas en contextos lógicos. Estos contextos representan los servicios candidatos.
- Definir los límites preliminares de los servicios, de tal manera que estos no se solapen con cualquier servicio existente o planeado.
- Identificar la lógica encapsulada con reutilización potencial.
- Asegurar que el contexto de lógica encapsulada es apropiado para su uso pretendido.

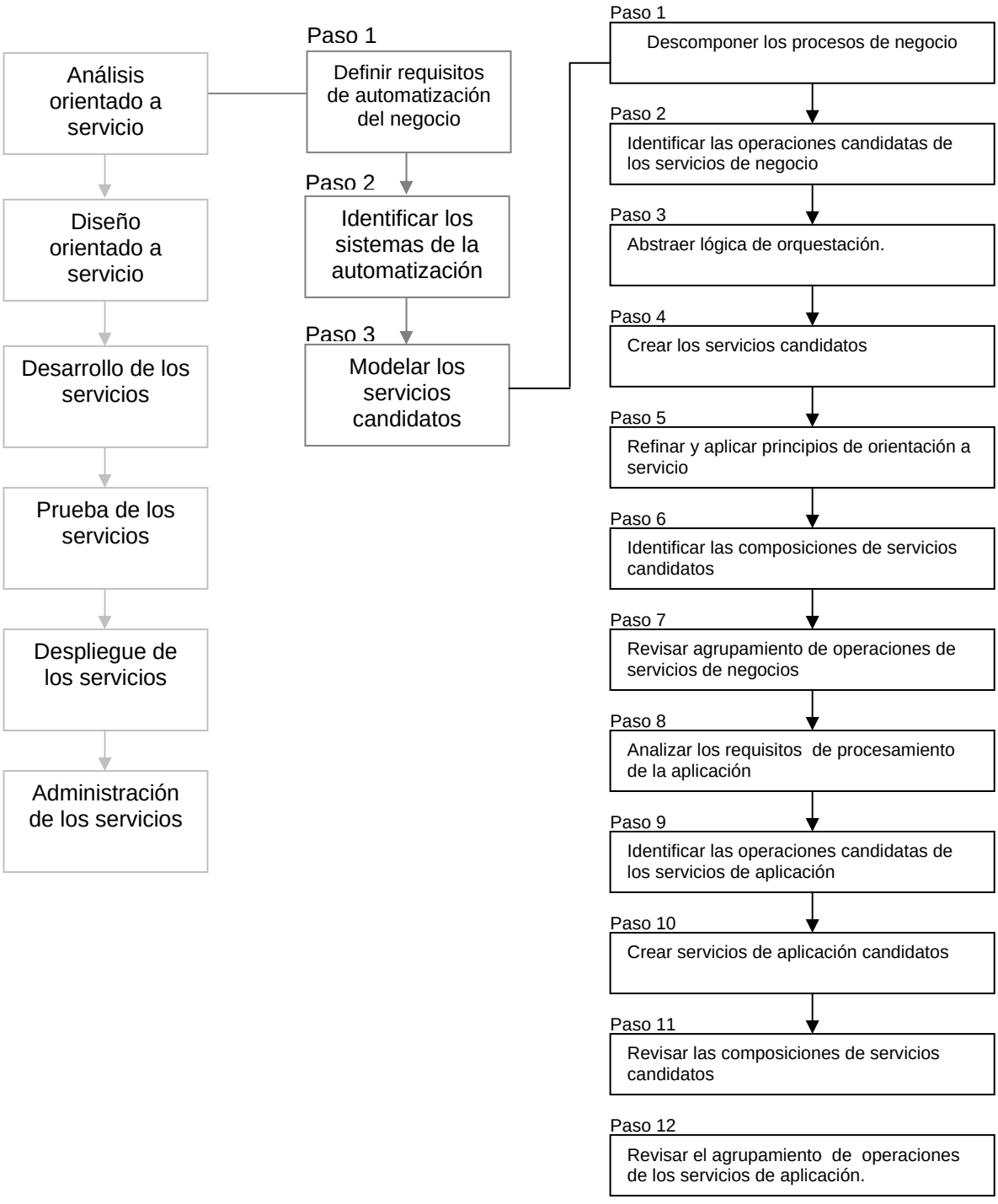


Figura 2. Guión para el análisis orientado a servicio.
Fuente: Earl (2005).

1.- Definir requisitos de automatización del negocio

Este paso consiste en conocer más el domino del problema valiéndose de cualquier técnica, supone que los requisitos del negocio se encuentren suficientemente maduros. En el caso de la DACE, una vez conocidos los resultados de la aplicación de los instrumentos de recolección de información, se determina que no existe una documentación en cuanto a especificación de requisitos y ontología, por lo que se procede a elaborar un glosario de términos, una cita de los requisitos funcionales y no funcionales y la elaboración de artefactos UML, como por ejemplo un diagrama conceptual o diagrama de dominio, y así definir un punto de arranque para el análisis orientado a servicio.

Glosario de términos del dominio.

DACE. Dirección de Admisión y Control de Estudios de la UCLA.

Registro Académico. Unidad organizacional perteneciente a los Decanatos de la UCLA, que tiene dentro de sus funciones gestionar la DATA de los bachilleres inscritos en los diferentes programas o carreras.

Bachiller. Individuo que opta por un cupo de estudio dentro de una carrera ofertada por la UCLA.

CNU. Ente gubernamental que tiene a su cargo la asignación de bachilleres para cursar estudios en las universidades publicas del país.

CENSO. Proceso interno de admisión llevado a cabo por la UCLA, donde participan los bachilleres que no fueron asignados por el CNU y que solicitaron alguna carrera en la UCLA dentro de sus opciones de estudio.

Carrera o programa. Unidad organizacional dentro de los Decanatos de la UCLA, en las cuales los bachilleres aspiran ser admitidos y para las cuales realizan la respectiva inscripción una vez culminado el proceso de admisión.

Decanato. Unidad organizacional dentro de la UCLA que agrupa varios programas o carreras.

Preinscripción vía Internet. Paso previo a la inscripción del bachiller, que confirma el interés de este por el cupo dentro de un programa o carrera dentro de la UCLA. En este paso se le indica al bachiller la posible fecha de inscripción en la carrera.

Inscripción vía Internet. Paso posterior a la preinscripción que se realiza en un lapso de tiempo establecido para cada Decanato o Carrera, en donde se le informa de la fecha y hora de consignación de documentos ante la DACE, para luego remitirlo al respectivo Decanato.

Inscripción por oficina. Apertura de expediente. Es el paso que se lleva a cabo en el que Bachiller, una vez realizado la preinscripción e inscripción, consigna todos los recaudos exigidos ante la DACE, es emitida la planilla de inscripción y completa el proceso de inscripción en lo que respecta a la DACE.

Convocatoria de inscripción. Documento emitido por la aplicación Web de la DACE, donde se especifican los recaudos a consignar por el Bachiller, así como también la fecha y hora de asistencia a la oficina de la DACE por parte del Bachiller.

Comprobante de Preinscripción. Documento emitido por la aplicación Web de la DACE, donde se especifica la confirmación del cupo en la UCLA y una posible fecha de inscripción en el portal Web de la DACE.

DATA de asignados por el CNU. Colección de registros suministrada por el CNU donde se encuentran los datos de los bachilleres asignados a las diferentes carreras de la UCLA, para un año específico, con sus respectivos datos personales y académicos tales como: el promedio de notas de Bachillerato (PN), el índice académico (IA), el promedio transformado de notas(PTPN), el promedio transformado de razonamiento verbal (PTRV), el promedio transformado de razonamiento matemático (PTRM), las carreras que solicitó el bachiller y la carrera donde fue asignado; esta ultima en la UCLA. Esta DATA es suministrada en diferentes formatos, Excell, Texto, etc. Esta constituye el insumo para realizar el proceso de preinscripción en la UCLA por la modalidad de CNU.

DATA de no asignados por el CNU. Colección de registros suministrada por el CNU donde se encuentran los datos de los bachilleres que no fueron asignados por el CNU para un año específico y que en algunas de las opciones solicitaron UCLA. Estos registros están compuestos por los datos personales del bachiller como por indicadores tales como: el promedio de notas de bachillerato (PN), el índice académico (IA), el promedio transformado de notas (PTPN), el promedio transformado de razonamiento verbal (PTRV), el promedio transformado de razonamiento matemático (PTRM), y las carreras que el bachiller solicito como opción. Esta DATA es suministrada en diferentes formatos, Excell, texto, etc. Esta constituye el insumo para el censo en la UCLA.

PN. Promedio de Notas alcanzado por el Bachiller desde séptimo grado de educación básica hasta primer año de educación diversificada, el cual es referencia para la asignación de bachilleres por censo dentro de la UCLA.

IA. Índice que establece el CNU para la asignación de los bachilleres el cual es compuesto por el promedio de notas y los resultados de pruebas aplicadas por el CNU a los bachilleres o Estudiantes.

Cohorte de asignación. Agrupación de Bachilleres que tienen en común la carrera a la cual fueron asignados, la modalidad de asignación (CNU o CENSO) y cuya identificación se basa en el año y el periodo de de asignación (1 o 2). Ej. 2007-1 o 2007-2.

Lapso de Inscripción. Es el lapso definido por el Decanato o el Programa en el cual les corresponde inscribirse una cohorte de asignación. Ej. en el lapso de inscripción 2007.1, le corresponde inscribirse los bachilleres de la cohorte de asignación 2006-2 del censo.

Especificaciones funcionales.

- Se debe proveer información de los bachilleres asignados e inscritos en la DACE a los diferentes Decanatos de la UCLA.
- La información suministrada debe estar compuesta por datos personales de los bachilleres, así como también de los datos académicos.
- Debe existir la posibilidad de que la información ofrecida este disponible para otros entes de la UCLA.
- Se debe colocar a disposición de los Decanatos la información referente a los procesos de admisión, como el total de bachilleres inscritos y con aperturas de expediente en cada carrera y los lapsos de inscripción respectivamente.

Especificaciones no funcionales.

- Se debe proveer información estandarizada principalmente a los Decanatos de la UCLA, hablando un mismo lenguaje para la comunicación.
- La información suministrada, al estar estandarizada debe ser de utilidad y de acceso sencillo a otras entidades organizacionales de la UCLA.
- Los servicios desarrollados deben servir como componentes a otros servicios y a cualquier aplicación que se encuentre dentro del dominio del negocio de la UCLA.
- La información debe estar disponible por medio de una estructura de red (Intranet o Internet).

Diagrama conceptual

Como parte del primer paso del desarrollo de la metodología de orientación a servicio, cuyo propósito es conocer mejor el dominio del problema, una vez que se conocen los conceptos involucrados y sus relaciones se puede hacer uso de artefactos tales como el diagrama conceptual o diagrama de dominio, tal como se describe en la figura 3.

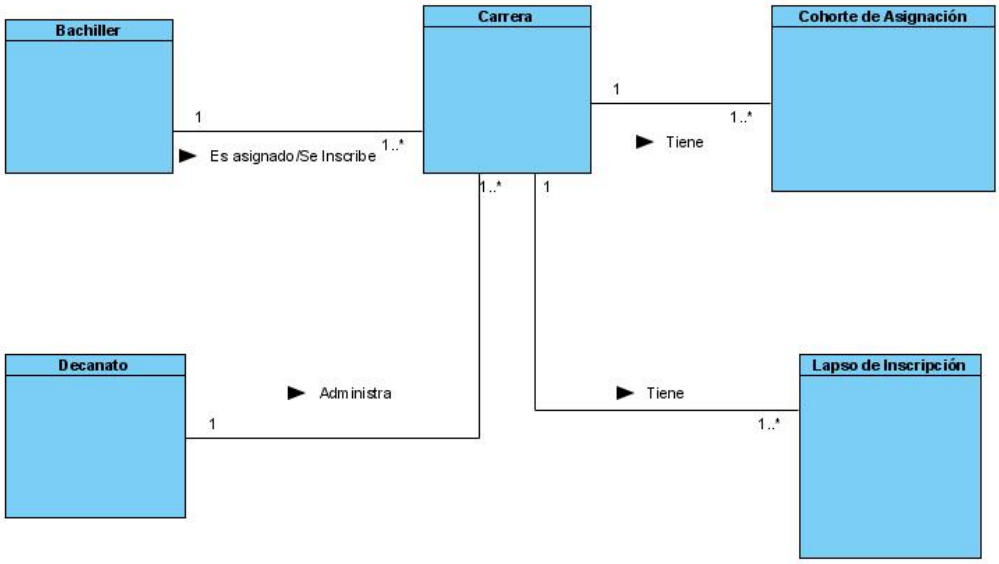


Figura 3. Diagrama conceptual de parte del dominio a tratar.
Fuente: El autor de la investigación.

2.- Identificar sistemas de automatización existentes.

Con la finalidad de identificar los sistemas involucrados y posiblemente afectados por la solución propuesta se procede a realizar un inventario de los existentes, con el nombre y una breve descripción de sus funcionalidades y la tecnología que utilizan.

Sistema para el registro y control de la Admisión de la UCLA. Se encarga de registrar los eventos relacionados con los bachilleres que se preinscriben, se inscriben y luego abren expediente en la DACE, así como también emitir informes relacionados con la información que capta, como por ejemplo listados de preinscritos por modalidad (Censo y CNU) para una cohorte

especifica, totales de inscritos por carrera y Decanato. También gestiona datos de planificación como por ejemplo calendarios de inscripción. Sus interfaces de usuario están desarrolladas en ambiente Web con lenguajes como ASP, C#, VB6 y los datos gestionados por el manejador de base de datos MS SQL Server. Actualmente este sistema provee información vía Web a los Decanatos y otras dependencias de la UCLA sobre la información que procesa en formato HTML y texto. Las funcionalidades del sistema mencionado se pueden resumir en el siguiente diagrama de ucase (Figura 4)

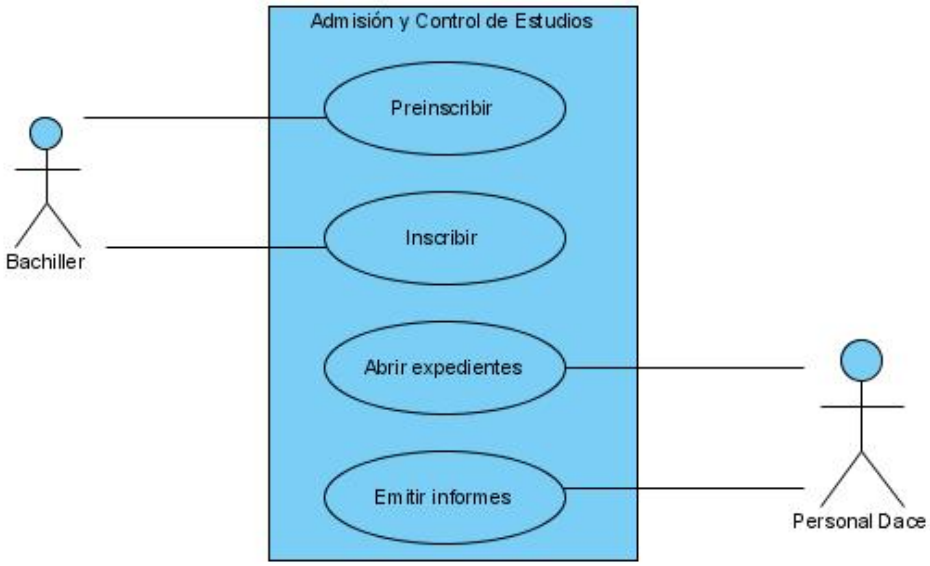


Figura 4. Diagrama de ucase del sistema de admisión y control de estudios.
Fuente: El autor de la investigación.

Sistemas de registro académico de los Decanatos. En cada Decanato existen aplicaciones para la gestión del proceso administrativo que se produce cuando el bachiller ingresa a la UCLA y prosigue su formación académica. Registra eventos de inscripción, trascripción de calificaciones, inclusión de asignaturas, etc. Así como también emite informes como boletines de información, actas de calificaciones, etc. La tecnología utilizada en los Decanatos difiere en cada uno de estos, dentro de los formatos de BD se encuentran XBase, MS SQLServer, ADABAS, en la herramientas de desarrollo se encuentran VB6,

Natural y Clipper. Algunas de las funcionalidades se expresan en el siguiente diagrama de ucase (Figura 5).

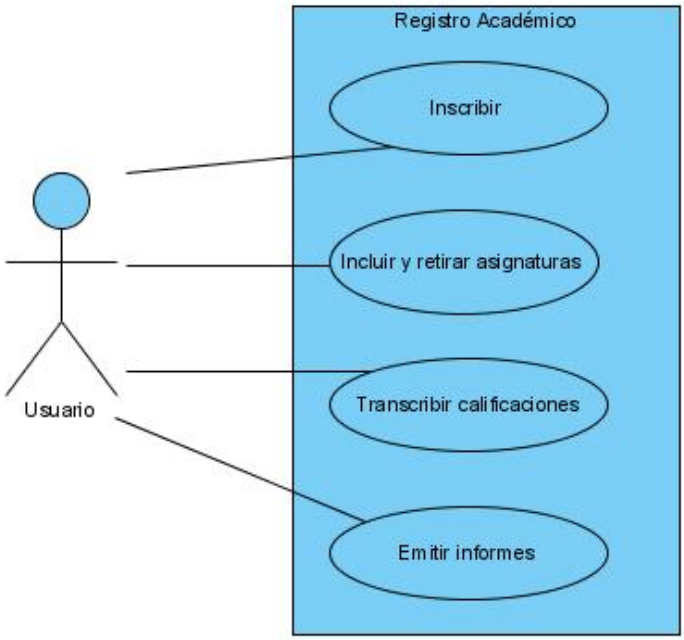


Figura 6. Diagrama de ucase del sistema de registro académico.
Fuente: El autor de la investigación.

Los sistemas mencionados no cumplen en su totalidad con los requisitos funcionales y no funcionales descritos anteriormente, por lo que no está planteado mantener una lógica de aplicación legada, sino crear nueva lógica de aplicación con tecnología posterior a la existente.

3.- Modelar servicios candidatos.

Para el modelado de los servicios candidatos se proceden con los siguientes pasos:

3.1.- Descomponer los procesos de negocio.

Antes de comenzar a descomponer los procesos de negocio a partir de los diagramas de flujo de datos, es conveniente realizar un análisis por medio de texto del principal proceso a abordar, para dar una idea general de cómo es el funcionamiento del negocio en la DACE, así como también exponer dicho proceso a través de una diagrama de actividad en UML, como se puede observar en la figura 6.

El proceso se inicia cuando el Bachiller es asignado para cursar una carrera en la UCLA por cualquiera de las modalidades establecidas (CNU, censo u otras), primero el Bachiller debe proceder a realizar la preinscripción en el portal Web de la UCLA, con lo que obtiene un comprobante con una fecha posible de inscripción. Luego en una fecha establecida para la cohorte de asignación del estudiante, el bachiller realiza una inscripción vía Web, donde obtiene una convocatoria con la fecha y la hora para la consignación de documentos y la correspondiente apertura de expediente en la DACE.

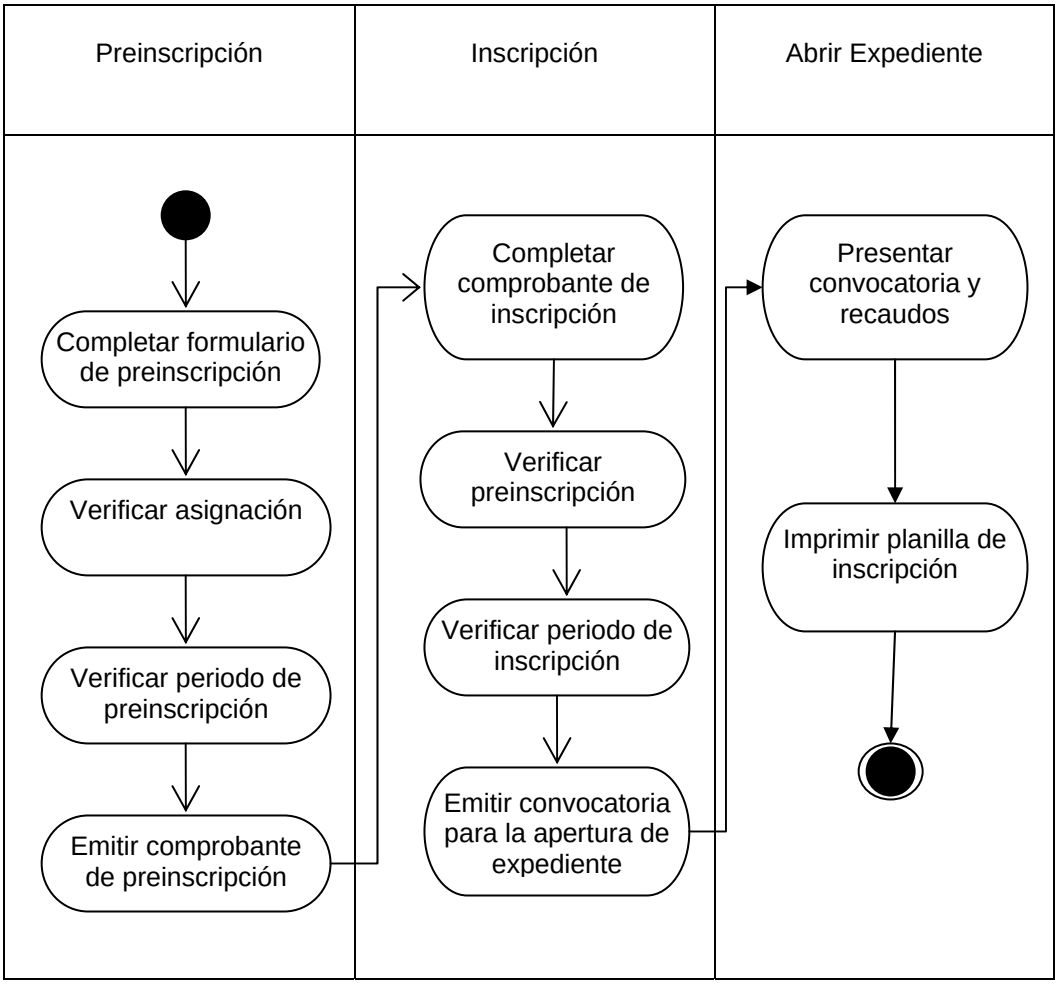


Figura 6. Diagrama de actividad descriptivo del proceso de inscripción del Bachiller.
Fuente: El Autor de la investigación.

Esta secuencias de actividades descritas producen información que es gestionada por el sistema de control de estudios y que su uso más frecuente es la emisión de informes que contienen cuadros de resumen de los proceso de inscripción por cohortes de asignación.

Este proceso macro se puede descomponer luego en los procesos de inscripción, preinscripción y emisión de cuadros resumen de los procesos de admisión e inscripción. Para cada uno de estos procesos se procede a conseguir los diagramas de flujo de datos tal como fueron documentados originalmente y realizar la respectiva descomposición de los procesos de negocio para ir identificando las operaciones candidatas que luego se agruparan de acuerdo al contexto y de esta manera obtener los servicios candidatos. El primer diagrama a abordar es el de inscripción del Bachiller, tal como se ilustra en la figura 7, con la respectiva descomposición expresada textualmente luego de representación gráfica.

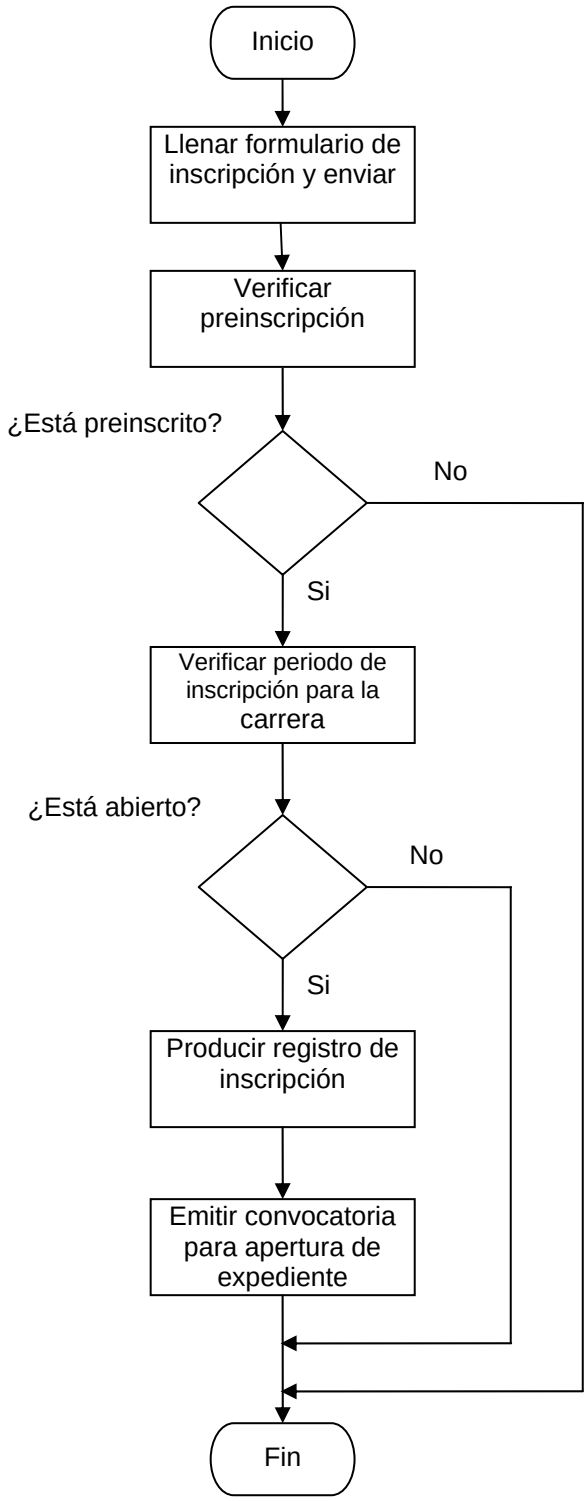


Figura 7. Diagrama de flujo de datos para inscribir bachiller.
Fuente: El autor de la investigación

La descomposición se sugiere de la siguiente manera:

- Llenar formulario de inscripción
- Enviar formulario de inscripción.
- Verificar preinscripción, si el Bachiller no esta preinscrito finaliza el proceso.
- Consultar cohorte de asignación del bachiller.
- Verificar el periodo de inscripción para la cohorte de asignación del bachiller, si no esta abierto finaliza el proceso.
- Producir registro de inscripción del Bachiller.
- Emitir convocatoria para el Bachiller que ha llenado el formulario por Internet.

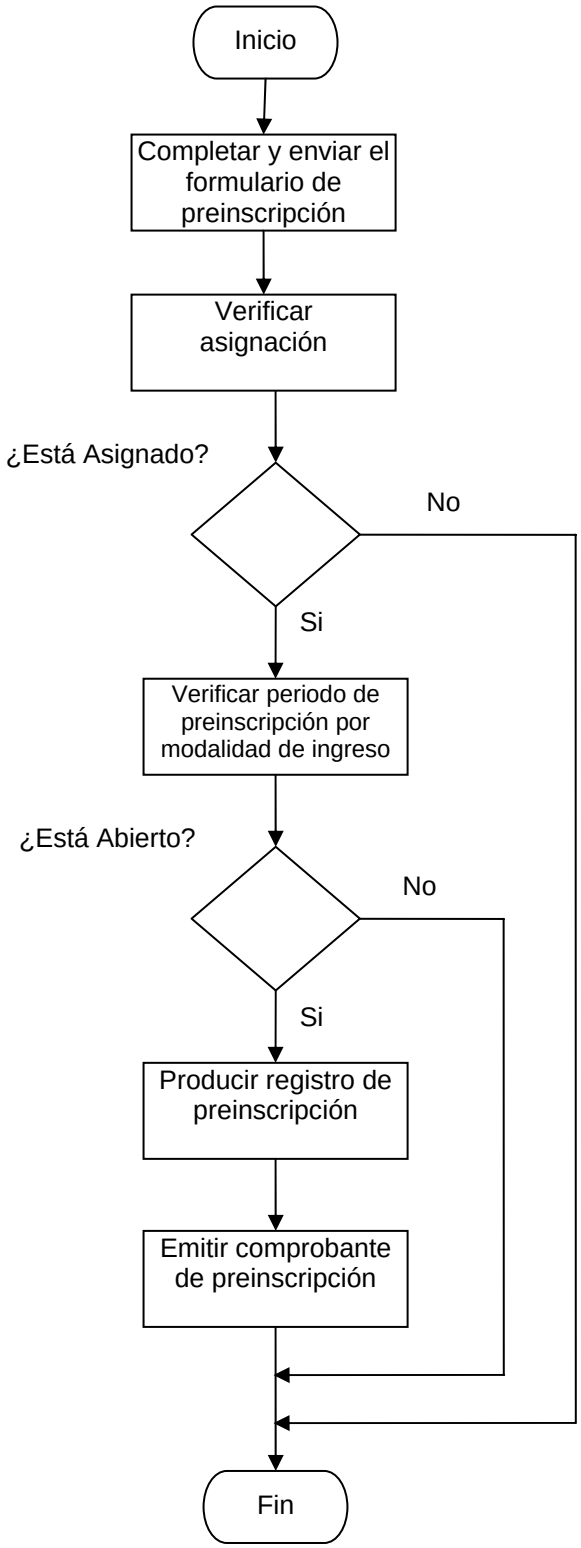


Figura 8. Diagrama de flujo de datos para preinscribir de Bachiller.
Fuente: El autor de la investigación.

La descomposición se sugiere de la siguiente manera:

- Completar formulario de preinscripción.
- Enviar formulario de inscripción.
- Verificar asignación, si no esta preinscrito finaliza el proceso.
- Consultar modalidad de ingreso del Bachiller.
- Verificar el periodo de preinscripción para la modalidad de ingreso, si no esta abierto el periodo finaliza el proceso.
- Producir registro del bachiller preinscrito.
- Emitir comprobante de preinscripción.

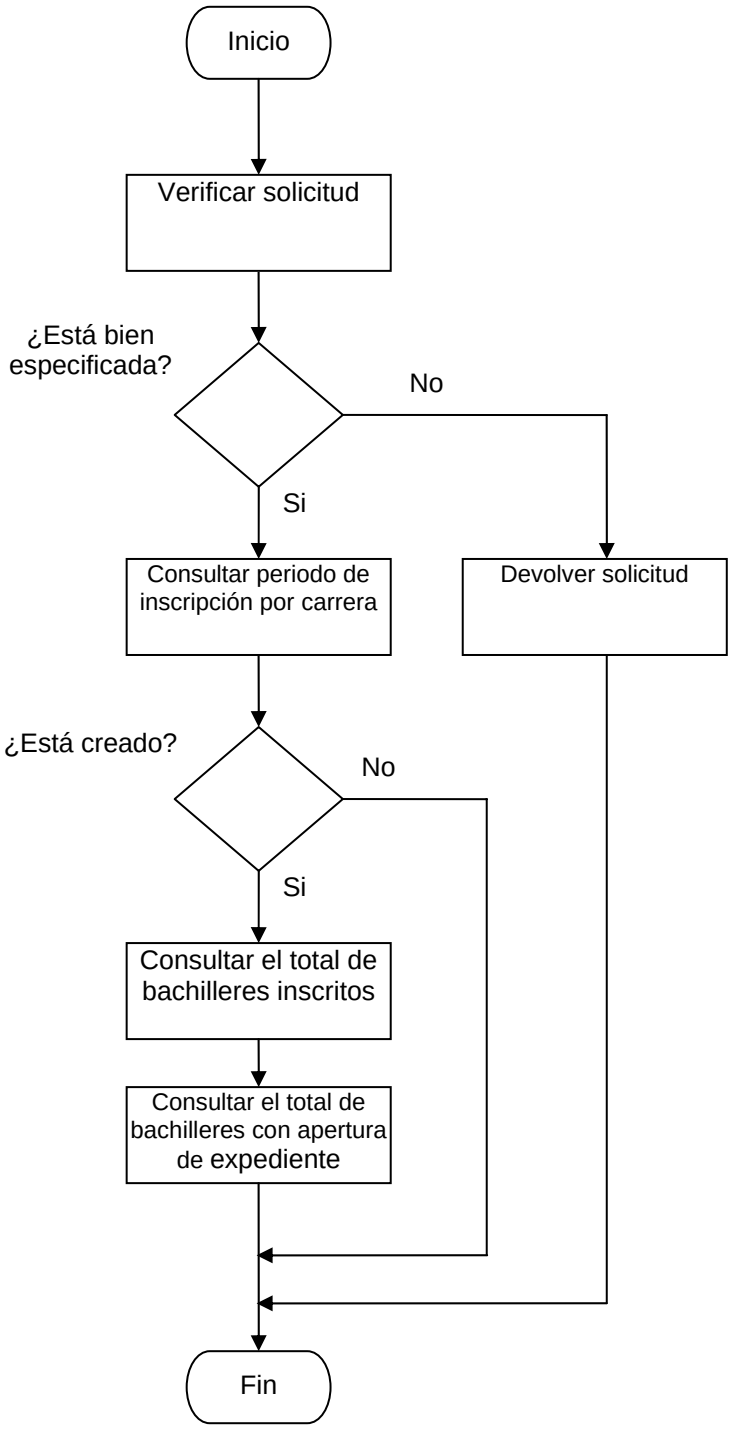


Figura 9. Diagrama de flujo de datos para Procesar solicitud de información de resumen del proceso de inscripción
Fuente: El autor de la investigación.

La descomposición se sugiere de la siguiente manera:

- Verificar solicitud de resumen de admisión, si no esta bien especificada se devuelve la solicitud y finaliza el proceso.
- Devolver la solicitud.
- Consultar periodo de inscripción para la carrera solicitada, si no está creado finaliza el proceso
- Consultar el total de bachilleres inscritos por carrera y lapso de inscripción.
- Consultar el total de bachilleres con apertura de expediente por carrera y lapso de inscripción.
- Publicar el cuadro del resumen de admisión por carrera y lapso de inscripción.

3.2.- Identificar las operaciones candidatas de los servicios de negocio.

Algunos pasos dentro del proceso de negocio, son descartados en primera instancia como parte potencial de la lógica que encapsularan los servicios candidatos. Dentro de dicho pasos se encuentran procesos manuales o algunos pasos realizados por la lógica legada, para los cuales la encapsulación en servicios no es una opción.

Después de revisar los pasos que se obtienen de la descomposición del proceso en el paso anterior, se procede a remover los que no formaran parte de la solución orientada a servicio. En este caso la manera de ilustrar esta actividad es dando el formato de tachado al texto que identifica los pasos del proceso.

Inscribir Bachiller:

Para el proceso de inscripción del Bachiller se tiene:

- ~~Llenar formulario de inscripción~~. (Paso manual realizado por el Bachiller).
- ~~Enviar formulario de inscripción~~. (Paso manual realizado por el Bachiller).
- Verificar preinscripción, si el Bachiller no esta preinscrito finaliza el proceso. (Realizado por un componente desarrollado a la medida, el cual si se mantiene o es sustituido, puede ser un candidato para operación).
- Consultar cohorte de asignación del bachiller. (Igual al anterior).
- Verificar el periodo de inscripción para la cohorte de asignación del bachiller, si no esta abierto finaliza el proceso. (Igual al anterior).
- Producir registro de inscripción del Bachiller. (Igual al anterior).
- Emitir convocatoria para el Bachiller que ha llenado el formulario por Internet. (Igual al anterior).

Luego se continúa con los demás procesos, en el caso de estudio específico, el que sigue es el proceso de preinscripción del Bachiller.

Preinscribir Bachiller:

- ~~Completar formulario de preinscripción~~. Paso manual realizado por el Bachiller.
- ~~Enviar formulario de preinscripción~~. Paso manual realizado por el Bachiller.
- Verificar asignación, si no esta asignado finaliza el proceso. (Realizado por un componente hecho a la medida, que de mantenerse o sustituirse es un candidato a operación).
- Consultar modalidad de ingreso del Bachiller. (Igual al anterior).
- Verificar el periodo de preinscripción para la modalidad de ingreso, si no esta abierto el periodo finaliza el proceso. (Igual al anterior).
- Producir registro del bachiller preinscrito. (Igual al anterior).
- Emitir comprobante de preinscripción. (Igual al anterior).

Procesar solicitud de información de resumen del proceso de inscripción

- ~~Verificar solicitud de resumen de admisión, si no esta bien especificada se devuelve la solicitud y finaliza el proceso~~. (Paso manual realizado por personal de la DACE).
- ~~Devolver la solicitud~~. (Paso manual realizado por el personal de la DACE).
- Consultar periodo de inscripción para la carrera solicitada, si no está creado finaliza el proceso. (Función realizada por un componente de software especializado, que de mantenerse o sustituirse, es un candidato para una operación de servicio).
- Consultar el total de bachilleres inscritos por carrera y lapso de inscripción. (Igual al anterior).

- Consultar el total de bachilleres con apertura de expediente por carrera y lapso de inscripción. (Igual al anterior).
- ~~Enviar el cuadro del resumen de admisión por carrera y lapso de inscripción~~. (Paso manual realizado por el personal de la DACE).

3.3.- Crear los servicios candidatos.

Esta actividad consiste en revisar los pasos que subsisten y agruparlos en contextos lógicos para formar los servicios candidatos. Estos servicios candidatos puede estar basados en tareas o entidades y cada opción ofrecer sus respectivas ventajas y desventajas. Observando esta situación y previendo que el presente estudio no persigue dentro de sus objetivos realizar una comparación entre estas dos modalidades, se optó por un modelado centrado en tareas, vista la menor cantidad de tiempo que hay que invertir en análisis, en comparación con la agrupación basada en entidad.

Los contextos lógicos según el enfoque de agrupación serían los procesos (ver figura 10):

Servicio de inscripción.

- Verificar preinscripción de Bachiller.
- Consultar cohorte de asignación de Bachiller.
- Verificar periodo de inscripción de Bachiller.
- Producir registro de inscripción de Bachiller.
- Emitir convocatoria de Bachiller inscrito.

Servicio de preinscripción.

- Verificar asignación de Bachiller.
- Consultar modalidad de ingreso de Bachiller.
- Verificar periodo de preinscripción por modalidad de ingreso.
- Producir registro de preinscripción de Bachiller.

- Emitir comprobante de preinscripción de Bachiller.

Servicio de resumen de proceso de inscripción

- Consultar periodo de inscripción por carrera.
- Consultar total de bachilleres inscritos.
- Consultar total de aperturas de expediente.

Se tiene de esta manera la primera ronda de servicios candidatos.

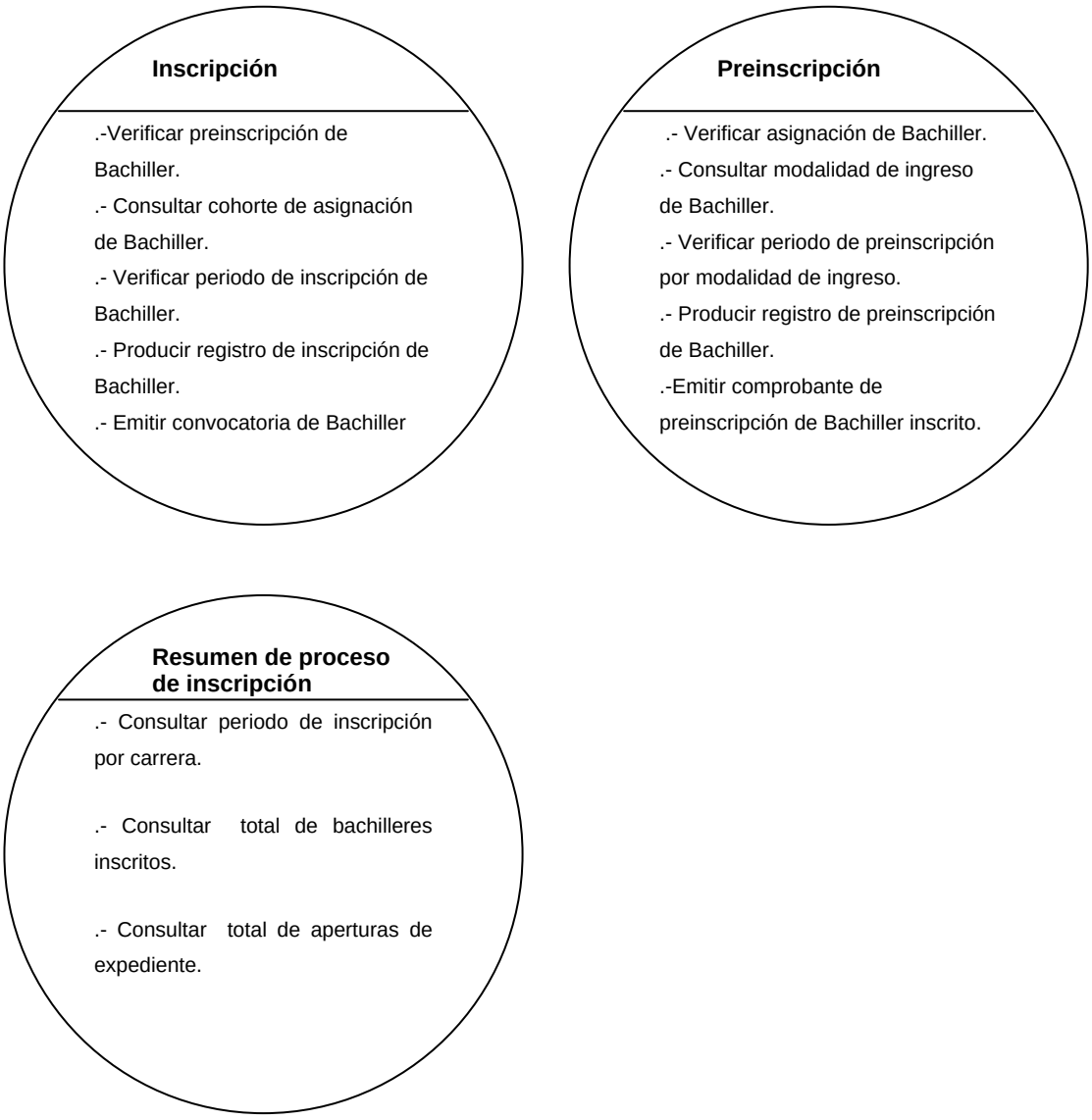


Figura 10. Primera ronda de servicios candidatos.
Fuente: El autor de la investigación.

3.4.- Refinar y aplicar principios de orientación a servicio.

Hasta ahora simplemente se han agrupado pasos de proceso derivados de un proceso de negocio existente. Cómo se puede recordar, se han identificado cuatro principios claves que no se proveen intrínsecamente a través del uso de servicios Web:

- Reusabilidad.
- Autonomía.

- Sin estado.
- Con capacidad de ser descubiertos (discoverability).

De estos cuatro, solo los dos primeros son importantes en la etapa de modelado de servicio, por lo tanto, el mayor interés en este paso es asegurar que cada operación de servicio candidata sea potencialmente reusable y tan autónoma como sea posible.

De esta manera se procede a revisar las agrupaciones de las operaciones que son las que conforman cada servicio candidato respectivamente y proceder a aplicar los criterios de orientación a servicio importantes en esta etapa de análisis. Así pues se pueden visualizar las siguientes modificaciones:

En el servicio candidato de inscripción, la operación de verificar preinscripción de Bachiller se puede mover al servicio de preinscripción y de esta manera ir dándole un grado de especialización al servicio de preinscripción.

Se puede proponer la creación de un servicio especializado en admisión y mover a esta nueva agrupación la operación de consultar cohorte de asignación.

Del servicio de preinscripción se puede remover la operación de verificar asignación y delegarla al servicio de asignación.

La modalidad de ingreso es igual a la modalidad de asignación, por lo que la operación de consultar modalidad de ingreso se puede mover hacia el servicio de asignación con el nombre de "consultar modalidad de asignación".

Al revisar que para verificar el periodo de inscripción es necesario que siempre se haya realizado el paso anterior de chequear la preinscripción del bachiller, estas dos operaciones de pueden fusionar en una sola operación de verificar preinscripción.

Luego al observar el servicio de asignación, también se puede llegar a la conclusión que para consultar la modalidad de asignación de debe verificar que el Bachiller este asignado, por lo que estas dos operaciones se resumen en solo una

de verificar asignación. A esta última operación se le asigna la responsabilidad de consultar la modalidad de asignación del Bachiller que está verificando.

El servicio de inscripción está equipado con dos operaciones candidatas que son producir registro de inscripción y emitir convocatoria de de Bachiller inscrito, al analizar esta situación se puede concluir que siempre se van a realizar cuando ocurra el evento de la inscripción, por lo que se decide fusionarlas en una sola de inscribir de Emitir inscripción.

La observación anterior hecha al servicio de inscripción es válida para el servicio de preinscripción que produce un registro y comprobante a la vez. La vista gráfica del resultado de este paso se puede observar en la figura 11.

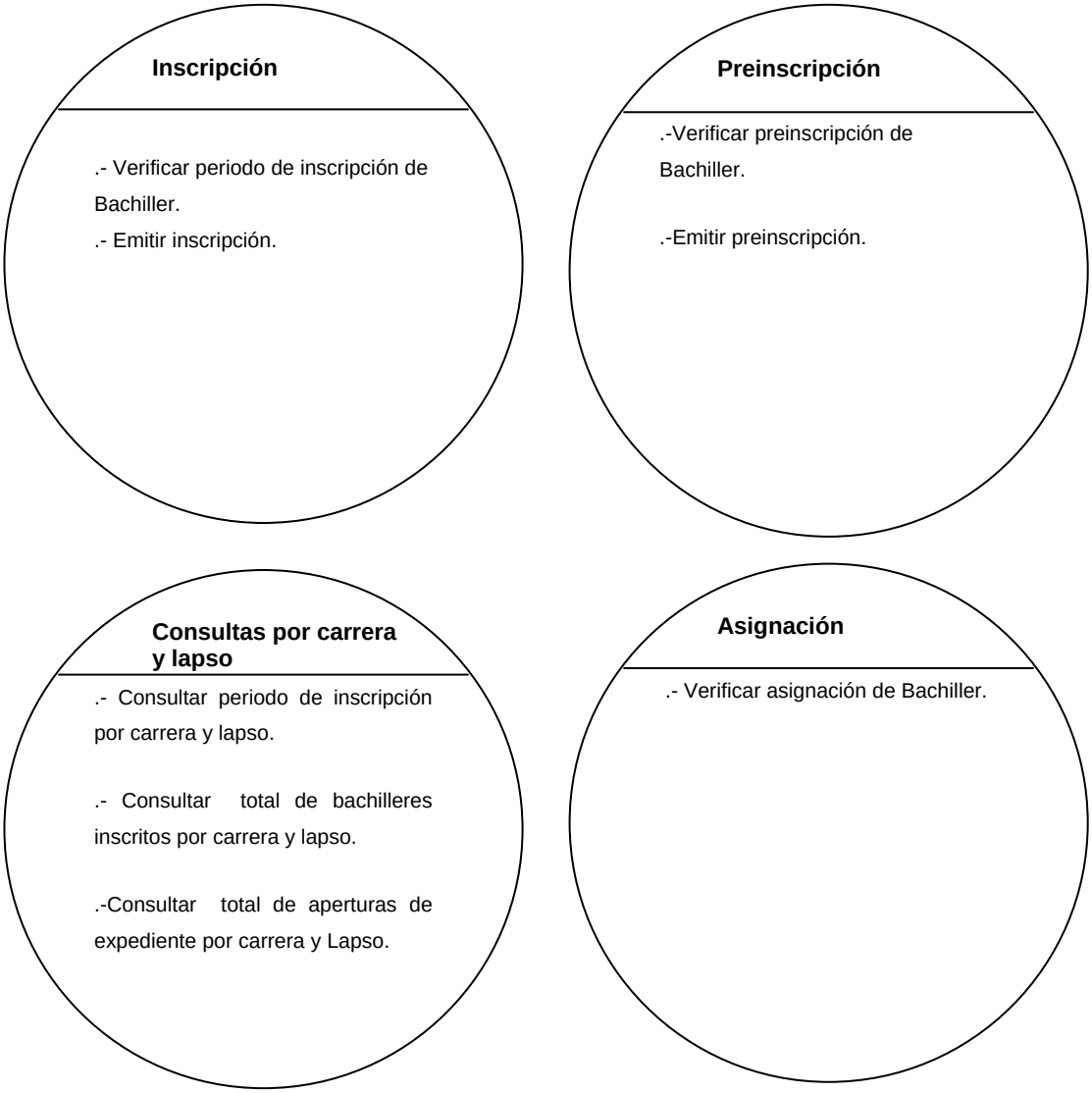


Figura 11. Conjunto de servicios candidatos revisados.
Fuente: El autor de la investigación.

3.5.- Identificar las composiciones de servicios candidatos.

Identificar un conjunto de los escenarios más comunes que pueden tener lugar dentro de las fronteras del proceso de negocios. Para cada escenario, se deben seguir los pasos del proceso requeridos, como se han establecido hasta ahora.

En esta actividad se lleva a cabo lo siguiente:

- Da una buena idea en cuanto a que tan apropiado es agrupar los pasos del proceso.
- Demuestra la relación potencial entre las capas de orquestación y de servicio de negocio.
- Identifica composiciones de servicios potenciales.
- Destaca cualquier lógica de workflow o pasos del proceso perdidos.

Nótese también que cualquier capa de servicio que se establece en este punto, aun esta en fase preliminar y sujeta a revisiones durante el proceso de diseño.

Anteriormente en el paso correspondiente se identificaron una serie de servicios candidatos que conforman de manera preliminar las capas de servicio de negocio y de aplicación.

Los servicios establecidos hasta ahora son:

- Servicio de inscripción.
- Servicio de preinscripción.
- Servicio de asignación.
- Servicio de resumen de proceso de admisión.

Cada uno de estos servicios candidatos representa lógica genérica, reusable y agnóstica (neutral). En otras palabras, estos pueden ser clasificados como

servicios de aplicación candidatos. En conjunto estos establecen preliminarmente una capa de servicio de aplicación.

Hay que recordar que el rol principal de los servicios de negocio centrados en tarea es actuar como controladores, componiéndose de servicios de aplicaciones para llevar a cabo la lógica de negocio requerida. El servicio de inscripción del Bachiller, el de preinscripción del Bachiller y el servicio de consulta de resumen de admisión conforman preliminarmente una capa principal de servicios de negocio y contiene todo lo de la lógica de proceso requerido para componerse de servicios candidatos subyacentes de aplicación. (Figura 12, 13 y 14).

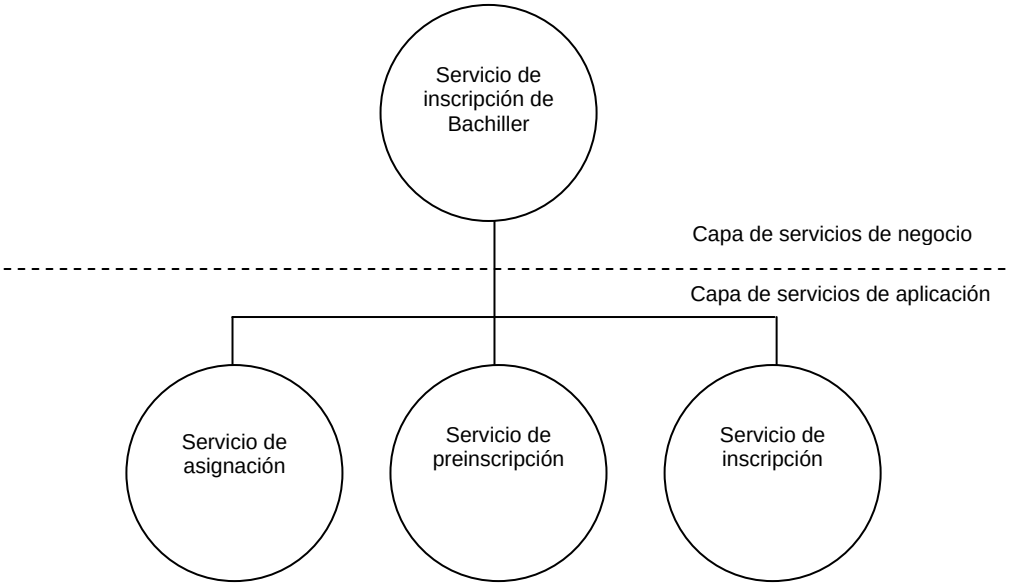


Figura 12. Una composición de ejemplo que representa el proceso inscripción del Bachiller.
Fuente: El autor de la investigación.

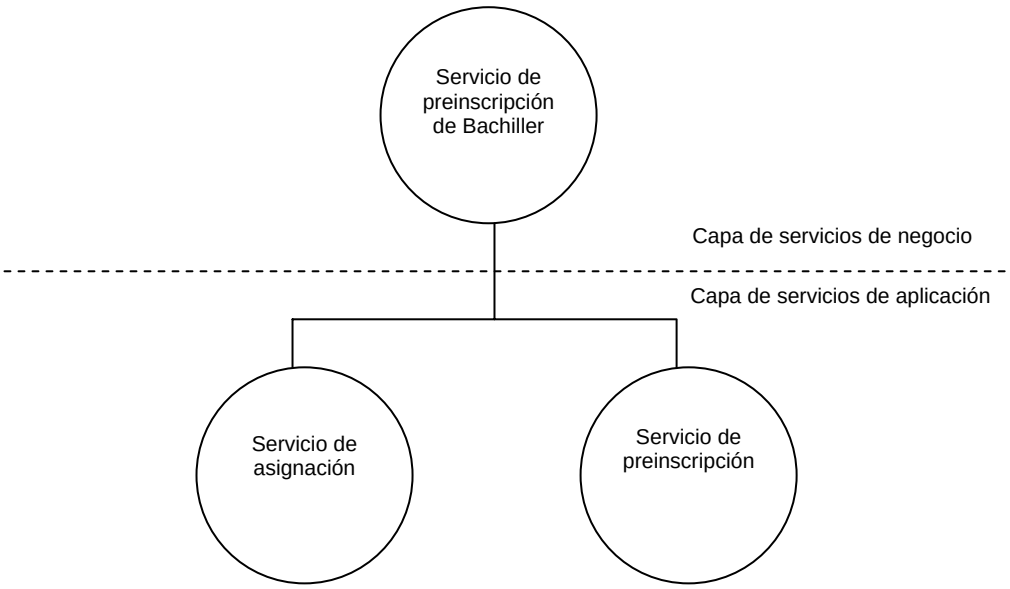


Figura 13. Una composición de ejemplo que representa el proceso de preinscripción del Bachiller.
Fuente: El autor de la investigación.

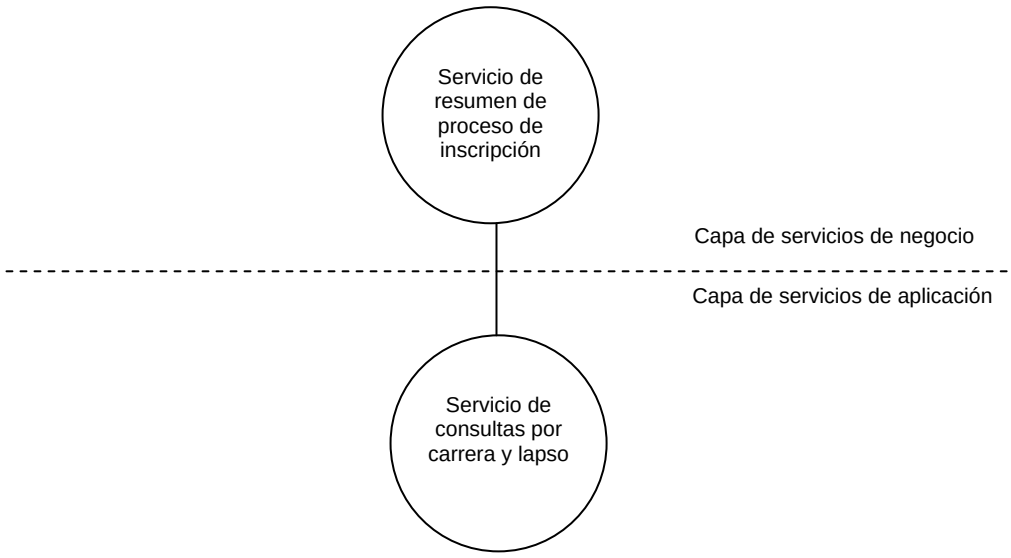


Figura 14. Composición que representa el proceso de elaborar resumen de proceso de inscripción.
Fuente: El autor de la investigación.

3.6.- Revisar agrupamiento de operaciones de servicios de negocios.

Basándose en los resultados del ejercicio de composición en el paso 6 (3.5), se vuelve a visitar el agrupamiento de los pasos del proceso de negocio y revisar la organización de operaciones de servicio candidatas cuantas veces sea necesario. No es inusual consolidar o crear nuevos grupos (servicios candidatos)

en este punto. Esta actividad y las que siguen cobran importancia cuando los procesos son grandes y complejos.

3.7.- Analizar los requisitos de procesamiento de la aplicación.

Al final del paso 6 (3.6), se debe haber creado una vista centrada en negocios de las capas de servicios. Esta vista, normalmente podría consistir de ambos servicios, tanto de los servicios candidatos de negocio como de los de aplicación, pero hasta el momento la atención se ha enfocado en representar la lógica del proceso de negocio.

Esta próxima serie de pasos son opcionales y más convenientes para los procesos de negocio complejos y grandes ambientes orientados a servicio. Se requiere de un estudio más a fondo de los requerimientos de proceso subyacentes de todos los servicios candidatos, para abstraer cualquiera de los servicios candidatos adicionales centrados en tecnología desde esta vista y que complementará una capa preliminar de servicios de aplicación. Para lograr esto, para cada paso del proceso identificado hasta ahora requiere de un mini análisis.

Específicamente, lo que se necesita determinar es:

- Que lógica de aplicación subyacente se necesita ejecutar para procesar la acción descrita por la operación candidata.
- Si la lógica de aplicación requerida ya existe o si se necesita desarrollar una nueva.
- Si la lógica de aplicación requerida abarca las fronteras de la aplicación. En otras palabras, ¿Se requiere más de un sistema para completar esta acción?

Nótese que la información recopilada durante el paso 2 (3.2) del principal proceso de análisis orientado a servicio será referenciada en este punto.

El proceso de negocio del caso de estudio de ninguna manera es complejo, y ya se han identificado todo lo de los servicios candidatos que representan la lógica de aplicación preliminar. Por ende, los pasos siguientes no se requieren.

Diseño orientado a servicio

Si se ha dicho que la etapa más importante en la orientación a servicio es el análisis orientado a servicio, el diseño viene a ser la segunda en importancia. Para realizar el diseño orientado a servicio se recomienda un orden en el diseño de cada uno de los tipos de servicios, bien sean basados en entidad o basados en tarea y de los servicios de aplicación y de negocios. La figura 15 muestra el orden sugerido para el diseño de los servicios y la ubicación del diseño en el proceso total de entrega de SOA.

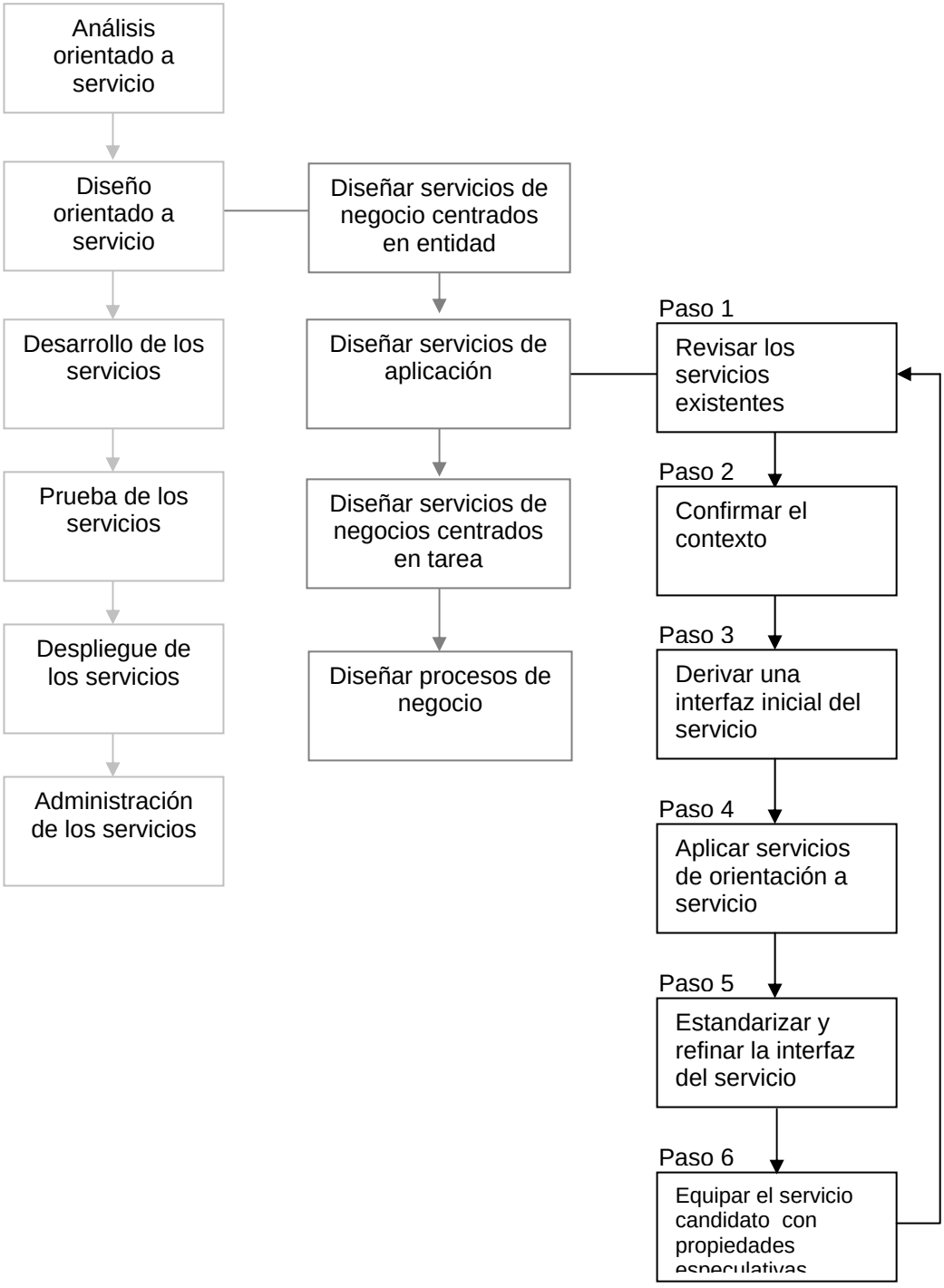


Figura 15. Pasos en el diseño de orientación a servicio, con expansión del paso para el diseño de servicios de aplicación.
Fuente: Earl(2005).

Al realizar una revisión de los servicios candidatos que se obtienen de la etapa de modelado, específicamente de la composición de los servicio, se tienen los siguientes:

- Servicio de inscripción de Bachiller (Servicio basado en tarea).
- Servicio de asignación (Servicio de aplicación).
- Servicio de preinscripción (Servicio de aplicación).
- Servicio de inscripción (Servicio de aplicación).
- Servicio de preinscripción de Bachiller (Servicio basado en tarea).
- Servicio de resumen de proceso de inscripción (Servicio basado en tarea).
- Servicio de consultas por carrera y lapso (Servicio de aplicación).

Los servicios de aplicación son los primeros en abordarse para su diseño. Los servicios de aplicación son el caballo de batalla de SOA. Estos representan la subcapa más baja de la capa de servicios compuesta, responsable de llevar a cabo cualquier paso del proceso demandado por las capas de negocio o de orquestación.

A diferencia de los servicios de capas centradas en negocios, el diseño de servicios de aplicación no requiere de experticia en análisis del negocio. La capa de servicios de aplicación es solamente una abstracción del ambiente técnico de la organización, mejor definida por aquellos que comprenden este ambiente de forma amplia.

Debido a que deben tomarse en cuenta muchas consideraciones del mundo real y de la tecnología específica, los servicios de aplicación puede ser el tipo de servicio más difícil de diseñar. Por otro lado el contexto establecido por estos servicios puede cambiar constantemente, la tecnología puede cambiar o ser reemplaza, y como la lógica de aplicación relacionada se puede que se construya o sea alterada.

Descripción del proceso para el diseño de servicios de aplicación:

1. Revisar los servicios existentes
2. Confirmar el contexto
3. Derivar la interfaz inicial
4. Aplicar orientación a servicio.
5. Estandarizar la interfaz de los servicios
6. Agregar propiedades de forma especulativa.

Paso 1. Revisar los servicios existentes.

Aún más con los servicios de aplicación que con otro tipo de servicios, es importante asegurarse que la funcionalidad requerida no exista previamente, de ninguna manera, diseñada o formada. Así que es muy necesaria revisar el inventario existente de servicios de aplicación en búsqueda de cualquier cosa parecida a lo que se quiere diseñar.

Adicionalmente, debido a que estos servicios proveen funcionalidad genérica es conveniente investigar si las características requeridas se pueden comprar o contratar a un tercero, ya que estos los construyen para que sean altamente reusables.

En el caso de la DACE la muestra de servicios que se tomó para el análisis y el diseño es pequeña, por lo que el trabajo se reduce evitar que la funcionalidad planteada en los servicios no sea redundante.

Paso 2. Confirmar el contexto.

Cuando se realiza un análisis orientado a servicio es natural que se haya realizado en base a los requisitos del negocio. Cómo resultado, los servicios de aplicación candidatos que se produjeron para esta fase no suelen tomar en cuenta los contextos establecidos por los servicios de aplicación existentes.

Por consiguiente, es importante que la agrupación de las operaciones candidatas propuestas por los servicios candidatos sea reevaluada y comparada con los diseños de servicios de aplicación existentes. Tras la reevaluación del contexto del servicio, se puede encontrar que una o más operaciones estén contenidas en otro servicio de aplicación.

En el caso de la DACE no existen servicios preexistente a la solución planteada y ya se ha realizado la revisión de las agrupaciones planteadas y se ha establecidos que son válidas.

Paso 3. Derivar una interfaz inicial del servicio.

Este paso se resumen en analizar las operaciones candidatas de los servicios y seguir los siguientes pasos para definir el primer corte de la interfaz del servicio.

1. Utilizar el servicio de aplicación candidato como la entrada primaria, asegurase que la granularidad de las particiones de lógica representadas por las operaciones candidatas sean totalmente genéricas y reusables.
2. Documentar los valores de entrada y salida requeridos para el proceso de cada operación candidata y definir las estructuras del mensaje utilizando conductores de esquemas XSD (Lo que establece esencialmente el constructor *type* de WSDL).
3. Completar la definición abstracta del servicio agregando el área de *portType* (o *interface*) (junto con sus constructores de *operation* hijas) y el los necesarios constructores *message* conteniendo los elementos *part* que reverencian los tipos de esquema apropiados.

Se debe notar que como unidades genéricas de lógica de proceso, los servicios serán usados por diferentes tipos de servicios candidatos. Cada servicio de negocio podrá procesar un tipo diferente de documento de negocios (comprobante de inscripción, de preinscripción, etc.). Por lo tanto, los servicios de

aplicación necesitan ser diseñados de manera tal que los pueden procesar múltiples tipos de documentos. Dependiendo de la naturaleza de la información que se procese, existen algunas opciones de diseño.

Los ejemplos incluyen:

- Crear un conjunto de operaciones que son genéricas pero de documento específico. Por ejemplo en vez de una simple operación de EmitirComprobante, se puede proveer operaciones separadas de EmitirComprobanteDeInscripcion, emitirComprobanteDePreinscripcion.
- Las aplicaciones de servicio pueden estar equipadas para soportar adjuntos SOAP, para permitir una operación genérica para emitir un mensaje SOAP genérico que contenga un documento de negocio específico.

El primer servicio candidato a afrontar es el servicio de asignación (Figura 16):

Figura 16. El primer corte del servicio de asignación.
Fuente: El autor de la investigación.

A continuación, se pasa a definir el constructor *type* de la definición de servicio para formalizar las estructuras del mensaje. En primer lugar, se aborda la petición y la respuesta para los mensajes de la operación.

Los tipos en el esquema XSD requeridos para la operación verificarAsignaciondeBachiller.

```
<xsd:schema
targetNamespace=http://www.ucla.edu.ve/dace/asignacion/sche
ma/">
xsd:element name="verificarAsignaciondeBachillerType">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="cedula"
               type="xsd:string"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
xsd:element
name="verificarAsignaciondeBachillerRetornoType">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="codigodeRetorno"
            type="xsd:integer"/>
         <xsd:element name="cohorte"
            type="xsd:string"/>
         <xsd:element name="modalidad"
            type="xsd:string"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

Seguidamente, la versión inicial de la definición abstracta del servicio de Asignación se completa proveyendo los constructores de `message` y `portType`.

Los constructores *message* y *portType* de la definición abstracta del servicio de asignación.

```
<message
name="verificarAsignaciondeBachillerSolicitudMessage">
   <part name="solicitudParameter"
      element="trn: verificarAsignaciondeBachillerType"/>
</message>
<message name="verificarAsignacionRespuestaMessage">
   <part name="respuestaParameter"
      element="trn:
verificarAsignaciondeBachillerRetornoType"/>
</message>


<portType name="asignacionInterfaz">
   <operation name=" verificarAsignaciondeBachiller">
      <input message=
            "tns:
verificarAsignaciondeBachillerSolicitudMessage "/>
         <output message=
            "tns: verificarAsignacionRespuestaMessage "/>
   </operation>

</portType>
```

Paso 4. Aplicar principios de orientación a servicio

Este paso hace énfasis en los cuatro principios de orientación a servicio citados anteriormente, como aquellos que no son provistos intrínsicamente por la plataforma de los servicios Web (resuablitidad, autonomía, sin estado y capacidad de ser descubiertos).

La reusabilidad fue discutida en el proceso de modelado de servicio, específicamente en el paso 5, donde se buscó hacer el servicio de aplicación tan útil para solicitantes de servicio como sea posible. Sin embargo las operaciones candidatas existente también debe ser revisadas para asegurarse que sean genéricas y reusables.

La autonomía es de interés primordial cuando se diseñan servicios de aplicación. Se debe asegurar que la lógica subyacente responsable de ejecutar las operaciones de servicio no impongan dependencias en el servicio, o el mismo servicio no tenga dependencias. Aquí es donde la información recabada en el paso 2 del proceso de análisis orientado a servicio provee un punto de partida para investigar la naturaleza de la lógica de aplicación que cada operación de servicio

necesita invocar. En paso 6 provee un análisis que cubre este y otros aspectos relacionados con la tecnología.

La característica de "sin estado" puede ser más difícil de lograr con servicios de aplicación. Porque estos se requieren para realizar interfaces con una variedad de plataformas de aplicación, estos están sujetos a ambientes de implementación altamente impredecibles. Tarde o temprano, los servicios de aplicación están ligados a desafíos que imponen los requisitos de desempeño irrazonables e inconsistente (Sistemas obsoletos son conocidos por esto). Por lo tanto la mejor manera de promover un diseño de servicio de aplicación sin estado es llevar a cabo un análisis amplio y veraz, tanto cómo sea posible. Sabiendo de antemano que las demanda de desempeño permitirá indagar sobre alternativas antes de acometer un diseño particular.

Así como servicios centrados en entidad, la capacidad de ser descubiertos puede se una parte importante de la evolución de la capa de servicios de aplicación. Para garantizar que este diseño no se solape con lógica que ya existente provista por otros servicios de aplicación, es útil un mecanismo de descubrimiento. Sin embargo es recomendable suplementar los servicios con metadata tanto como sea posible, para lo cual se recomienda la aplicación de guías de documentación de servicios con metadata.

En la revisión del servicio de asignación para verificar los principios de orientación a servicio enunciados, se evalúa la reutilización potencial de las operaciones en este caso es solo una "verificarAsignacióndeBachiller", en donde se discutió en dividir en dos operaciones de verificar lapso y otra de verificar modalidad, pero se llegó a la conclusión que era regresar al estado previo al modelado del servicio, por lo que se decide mantener la operación.

Lo siguiente es la discusión de la autonomía y el "sin estado". La autonomía no es un problema ya que la lógica necesaria para llevar a cabo la verificación de la asignación está contenida dentro de la lógica de aplicación subyacente del servicio. En otras palabras no hay dependencia de otros

programas. El “sin estado” no es considerado porque el servicio es responsable de su propio proceso.

Finalmente se acepta que pensando en una futura capacidad de descubrimiento, la definición del servicio se debe acondicionar con documentación adicional de metadata.

El constructor portType con documentación de metadata suplementaria.

```
<portType name="asignacionInterfaz">
<documentation>
      Verifica que un Bachiller se encuentre asignado, y de
estar asignado devuelve la cohorte o lapso de asignación y
la modalidad de asignación
 </documentation>
 <operation name=" verificarAsignaciondeBachiller">
      <input message=
            "tns:
verificarAsignaciondeBachillerSolicitudMessage "/>
         <output message=
            "tns: verificarAsignacionRespuestaMessage "/>
   </operation>

</portType>
```

Paso 5. Estandarizar y refinar la interfaz del servicio.

Aún cuando el rol y propósito de los servicios de aplicación difiere de otros tipos de servicios, es importante que estos sena diseñados en la misma manera fundamental. Para cumplir esto hay que asegurarse que la definición WSDL del servicio de aplicación esta basada en los mismos estándares y convenciones usados por otros.

La siguiente es una lista de acciones que se pueden tomar para lograr un diseño estandarizado y estilizado:

- Aplicar cualquier estándar de diseño existente y pertinente a la interfaz del servicio.
- Revisar cualquiera de las características de SOA contemporánea que se hayan escogido para el soporte del servicio y evaluar si es posible construir soporte para dicha característica en el diseño del servicio.
- Opcionalmente incorporar las reglas y mejores prácticas del perfil básico WS-I para cualquier extensión posible.

Con relación al servicio que se está revisando se puede estandarizar el nombre de la operación verificarAsignacióndeBachiller con un solo verbo y un solo nombre en este caso verificarBachiller. El nombre del servicio se decide mantenerlo con el nombre de admisión (Figura 17). Estas características es con vista a otorgarle mayor grado de extensibilidad para que en un futuro se reduzca el esfuerzo de redeserrallo para agregar nuevas características. Hay que tomar en cuenta que estos cambios afectan los nombres de *element* y *message* en su definición.

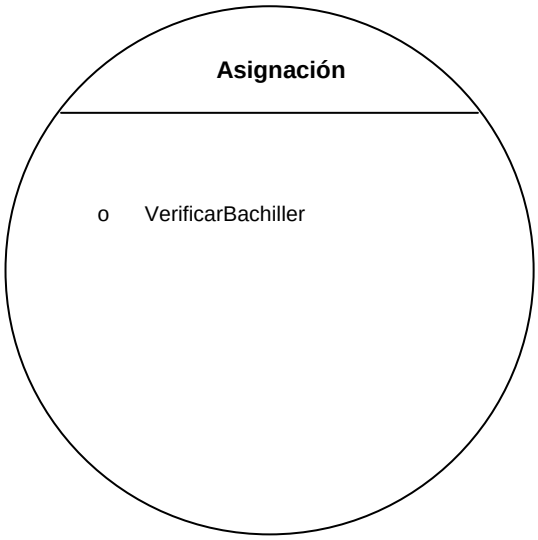


Figura 17. El diseño final del servicio de asignación.
Fuente: El autor de la investigación.

El constructor *Types* revisado.

```
<types>
<xsd:schema
targetNamespace=http://www.ucla.edu.ve/dace/asignacion/sche
ma/">
xsd:element name="verificarAsignacionType">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="cedula"
               type="xsd:string"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
xsd:element name="verificarAsignacionRetornoType">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="codigodeRetorno"
            type="xsd:integer"/>
         <xsd:element name="cohorte"
            type="xsd:string"/>
         <xsd:element name="modalidad"
            type="xsd:string"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
</types>
```

Paso 6. Equipar el servicio candidato con propiedades especulativas.

Si se está interesado en entregar servicios de aplicación altamente reusables, se puede aprovechar esta oportunidad para agregar propiedades al diseño. Estas nuevas propiedades pueden afectar operaciones existentes o pueden resultar en la adición de nuevas operaciones. Para los servicios de aplicación, las extensiones especulativas están basadas en el tipo de procesamiento que cae dentro del contexto del servicio.

Por supuesto, antes de agregar extensiones especulativas al servicio de aplicaciones, se debe repetir el paso 1 para verificar que estas operaciones o parte de estas no existan en otras servicios. Adicionalmente, cuando se agregar nuevas extensiones, es necesario repetir desde el paso 2 hasta el 5 para asegurar que los servicios están correctamente estandarizados y diseñados de acuerdo a la interfaz de servicio creada.

En el caso de la DACE la intención de agregar extensiones especulativas se comienza proponiendo una operación de consultar el total de asignados por

cohorte y modalidad, pero al revisar los demás servicios se llegó a la conclusión que dicha operación se encuentra en el servicio de resumen de admisión.

Paso 7. Identificar restricciones técnicas.

Hasta este punto se ha creado una interfaz de servicio en un poco de vacío. A diferencia de nuestros servicios, los servicios de aplicación necesitan funcionar a bajo nivel y tomar en cuenta consideraciones del mundo real.

Se necesita estudiar y documentar las demandas de procesamiento para cada operación de servicio en una forma más detallada a fondo. Primero para cada operación, se escribe una lista de funciones de procesamiento requeridas para que la operación lleve a cabo su proceso. Entonces para cada elemento de esta lista, se encuentra exactamente el procesamiento de la función que se necesitará ejecutar en el ambiente técnico existente.

Los tipos de detalles que se están buscando específicamente son:

El punto de conexión físico a la función particular (en otras palabras que componentes necesitan ser invocados, que funciones API necesitan ser llamadas, o que adaptadores se necesitan activar)

- Restricciones de seguridad relacionadas a cualquier parte del proceso.
- Tiempo de respuesta de cada función del proceso.
- La disponibilidad para ejecutar sistemas subyacentes para realizar una función.
- Factores de ambiente relacionados al lugar de despliegue del servicio.
- Limitaciones técnicas de lógica de aplicación subyacente (especialmente cuando se está expuesto a sistemas legados).
- Requisitos de administración impuestos por el servicio.
- Potencial requisitos de SLA (Service License Agreement).

Luego de que las características de las funciones de proceso se han obtenido de forma individual, estas necesitan ser vistas colectivamente. Por ejemplo el tiempo de respuesta necesario para agregar al tiempo estimado total de ejecución de la operación. El resultado de este estudio es normalmente una serie de restricciones y limitaciones impuestas por el ambiente técnico en la interfaz del servicio. En algunos casos, las restricciones pueden ser tan severas que una operación puede necesitar ser expandida significativamente.

Nótese que cuando ocurre la transición de una organización hacia un SOA empresarial amplia, existe la tendencia a querer orientar todo a servicio. Sin embargo es importante identificar que requisitos de proceso no pueden ser cumplidos por el conjunto de servicios Web. Puede que no tenga sentido exponer algunas partes de la lógica de aplicación subyacente como servicios Web. De cualquier manera. De cualquier forma es conveniente recordar que aunque la implementación se realiza con Servicios Web, SOA es un modelo arquitectónico neutral y la orientación a servicio es un paradigma de diseño de implementación neutral. Las formas actuales de lógica de aplicación que no están disponibles a través de servicios Web sen embargo se pueden modelar como servicio. Esto es de particular relevancia para los servicios de aplicación, donde exponer la lógica de aplicación a través de servicios Web no siempre es la mejor decisión. Por ejemplo a menudo los componentes de fachada (façade) a menudo se crean para encapsular funcionalidad desde diferentes fuentes para luego exponer un contexto diferente representando un conjunto de funciones reusables. Esto es un registro legitimo, el cual inclusive puede ser expresado en un futuro vía servicio Web.

En esta etapa uno de los fenómenos que siempre hay que tomar en cuenta es la aparición de un cuello de botella en el procesamiento, sin embargo la operación del servicio de asignación, se limita solo a realizar una consulta en una base de datos y no constituye por lo menos hasta ahora un proceso más extenso que los de los demás servicios. Otro observación que hay que hacerse es que la funcionalidad entera está bajo el control del servicio (no existe una aplicación

aparte ni legada). Esta misma sencillez del servicio y sus operaciones garantiza la labor de mantenerlo sin estado, ya que el proceso no ex extenso.

De esta manera se ha llegado a la definición abstracta final del servicio de asignación, incorporando los cambios en los nombres de *element* y todas las revisiones previas.

Definición abstracta final del servicio

```
<definitions name="Asignacion"
   targetNamespace=
      "http://www.ucla.edu.ve/dace/asignacion/wsdl/"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns="http://www.ucla.edu.ve/dace/asignacion/wsdl/"
 xmlns:trn="http://www.ucla.edu.ve/dace/asignacion/schema/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<types>
<xsd:schema
targetNamespace=http://www.ucla.edu.ve/dace/asignacion/sche
ma/">
xsd:element name="verificarAsignacionType">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="cedula"
               type="xsd:string"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
xsd:element name="verificarAsignacionRespuestaType">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="codigodeRetorno"
            type="xsd:integer"/>
         <xsd:element name="cohorte"
            type="xsd:string"/>
         <xsd:element name="modalidad"
            type="xsd:string"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
</types>

<message name="verificarAsignacionSolicitudMessage">
   <part name="solicitudParameter"
      element="trn: verificarAsignacionType"/>
</message>
<message name="verificarAsignacionRespuestaMessage">
   <part name="respuestaParameter"
      element="trn: verificarAsignacionRespuestaType"/>
</message>

<portType name="asignacionInterfaz">
  <documentation>
      Verifica que un Bachiller se encuentre asignado, y
de estar asignado devuelve la cohorte o lapso de asignación
y la modalidad de asignación. Si no está asignados devuelve
un codigo de retorno -1 y los demás datos en blanco
 </documentation>
 <operation name=" verificarAsignacion">
      <input message=
            "tns: verificarAsignacionSolicitudMessage "/>
      <output message=
            "tns: verificarAsignacionRespuestaMessage "/>
 </operation>

</portType>
...
</definitions>
```

Diseño de servicios basados en tarea.

El proceso para diseñar servicios basados en tarea usualmente requiere menos esfuerzo que los diseños de servicios de aplicaciones y los servicios basados en entidad, simplemente porque la reutilización no es una consideración principal. Por lo tanto, aquí solo se trata las operaciones de servicio candidatas identificadas como parte del proceso de modelado.

Descripción del proceso:

- Definir la lógica del flujo de trabajo (workflow).
- Derivar la interfaz inicial.
- Aplicar orientación a servicio.
- Estandarizar la interfaz del servicio.
- Identificar el procesamiento requerido.

Nótese que no existe un paso desafiante para extender el diseño del servicio más allá del que fue definido en la etapa de modelado de servicio. Como se mencionó anteriormente, proveer una interfaz genérica y reusable no es una prioridad para los servicios basados en tarea.

Dicho esto es tiempo de comenzar el diseño del servicio.

El proceso de modelado de servicio identificó la necesidad de un servicio de negocio basado en tarea para controlar el proceso de inscripción de los bachilleres, el resultado es el servicio candidato de inscripción de Bachiller que se muestra en la figura 18 y la composición en la figura 19.

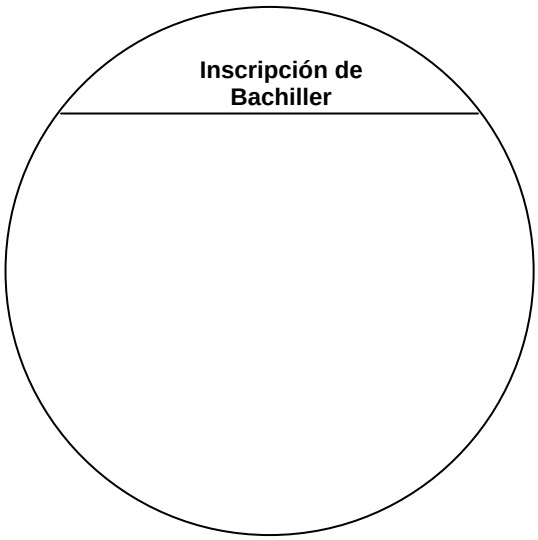


Figura 18. Servicio de inscripción de Bachiller.
Fuente: El autor de la investigación.

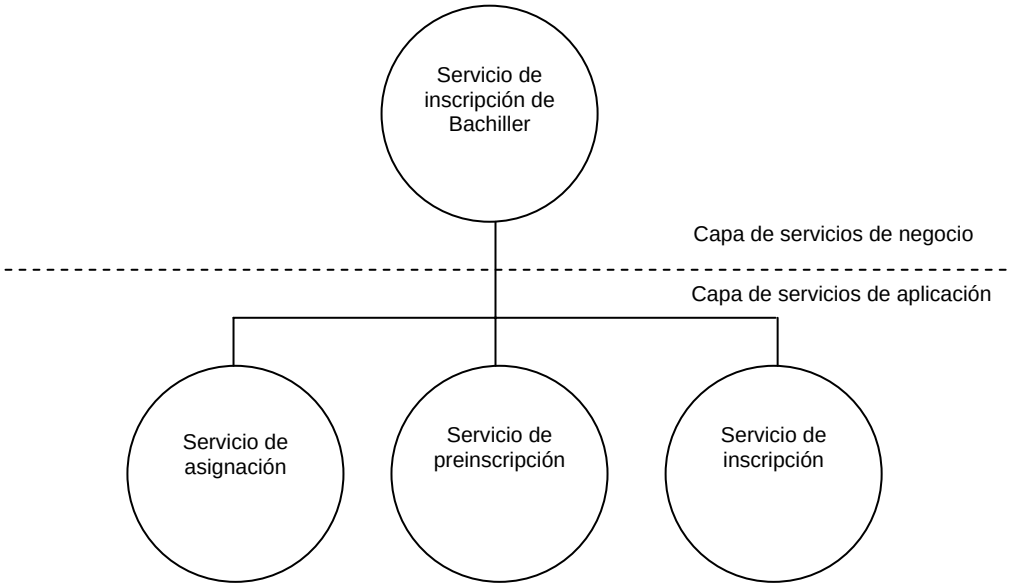


Figura 19. Composición del servicio de inscripción de Bachiller.
Fuente: El autor de la investigación.

Por la ilustración se puede inferir que para inscribir un Bachiller, el servicio necesita componerse de tres servicios diferentes.

Paso 1. Definir la lógica de workflow.

Normalmente los servicios centrados en tarea contendrán lógica de workflow usada para coordinar una composición subyacente de servicio. El primer paso, por consiguiente, es definir la lógica para cada escenario de interacción posible que se pueda imaginar. Si se ha realizado el ejercicio de mapeo, en el paso de identificar composiciones de servicios candidatos en el proceso de modelado, entonces se ya se tiene documentado un detalle preliminar de la composición.

Debido a que se está diseñando el servicio de negocio luego del diseño de los servicio de aplicación, se necesita volver a visitar estos documentos de escenario y convertirlo en modelos concretos de interacción de servicios.

En este paso se pueden utilizar diferentes enfoques de modelado tradicionales, en este caso se utiliza diferentes diagramas de actividad. El propósito de este ejercicio es documentar cada ruta de ejecución posible, incluyendo todas las excepciones. El diagrama resultante será la entrada útil para los subsiguientes casos de prueba.

La lógica de workflow no reside en la interfaz del servicio que se está diseñando en este proceso. La lógica de workflow se está definiendo con el propósito de extraer el intercambio de mensajes con los cuáles se vera envuelto en servicio. Esto provee información para ayudar a definir tipos, operaciones y los formatos de mensaje.

En el siguiente diagrama se ilustra el proceso completo de inscripción de un Bachiller (Figura 20).

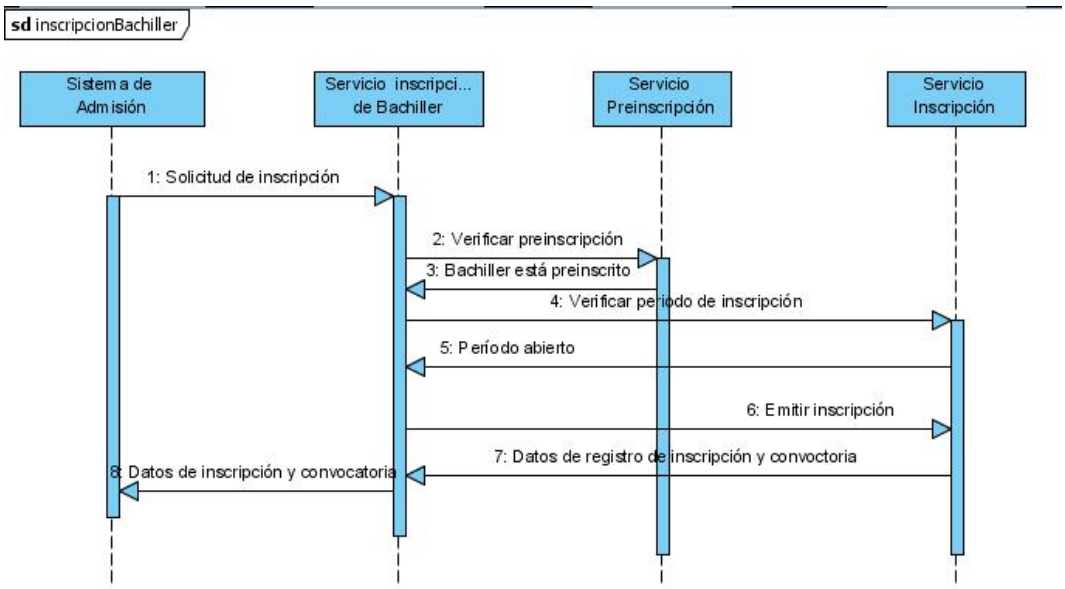


Figura 20. Diagrama de secuencia del proceso de inscripción de Bachiller.
Fuente: El autor de la investigación.

Para los escenarios alternos de workflow también se puede hacer uso de los diagramas de secuencia (figura 21 y 22).

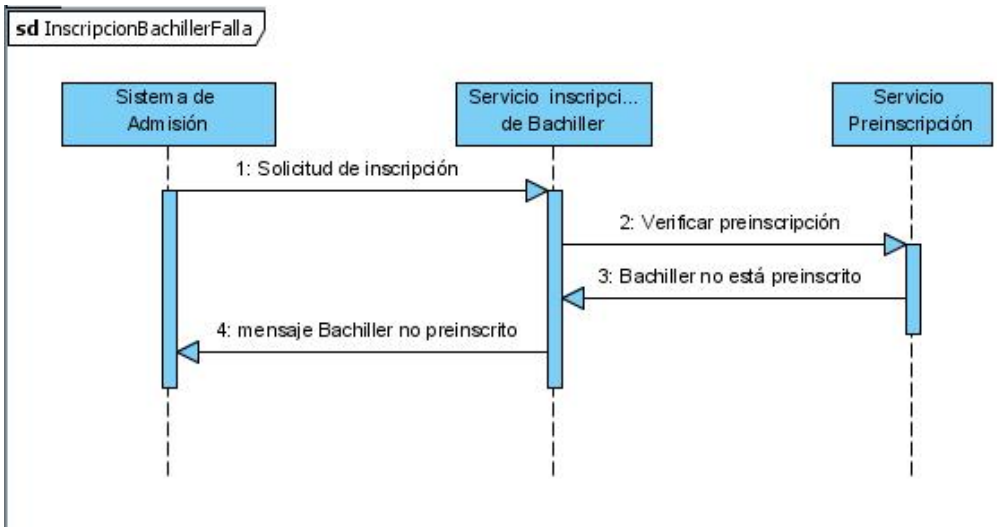


Figura 21. Diagrama de secuencia del proceso de Inscripción, cuando el Bachiller no esta preinscrito.
Fuente: El autor de la investigación.

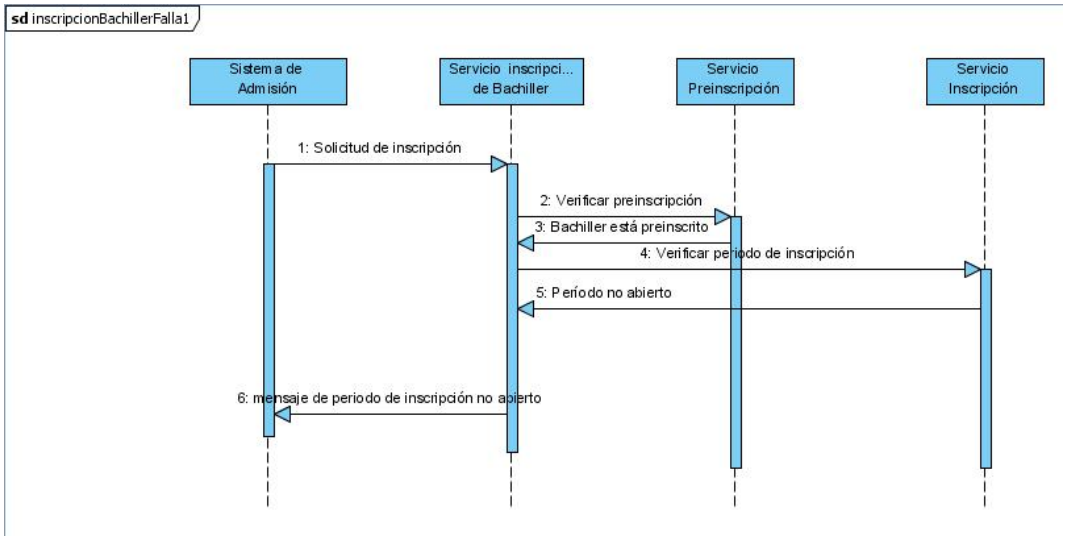


Figura 22. Diagrama de secuencia del proceso de Inscripción, cuando el periodo de inscripción no está abierto.
Fuente: El autor de la investigación.

Paso 2. Derivar la interfaz del servicio.

Para esta actividad se sugieren los siguientes pasos para ensamblar la interfaz de servicio inicial:

1. Utilizar las operaciones de servicio candidatas para derivar un conjunto de operaciones correspondientes.
2. A diferencia de los procesos de diseño anteiores, la fuente de donde se deriva la interfaz del servicio esta vez también incluyen los diagramas de secuencia y la lógica de workflow que se documentaron en el paso 1. Esta información da una buena idea de que operaciones adicionales puede requerir el servicio.
3. Documentar los valores requeridos de entrada y salida para el procesamiento de cada operación y llenar la sección de *types* con los tipos del esquema XSD para procesar las operaciones.
4. Construir la definición WSDL para crear el área de *portType* (o *interface*), insertando los constructores de operación identificados. Agregar los constructores necesarios de *message* conteniendo los

elementos *part* que hacen referencia a los tipos de esquema apropiados.

En la DACE se apunta hacia el diagrama de secuencia creado para derivar el conjunto de acciones que el servicio necesita para operar (figura 23), para realizar las siguientes observaciones:

- El servicio de inscripción recibe la solicitud de inscripción enviada en este caso por el sistema de admisión, aunque puede ser por cualquier demandante del servicio, para arrancar el proceso de inscripción.
- El servicio de inscripción de Bachiller emite una solicitud de verificación de preinscripción al servicio de preinscripción.
- El servicio de inscripción del bachiller una vez que esta verificada la preinscripción procede a enviar solicitud al servicio de inscripción para verificar que el periodo de inscripción para la cohorte del bachiller se encuentre abierto.
- Finalmente luego que está verificada la preinscripción y que el periodo de inscripción se encuentra abierto, el servicio de inscripción de Bachiller enviar solicitud de emitir inscripción al servicio de inscripción y este último retorna los datos de la inscripción y la convocatoria para el Bachiller en cuestión.

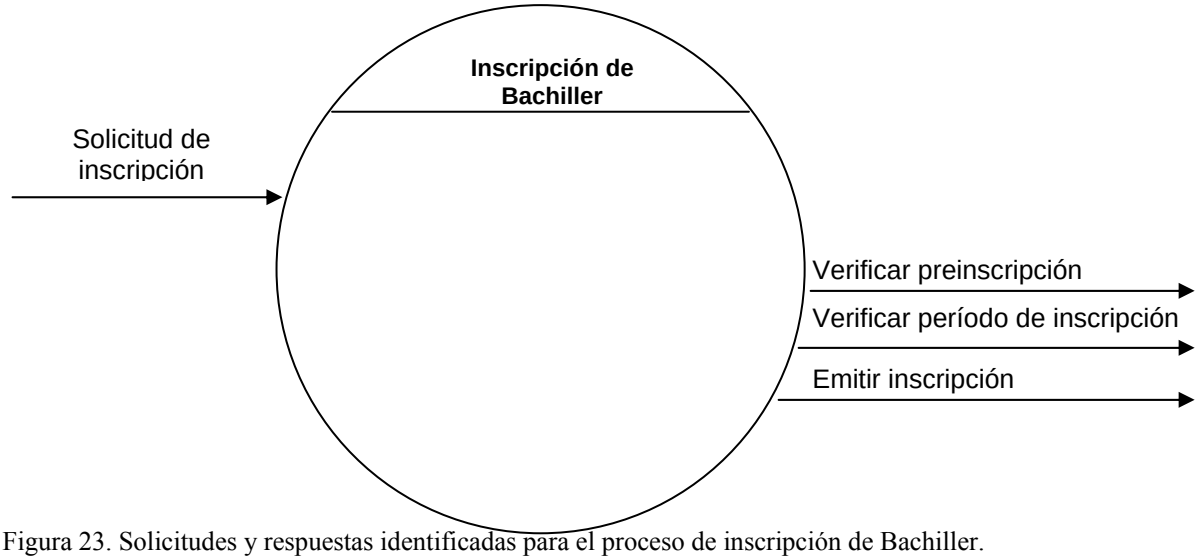


Figura 23. Solicitudes y respuestas identificadas para el proceso de inscripción de Bachiller.
Fuente: El autor de la investigación.

De estas acciones, la ultimas tres requieren que el servicio actué como un solicitante para hincar un intercambio de mensajes con otros servicios. Por lo tanto estas acciones se implementaran como parte de la lógica de negocios subyacente del servicio.

La acción “solicitar inscripción de Bachiller”, aunque es iniciada por el sistema de admisión, significa que el servicio de inscripción de Bachiller recibe la solicitud mientras que actúa como proveedor de servicio. Esta acción por lo tanto necesita ser expresada en la interfaz del servicio.

Por lo tanto se necesita agregar una operación a la interfaz del servicio, en este caso con el nombre de “inscribir Bachiller” (Figura 24).

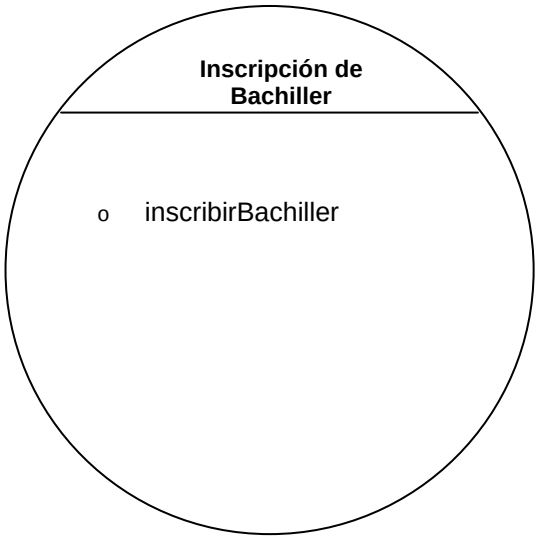


Figura 24. El servicio de inscripción de servicio con la operación agregada.
Fuente: El autor de la investigación.

Al seguir adelante con el cambio de nombre en la operación, la DACE comienza a definir los requisitos para definir el intercambio de datos para esta operación. La interfaz del servicio requiere de un identificador del Bachiller en este caso la cédula es lo convenido.

Para cumplir con el requisito de intercambio de información se crean los constructores types con el respectivo complexType y así coincidir con los parámetros requeridos. En el caso del complexType se agrega previendo la

inclusión de parámetros adicionales a la cédula, porque con un *type* pudiera ser suficiente.

El constructor complexType diseñado para recibir el parámetro desde el sistema de admisión y devolver los datos solicitados.

```
<types>
   <xsd:schema targetNamespace=

"http://www.ucla.edu.ve/dace/inscripcionBachiller/schema/">
      <xsd:element name="inscribirBachillerType">
         <xsd:complexType>
            <xsd:sequence>
               <xsd:element name="cedula"
                  type="xsd:string"/>
            </xsd:sequence>
         </xsd:complexType>
      </xsd:element>
      <xsd:element name="inscribirBachillerRespuestaType">
         <xsd:complexType>
            <xsd:sequence>
               <xsd:element name="retorno"
                  type="xsd:integer"/>
            </xsd:sequence>
            <xsd:sequence>
               <xsd:element name="apellidos"
                  type="xsd:string"/>
            </xsd:sequence>
               <xsd:element name="nombres"
                  type="xsd:string"/>
            </xsd:sequence>
            <xsd:element name="fechaInscripcion"
                  type="xsd:string"/>
            </xsd:sequence>
           <xsd:element name="programa"
                  type="xsd:string"/>
            </xsd:sequence>
            <xsd:element name="fechaHoraConvocatoria"
                  type="xsd:string"/>
            </xsd:sequence>
         </xsd:complexType>
      </xsd:element>
   </xsd:schema>
</types>
```

Lo que sigue es definir la operación y sus mensajes asociados dentro de los constructores *portType* y *message*.

La parte restante de la definición abstracta del servicio de inscripción de Bachiller

```
<message name="solicitudMessage">
   <part name="solicitudParameter"
      element="ins:incribirBachillerType"/>
</message>
<message name="respuestaMessage">
   <part name="respuestaParameter"
      element="ins:incribirBachillerRespuestaType"/>
</message>

<portType name="inscribirBachillerInterface">
   <operation name="inscribirBachiller">
      <input message="ins: solicitudMessage"/>
      <output message="ins: respuestaMessage"/>

   </operation>
</portType>
```

Paso 3. Aplicar principios de orientación a servicio.

Antes de avanzar en el diseño del servicio, es conveniente echar otro vistazo a los cuatros principios de orientación que se han estudiado antes, los cuales no se proveen automáticamente con los servicios Web (reusabilidad, autonomía, sin estado y con capacidad de ser descubiertos).

Las oportunidades de reutilización para los servicios centrados en tarea son mucho más raras que para los servicios basados en entidad y los de aplicación. Esto es porque los servicios basados en tarea representan una porción de la lógica de workflow específica para un proceso de negocio. Sin embargo, la reutilización se puede lograr.

Debido que estos servicios casi siempre actúan como servicios controladores padres en las composiciones, la autonomía generalmente depende de la autonomía de los servicios hijos.

Los servicios de basados en tarea contienen lógica de workflow que puede imponer dependencias en las composiciones de servicio. Esto puede sugerir la necesidad de manejo de estado. Sin embargo el uso de mensaje SOAP estilo documento permite a los servicios delegar la persistencia de de alguno o toda la información referente al estado al mensaje mismo.

Siempre es útil que un servicio sea descubrible, pero la necesidad para un servicio basado en tarea no esta presente como para otros servicios más genéricos. En el servicio de inscripción de Bachiller solo se documentará con metadata adicional para aumentar en cierto modo la capacidad de descubrimiento (discoverability). El diseño del servicio se complementa con metadata de documentación adicional para soportar la capacidad de descubrimiento.

El constructor *portType* con el elemento de documentación adicional.

```
<portType name="inscribirBachillerInterface">
   <documentation>
        Inicia el proceso de inscripción de un bachiller y
devuelve los datos relacionados con la inscripción y la
convocatoria de la inscripción
   </documentation>
   <operation name="inscribirBachiller">
      <input message="ins: solicitudMessage"/>
      <output message="ins: respuestaMessage"/>

   </operation>
</portType>
```

Paso 4. Estandarizar y refinar la interfaz del servicio.

Aunque los servicios de negocio centrados en tarea tenderán a tener más nombres de operación creativos, no obstante es necesario aplicar convenciones existentes. A continuación se recomiendan algunas acciones para lograr un diseño estandarizado y refinado:

- Incorporar estándares de diseño y directivas existentes.
- Asegurarse de escoger que las características de SOA contemporánea son soportadas por el diseño de la interfaz del servicio.
- Tomar en cuenta los estándares y mejores prácticas del perfil básico WS-I.

Con relación a los estándares de diseño relacionados con la granularidad de la operación, se puede requerir cierta tolerancia para acomodar el procesamiento de la lógica de workflow secuencial incorporada del servicio. Los servicios de negocio basados en tarea también se pueden beneficiar de la reutilización de módulos WSDL existentes, en particular de esquemas de definición XSD.

En el caso de la DACE el nombre de la operación del servicio es “inscribirBachiller” y siguiendo una convención de nombres interno, se decide acortarla y dejarla en “inscribir” ya que el nombre del servicio contiene la palabra “Bachiller” y no es necesario repetirla en los nombres de las operaciones (Figura 25).

Previendo que el documento de identidad del Bachiller no sea sólo la cédula (puede ser el pasaporte), se decide cambiar el parámetro “cedula” por el de “id”. De esta manera se tocan los componentes *types, message y porttype,* donde se denota la definición abstracta definitiva del servicio.

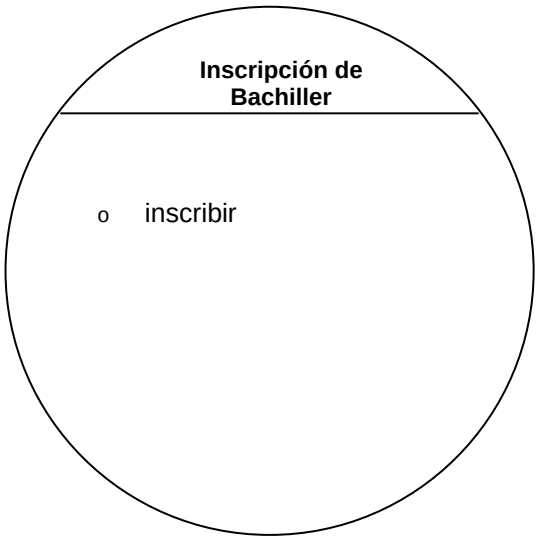


Figura 25. Servicio de inscripción de Bachiller con el nombre de operación revisada.
Fuente: El autor de la investigación.

Paso 5. Identificar el procesamiento requerido.

Para llevar a cabo la labor de compartir la lógica de proceso de la solución, los servicios basados en tarea pueden componerse de servicios de aplicación y de servicios centrados en tarea y servicios adicionales de negocios basados en tarea. Por lo tanto la implementación de una interfaz de servicio basado en tarea requiere de cualquier capa de servicios subyacente que se necesite en lugar de soportar los requisitos de procesamiento de sus operaciones.

Debido a que este es el último paso de los procesos de diseño, se necesita identificar todos los servicios de aplicación requeridos. Estos pueden ser servicios que ya existan y/o servicios que se han diseñado durante el anterior proceso de diseño de servicios de aplicación. El diseño de la lógica de proceso dentro de los servicios de negocio basados en tarea también puede revelar la necesidad de servicios de aplicación adicionales que aún no se han considerado.

En el caso de la DACE, no se identifican servicios adicionales para llevar a cabo la labor expuesta por el servicio de inscripción de estudiantes. A continuación se expone la versión definitiva de la definición del servicio de inscripción de Bachilleres.

Definición final abstracta del servicio

```
<definitions name="inscripcionBachiller">
targetNamespace=

"http://www.ucla.edu.ve/dace/inscripcionBachiller/wsdl/"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:ins="http://
www.ucla.edu.ve/dace/inscripcionBachiller/schema/"
   xmlns:inss=
      "http://www.ucla.edu.ve/dace/ inscripcionBachiller
/schema/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"

xmlns:tns="http://www.ucla.edu.ve/dace/xxxxxxxtransform/wsd
l/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<types>
   <xsd:schema targetNamespace=

"http://www.ucla.edu.ve/dace/inscripcionBachiller/schema/">
      <xsd:element name="inscribirBachillerType">
         <xsd:complexType>
            <xsd:sequence>
               <xsd:element name="idBachiller"
                  type="xsd:string"/>
            </xsd:sequence>
         </xsd:complexType>
      </xsd:element>
      <xsd:element name="inscribirBachillerRespuestaType">
         <xsd:complexType>
            <xsd:sequence>
               <xsd:element name="retorno"
                  type="xsd:integer"/>
            </xsd:sequence>
            <xsd:sequence>
               <xsd:element name="apellidos"
                  type="xsd:string"/>
            </xsd:sequence>
               <xsd:element name="nombres"
                  type="xsd:string"/>
            </xsd:sequence>
            <xsd:element name="fechaInscripcion"
                  type="xsd:string"/>
            </xsd:sequence>
           <xsd:element name="programa"
                  type="xsd:string"/>
            </xsd:sequence>
            <xsd:element name="fechaHoraConvocatoria"
                  type="xsd:string"/>
            </xsd:sequence>
         </xsd:complexType>
      </xsd:element>
   </xsd:schema>
</types>
<message name="solicitudMessage">
   <part name="solicitudParameter"
      element="ins:incribirBachillerType"/>
</message>
<message name="respuestaMessage">
   <part name="respuestaParameter"
      element="ins:incribirRespuestaType"/>
</message>

<portType name="inscribirBachillerInterface">
   <documentation>
```

```
        Inicia el proceso de inscripción de un bachiller y
devuelve los datos relacionados con la inscripción y la
convocatoria de la inscripción, a partir de un id de
Bachiller (Cédula o Pasaporte)
   </documentation>
   <operation name="inscribir">
      <input message="ins: solicitudMessage"/>
      <output message="ins: respuestaMessage"/>

   </operation>
</portType>
…
</definitions>
```

En resumen se puede afirmar que el interés principal para los servicios basados en tarea es una representación acertada de los procesos de la lógica de negocio que se necesitan ejecutar y que la reutilización y las extensiones de diseño especulativo son de interés secundario.

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

Con el desarrollo de la presente investigación se pudieron establecer las siguientes conclusiones:

- Con la implementación de los instrumentos de recolección de datos se pudo determinar la ausencia de documentación en cuanto al funcionamiento del negocio, como por ejemplo documentos que ilustren la forma de interacción de los conceptos del dominio y de los procesos que se llevan a cabo para el funcionamiento de la organización; así como también que la DACE no está en capacidad de ofrecer servicios de información que puedan ser utilizados como componentes en desarrollos oportunistas.
- Al afrontar el desarrollo de sistemas con el paradigma de orientación a servicios, la separación de intereses que se realizó condujo a una arquitectura inicial cuyos componentes cumplen con los criterios de reusabilidad, sin estado, que puedan participar en composiciones y que tengan la capacidad de ser descubiertos.
- Se confirmó la premisa de la orientación a servicio, que establece que la fase más importante en el ciclo de vida de SOA es el análisis, ya que de los resultados obtenidos en esta, permitió definir el posterior diseño de los servicios y de esta manera cumplir con los requisitos de automatización especificados en la fase diagnóstica.
- La fase de análisis orientado a servicio permitió obtener una organización de la lógica del negocio y de las aplicaciones en componentes llamados servicios, los cuales a su vez quedan organizados en capas que reflejan la naturaleza de la lógica que encapsulan.

- El principal objetivo de la etapa del diseño orientado a servicio es proveer una definición abstracta de la interfaz de los servicios, donde se definan los datos de entrada, las operaciones que realiza y los datos que devuelve, con la documentación específica de su comportamiento.
- Al aplicar el paradigma de orientación a servicio en la DACE, se dota a la organización de capacidad para ofrecer información estandarizada, con acuerdos preestablecidos para el intercambio y se sientan las bases para que no ocurran "redesarrollos" en otros entes, que al final repiten lógica que ya se encuentra encapsulada en los servicios que se construyen.

Recomendaciones.

- Utilizar el paradigma de orientación a servicio para los desarrollos que se emprendan dentro de la DACE como a través de toda la UCLA.
- Mantener un inventario automatizado de todos los servicios producidos para no incurrir en desarrollar lo que ya está desarrollado y evitar la redundancia y el retrabajo en todo el ámbito de la UCLA.
- Establecer normas para que los desarrollos que se inicien cumplan con estándares metodológicos y tecnológicos.
- Utilizar las bases de SOA sentadas por el presente estudio para expandir su campo de acción y apuntar hacia un SOE (Software Oriented Environment).

BIBLIOGRAFÍA

Ambroszkiewicz S. (2003). Agile modelling and design of component- and service-oriented architecture. First European Workshop on Object Orientation and Web Services.http://domino.watson.ibm.com/library/cyberdig.nsf/papers/2F51152815EA24DD85256D73004E00A5/$File/ra220.pdf. (Consulta Junio 2007).

Blanco S. (2006). Utilización de UML como un ADL para documentación de Arquitecturas. Regional Architect Forum 2006. http://raf06-pta-del-este.spaces.live.com/default.aspx. (Consulta Agosto 2007).

Chen R. (2007). Defining Business Services: SOA from a Corporate Perspective. The SOA Magazine. http://www.soamag.com/default.asp. (Consulta Noviembre 2007).

Booch G. (1999). Is UML an architectural Description Language?. OOPSLA. http://www.sigplan.org/oopsla/oopsla99/2_ap/tech/2d1a_uml.html (Consulta Julio 2007).

Cámara Venezolana de Comercio Electrónico (2007). http://www.cavecom-e.org.ve. (Consulta Enero 2007).

Computer Industry Almanac (2007). http://www.c-i-a.com. (Consulta Enero 2007).

D'andrea V. y Aiello M (2003). Agile modelling and design of component- and service-oriented architecture. First European Workshop on Object Orientation and Web Services. http://domino.watson.ibm.com/library/cyberdig.nsf/papers/2F51152815EA24DD85256D73004E00A5/$File/ra220.pdf. (Consulta Junio 2007).

Earl T (2005). Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall PTR.

Garlan D. (1999). Is UML an architectural Description Language?. OOPSLA. http://www.sigplan.org/oopsla/oopsla99/2_ap/tech/2d1a_uml.html (Consulta Julio 2007).

Instructivo presidencial número 14 (2001). Chile: Sistema nacional de coordinación de información. http://www.snit.gob.cl/Presentacion_Instructivo_Presidencial_14.aspx. (Consulta Mayo 2007).

Manual de Trabajos de Grado de Especialización y Maestría y Tesis Doctorales. Universidad Pedagógica Libertador. Vicerrectorado de Investigación y Postgrado. Caracas 2005.

Martin S y Mike D (2003). Agile modelling and design of component- and service-oriented architecture. First European Workshop on Object Orientation and Web Services.http://domino.watson.ibm.com/library/cyberdig.nsf/papers/2F51152815EA24DD85256D73004E00A5/$File/ra220.pdf. (Consulta Junio 2007).

Papazoglou M, et al. (2005). Service Oriented Architectures:Approaches, Technologies and Research Issues. http://infolab.uvt.nl/pub/papazogloump-2005-81.pdf. (Consulta Mayo, 2007).

Pelechano V, et al. (2002). Desarrollo de Aplicaciones WEB basadas en Servicios WEB XML. Un Caso Práctico. http://oomethod.dsic.upv.es/anonimo/..%5Cfiles%5CBookChapter%5Ceecw02.pdf. (Consulta Mayo 2007).

Pelechano V, et al. (2005). Servicios Web. Estándares, Extensiones y Perspectivas de Futuro. http://petra.euitio.uniovi.es/~i1893878/WebServices/ServiciosWeb.pdf. (Consulta Mayo 2007).

Perepletchikov M, et al. (2005). The Impact of Software Development Strategies on Project and Structural Software Attributes in SOA. http://goanna.cs.rmit.edu.au/~caspar/downloads/INTEROP2005paper.pdf. (Consulta Mayo, 2007).

Reynoso C. y Kicillof R. (2004). Leguajes de Descripción Arquitectónica. http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/lenguaje.mspx#top. (Consulta Junio 2007).

Sommerville I. (2005). Ingeniería del Software. Pearson Educación. Madrid España.

Skonnard y Gudgin. (2002). Essential XML Quick Reference, Addison-Wesley.

Stojanovic Z, et al. (2003). Agile modelling and design of component- and service-oriented architecture. First European Workshop on Object Orientation and Web Services.http://domino.watson.ibm.com/library/cyberdig.nsf/papers/2F51152815EA24DD85256D73004E00A5/$File/ra220.pdf. (Consulta Junio 2007).

Tsenov M (2006), et al. 2006. Web Services Example with PHP/SOAP. http://ecet.ecs.ru.acad.bg/cst06/Docs/cp/SIII/IIIA.10.pdf. (Consulta Mayo 2007).

UCLA (2005). Informe técnico sobre la situación de los sistemas de información de los registros académicos de la UCLA. (Consulta Mayo 2007)

UPEL (2005). Manual de Trabajos de Grado de Especialización y Maestría y Tesis Doctorales.

ANEXOS

ANEXO 1

Cuestionario No.1. Entrevista estructurada

UNIVERSIDAD CENTROCCIDENTAL "LISANDRO ALVARADO"
DECANATO DE CIENCIAS Y TECNOLOGIA
MAESTRIA EN CIENCIAS DE LA COMPUTACIÓN

PROPUESTA DE UNA ARQUITECTURA ORIENTADA A SERVICIO PARA LA DIRECCION DE ADMISIÓN Y CONTROL DE ESTUDIOS DE LA UCLA

ENTREVISTA ESTRUCTURADA

Apellidos y Nombres	
Cargo	
Departamento	

1. ¿Existe un documento donde se plasme el vocabulario utilizado en funcionamiento del negocio, donde se refleje la descripción de los conceptos utilizados en el negocio?
2. ¿La información que se ofrece a los demás entes de la institución está organizada o clasificada en servicios?
3. ¿Los datos que son proporcionados por la DACE son consistentes, reflejan los eventos ocurridos en la realidad?
4. ¿Los mecanismos establecidos para acceder y consumir la información de la DACE, permiten un acceso sencillo a dichos datos?

5. ¿Los mecanismos para la publicación y habilitación de la información, (para que esta se encuentre disponible a los demás entes) son eficaces? ¿Son automatizados, no requiere que una persona realice actividades para habilitación posterior al requerimiento. Ej. No hay que enviar un email cada vez que se solicita información?
6. ¿Existen acuerdos preestablecidos para el intercambio de información entre los entes involucrados en el transito de la información (caso DACE y Registros Académicos)?
7. Si existen componentes de software, ¿Se encuentran modelados de acuerdo a conceptos del dominio (entidades o procesos)?.
8. Si existen componentes de software, ¿Cumplen con características esbozadas en la teoría de lo que es un servicio (bajo acoplamiento, con capacidad de composición, con capacidad de ser descubierto, etc.)?
9. Si existen componentes de software, ¿Están basados en estándares, Ej. xml y cuales son?
10. Si existen componentes de software, ¿Cumplen con exigencia de la industria (enunciados teóricos que ya se encuentran implementados con la tecnología actual), Ej. Ejecutándose en un servidor de aplicaciones, organizados en capas, etc. ?

ANEXO 2

Cuestionario No.1. Entrevista no estructurada

UNIVERSIDAD CENTROCCIDENTAL "LISANDRO ALVARADO"
DECANATO DE CIENCIAS Y TECNOLOGIA
MAESTRIA EN CIENCIAS DE LA COMPUTACIÓN

PROPUESTA DE UNA ARQUITECTURA ORIENTADA A SERVICIO PARA LA DIRECCION DE ADMISIÓN Y CONTROL DE ESTUDIOS DE LA UCLA

ENTREVISTA NO ESTRUCTURADA

Apellidos y Nombres	
Cargo	
Departamento	

1. Dentro de los procesos que se llevan a cabo en la DACE, en cuanto a la admisión e inscripción de Bachilleres ¿Cuáles son los más significativos para el funcionamiento de la Dirección?
2. Puede describir brevemente en que consisten las diferentes fases que se suceden para que un bachiller abra expediente o resulte formalmente inscrito ante la DACE.
3. ¿Por lo general qué otras dependencias de la UCLA utilizan la información administrada y tratada por los procesos que lleva a cabo la DACE?
4. ¿Considera UD. que un cambio en las reglas del negocio pueda ser enfrentado con un bajo impacto en las aplicaciones o por el contrario

haya que emprender modificaciones estructurales en la infraestructura de los sistemas de TI?

5. ¿Considera UD. qué la información que ofrece la DACE y la lógica del negocio que utiliza, pueda estar disponible de manera oportuna y automática a través de la infraestructura de red de la UCLA, tomando en cuentas restricciones propias del negocio, como por ejemplo previa autorización de las autoridades de la UCLA?
6. ¿Considera UD. qué se pueden introducir cambios en los sistemas de TI para que mejore la eficiencia de los procesos y el desempeño de la DACE?

ANEXO 3

Carta de Solicitud de Validación a los Expertos

UNIVERSIDAD CENTROCCIDENTAL "LISANDRO ALVARADO"
DECANATO DE CIENCIAS Y TECNOLOGIA
MAESTRIA EN CIENCIAS DE LA COMPUTACIÓN

Barquisimeto, Febrero de 2008

Ciudadano (a):_____

Reciba un cordial saludo en mi nombre. Sirva la presente para solicitar ante Ud. su colaboración en calidad de experto para determinar la validez de contenido de los cuestionarios con los cuales se pretende recabar información necesaria para el Trabajo de Grado titulado "Propuesta de una arquitectura orientada a servicio para la Dirección de Admisión y Control de Estudios de la UCLA", el cual será aplicado al personal que labora en la Dirección de Admisión y Control de Estudios.

Para tal propósito se anexa a la presente: 1) Matriz de Operacionalización de las Variables; 2) Formatos de Validación con sus respectivas instrucciones; 3) Cuestionarios que se aplicarán.

Su opinión al respecto representa un gran aval para esta investigación, dada la excelente labor docente, de investigación y extensión que Ud. ha realizado en pro de la formación profesional en la Universidad.

Sin otro particular al cual hacer referencia y agradeciendo de antemano su colaboración, quedo de Ud.

Atentamente,

_____
Prof. Aly González.
C.I.: 10.564.328

ANEXO 4

Formato de Validación de los Instrumentos

UNIVERSIDAD CENTROCCIDENTAL "LISANDRO ALVARADO"
DECANATO DE CIENCIAS Y TECNOLOGIA
MAESTRIA EN CIENCIAS DE LA COMPUTACIÓN

PROPUESTA DE UNA ARQUITECTURA ORIENTADA A SERVICIO PARA LA DIRECCION DE ADMISIÓN Y CONTROL DE ESTUDIOS DE LA UCLA

Formato de Validación de los Instrumentos

Ciudadano (a): _____

Para efectos de la evaluación correspondiente a los ítems planteados se determinará la validez de cada instrumento en los siguientes términos:

Se tomarán en cuenta los siguientes aspectos: a) Pertinencia: Es la correspondencia del ítem con el aspecto; b) Claridad: se refiere a la redacción precisa y sencilla del ítems; y c) Congruencia: entendida como la lógica interna del ítem.

Se le agradece seleccionar una de las 2 posibles opciones (Si/No) para cada ítems con el objetivo de señalar el grado de pertinencia, claridad y congruencia de los ítems.

Cuestionario N° 1

	Pertinencia		Claridad		Congruencia		
Ítem	**SI**	**NO**	**SI**	**NO**	**SI**	**NO**	**OBSERVACIONES**
1	☐	☐	☐	☐	☐	☐	
2	☐	☐	☐	☐	☐	☐	
3	☐	☐	☐	☐	☐	☐	
4	☐	☐	☐	☐	☐	☐	
5	☐	☐	☐	☐	☐	☐	
6	☐	☐	☐	☐	☐	☐	
7	☐	☐	☐	☐	☐	☐	
8	☐	☐	☐	☐	☐	☐	
9	☐	☐	☐	☐	☐	☐	
10	☐	☐	☐	☐	☐	☐	

Cuestionario N° 2

	Pertinencia		Claridad		Congruencia		
Ítem	**SI**	**NO**	**SI**	**NO**	**SI**	**NO**	**OBSERVACIONES**
1	☐	☐	☐	☐	☐	☐	
2	☐	☐	☐	☐	☐	☐	
3	☐	☐	☐	☐	☐	☐	
4	☐	☐	☐	☐	☐	☐	
5	☐	☐	☐	☐	☐	☐	
6	☐	☐	☐	☐	☐	☐	