UNIVERSIDAD DE LAS CIENCIAS INFORMATICAS

FACULTAD 5

“Propuesta de Estrategia para el Desarrollo de un Modelo de Datos Canónicos”

Trabajo de diploma para optar por el título de ingeniero en ciencias informáticas

Autor: Yohana García Gonzalo

Tutores: Ing. Luis Enrique Hernández Vega

Ing. Jessie Castell González

La Habana, Julio de 2012

Declaración de autoría

Declaro ser autora de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los 28 días del mes junio del año 2012.

_____

Yohana García Gonzalo

Autora

_____

Ing. Luis Enrique Hernández Vega

Tutor

_____

Ing. Jessie Castell González

Cotutora

Agradecimientos

A mis padres, por el apoyo y respeto a mis decisiones.

A Jorgito, por el sacrificio y la paciencia en momentos de stress.

A mis tutores, por la orientación, la ayuda y el tiempo dedicado.

A mi familia, por compartir la preocupación.

A mis amigos, los que siempre veo y los que ya no.

A Mary y a Carmen Rosa, por los consejos.

A mis compañeros de aula, por la ayuda mutua.

A Jorge Infante, por la exigencia.

Y a todos los que cooperaron con la realización de esta tesis.

A mis padres

Por el sueño cumplido y el sacrificio recompensado.

Resumen

Las organizaciones de hoy día gestionan sus procesos de negocio en aplicaciones informáticas que fueron diseñadas con esta finalidad. El enfoque de desarrollo de sistemas que se ha venido aplicando hasta el momento introduce las aplicaciones como grandes bloques altamente acoplados, en los que las dependencias en la estructura interna de la aplicación implican que sea un sistema inmutable. Esto último afecta la interoperabilidad entre sistemas y la flexibilidad de las aplicaciones ante los cambios que ocurren en los procesos de negocio en la organización. A fin de resolver los problemas de interoperabilidad se han desarrollado diferentes soluciones de integración que son aplicables en cualquier organización teniendo en cuenta las peculiaridades del problema de integración presentado y de la empresa en sí, por lo que no se puede decir que existe una fórmula para lograr la integración empresarial. El presente trabajo diploma enfoca la perspectiva de integración desde el principio de la transformación de los mensajes que intercambian las aplicaciones durante la comunicación y en lograr que estas se entiendan satisfactoriamente. Para ello se propone una estrategia para el desarrollo de un Modelo de Datos Canónicos que incluye premisas, roles y actividades. Finalmente se aplicó la estrategia en un proyecto de interoperabilidad de la Universidad de las Ciencias Informáticas aun en desarrollo, a cargo del Centro de Consultoría y Desarrollo de Arquitecturas Empresariales y se comprobó la calidad de la estrategia propuesta mediante la evaluación por Método Experto de aspectos cualitativos que responden a la calidad de la misma.

Palabras Claves

Estrategia, Interoperabilidad, Integración, Modelo de Datos Canónicos.

Índice

Índice de figuras y tablas

Introducción

La gestión de la información en las empresas de hoy se convierte cada vez más en un problema complejo de resolver, pues estas trabajan con gran cantidad de datos ya sea de sus procesos de negocio, áreas departamentales u otra información significativa para la organización. Cuando esto sucede, se desarrollan diferentes sistemas informáticos con la intención de lograr una mejor gestión de la información. Al coexistir en un mismo entorno tantas aplicaciones tienden a aparecer problemas de interoperabilidad entre los sistemas. Para darle solución a este tipo de problemática mundial se han desarrollado iniciativas de integración que intentan minimizar los efectos negativos de la falta de comunicación que presentan los sistemas.

Una solución de integración debe ser capaz de permitir la transmisión información entre varios sistemas independientemente de las plataformas de los sistemas, los formatos de los datos o los lenguajes de programación. Además debe ser persistente a través del tiempo, aunque las aplicaciones que conecta cambien, la integración no debe sufrir el mismo efecto. Otra característica que debe cumplir es el bajo acoplamiento que minimizaría las dependencias de un sistema a otro [1].

Existen estilos de integración como la mensajería, base de datos compartida, transferencia de archivos e invocación a proceso remoto que ofrecen vías para establecer la integración, además de patrones para facilitar el desarrollo por medio de la reutilización y las buenas prácticas. Estos patrones no están asociados a ninguna tecnología en específico, solo intentan estandarizar técnicas de diseño a fin de ayudar a superar obstáculos que comúnmente se encuentran durante el desarrollo de soluciones de integración [1, 2].

La mensajería es un estilo de integración que introduce los mensajes como bloque principal de comunicación. Las aplicaciones interpretan los mensajes que reciben desde un mismo canal de comunicación, con el que deben estar de acuerdo, así como con el formato de los mensajes intercambiados. Este estilo tiene como principales beneficios la introducción de la comunicación asíncrona, el bajo acoplamiento y la independencia de las aplicaciones implicadas en la integración. Los patrones de integración empresarial que propone la mensajería están agrupados en dependencia de las funciones que realizan. Uno de estos grupos es el de transformación de mensajes, en el que se incluyen

los patrones que se encargan del entendimiento entre los mensajes, transformándolos a nuevos formatos que sean perceptibles por las demás aplicaciones [1].

Uno de los patrones que se desarrollan en este grupo es el Modelo de Datos Canónicos o MDC (en inglés Canonical Data Model) que constituye una estructura global independiente de las aplicaciones que brinda un formato único para las comunicaciones, logrando que aplicaciones que trabajan fuentes de datos semejantes, pero definidas con diferente estructura y formato logren entenderse entre ellas [1].

Existen numerosas empresas que han desarrollado soluciones y tecnologías que dan solución a problemas de integración usando los estilos arquitectónicos SOA (del inglés Service Oriented Architecture) y EDA (del inglés Event Driven Architecture), tales como: IBM, Oracle y Progress Software [3].

El desarrollo de un MDC no requiere la adopción de SOA para las aplicaciones a integrar ya que las transformaciones de datos ocurren en una capa intermedia entre las aplicaciones y por lo tanto el impacto en las aplicaciones es nulo.

El valor de SOA está dado por la capacidad de expresar capacidades técnicas en el marco de los negocios y de recombinarlos rápidamente según sea la necesidad de la empresa, convirtiéndose en la solución flexible que los negocios necesitan. Esto ofrece facilidades a la hora de actualizar un proceso de negocio pues solo se tendría que alterar el o los servicios involucrados a este, de otra manera habría que actualizar y re-implementar toda la aplicación. SOA indica, entre otras cosas, que se deben tener claro los criterios con respecto a la compatibilidad con recursos futuros, se deben hacer los servicios reusables, pensar en las comunicaciones asíncronas y en la interoperabilidad. La integración de los sistemas es otra de las grandes ventajas que ofrece SOA, esto resultaría sumamente útil en empresas que gestionan diferentes sistemas de información. [4]

La Universidad de las Ciencias Informáticas (UCI) es un centro que conjuga los procesos de formación del profesional de informática con el desarrollo de proyectos de software, por lo cual aporta a la sociedad profesionales bien capacitados y comprometidos, así como la generación de servicios y productos que con sus ingresos contribuyen al país y a la sociedad cubana. Gran parte del buen funcionamiento de la misma se debe a la infraestructura tecnológica que existe internamente en la universidad, que proporciona gran disponibilidad de la información. Para ello existe una gran diversidad de aplicaciones que gestionan

información referente a los servicios telemáticos, capital humano, gestión académica, servicios de reservación, o de consulta de información de cualquier índole.

Hoy día, es interés de la universidad que se integren estas aplicaciones. El principal problema que se encuentra es que no hay una vía aparente para comunicarlas. Estos sistemas fueron concebidos de forma monolítica, es decir, la estructura está basada en grupos funcionales altamente acoplados y por esta razón no logran comunicarse entre ellos. Al estar dichas aplicaciones y sistemas independientemente construidos, con insuficientes o nulos mecanismos de integración entre sí por la no existencia de un entorno o infraestructura de integración, se ocasiona que:

- La comunicación e intercambio de datos entre los sistemas no esté regulado y estandarizado.
- Redundancia en el manejo de información. Se duplican los datos sin necesidad al existir diferentes bases de datos que contienen repetidamente los mismos recursos lo que conlleva a que los servidores están sobrecargados, de ahí que,
- Hay escasa reutilización: Los esfuerzos se multiplican al tener que re implementar funcionalidades que ya existen y pudieran ser aprovechadas.
- No hay transparencia en las operaciones: pues no se conoce como funcionan internamente las funcionalidades de un proceso de negocio dado.
- Se ve afectada la integridad de la información.
- No son lo suficientemente flexibles los procesos de negocio: pues no se pueden transformar fácilmente las funcionalidades de los sistemas en actual uso en la universidad.
- Para el usuario final se dificulta el manejo de la información.

Por la **situación problémica** antes expuesta se define como **problema de la investigación**: ¿Cómo estandarizar la comunicación que se intercambia entre los sistemas de una organización?

El **objeto de estudio** lo constituyen la integración de sistemas empresariales y el **campo de acción** se enmarca en la integración de sistemas mediante Modelos de Datos Canónicos.

Como **objetivo general** se propone: Elaborar una estrategia para el desarrollo de un modelo de datos canónicos que estandarice el intercambio de mensajes entre los sistemas de una organización dentro de una solución de integración.

La **idea a defender** establece que si se logra desarrollar una estrategia para el desarrollo de un modelo de datos canónicos se logrará la integración entre los sistemas informáticos mediante estandarización de los mensajes que se envían los sistemas.

Para darle cumplimiento al objetivo general se presentan los siguientes **objetivos específicos**:

- Realizar un estudio sobre el estado del arte en cuanto a los conceptos relacionados a las soluciones de integración.
- Elaborar la propuesta de solución haciendo uso de los aspectos estudiados en la fundamentación teórica, describiendo los procedimientos que la componen.
- Aplicar y evaluar la estrategia resultante en el escenario de la UCI, haciendo uso de un caso de estudio práctico y mediante el Método Experto.

A fin de dar cumplimiento a los objetivos antes descritos se definen las **tareas de investigación** siguientes:

- Elaborar un análisis sobre las tendencias actuales sobre las soluciones de integración y valorar las influencias que pudieran tener sobre la solución.
- Seleccionar lenguajes, tecnologías y herramientas a partir de un estudio de las existentes en el mercado con el objetivo de utilizarlas en la elaboración de la solución.
- Elaborar la Estrategia para desarrollo de un MDC apoyándose en el estudio teórico previamente elaborado.
- Validar la efectividad y calidad de la propuesta desarrollada exponiendo servicios en la red de la UCI y mediante el Método Experto.

Para dar cumplimiento a las tareas anteriormente descritas será llevada a cabo una investigación exploratoria y explicativa en la que se estudian primeramente las soluciones de integración ya desarrolladas y se valoran las posibles influencias que tienen en la situación problémica actual. Se propone además una solución descrita de forma tal que se establecen los procedimientos para el desarrollo de una solución de integración que minimice los efectos de los fenómenos descritos en el problema.

Al dar cumplimiento a cada una de las tareas de investigación abordadas, los **resultados esperados** se centran en la obtención de una estrategia para el desarrollo de un modelo de datos canónicos que permita

la unificación de datos e integración de sistemas en la Universidad de las Ciencias Informáticas, de modo que las tecnologías de información se identifiquen con los procesos de negocio, facilitando las actividades que se desarrollan en el centro, y se ahorren en esfuerzos y recursos. Se obtendrán además un conjunto de artefactos de entrada y salida relacionadas a la estrategia que apoyan su desarrollo.

La estructura del presente trabajo diploma se define:

- **Capítulo 1:** En este capítulo se describen los conceptos fundamentales que engloban el desarrollo de MDC, así como las tecnologías, seleccionando la más idónea según los requisitos de integración e interoperabilidad y las características de los sistemas que presentan deficiencias en la comunicación.
- **Capítulo2:** Se desarrolla la propuesta de solución para la cual se hace una descripción general de la misma. Se describe y especifican los aspectos que intervienen en el desarrollo de la solución. Se detalla la estrategia mediante el uso de diagramas. Se especifican además los patrones utilizados en dicho desarrollo.
- **Capítulo 3:** Se exponen los resultados de la aplicación de la estrategia en el escenario de la UCI. Luego se realiza la validación de la solución y se resumen los resultados de la misma que garantizan la aceptación de la solución.

Capítulo 1. Fundamentación Teórica

Introducción

En el presente capítulo se muestran los principales conceptos relacionados con los Modelos de Datos Canónicos, para luego analizar las herramientas y procesos de desarrollo que se utilizan en la integración de aplicaciones usando este patrón en el escenario de la Universidad de las Ciencias Informáticas (UCI). Se presenta además el estado del arte referente al desarrollo de modelos de datos canónicos como patrón de integración en el mundo.

1.1. Consideraciones acerca la Integración de Aplicaciones

Cuando se pretende integrar la información que se gestiona en una empresa no es suficiente con desarrollar una única aplicación que contenga todos los datos que hasta el momento se conocen, pues esto soluciona el problema temporalmente, pero pronto comenzará a generar otros problemas de integración, por ejemplo: al necesitar adicionar nueva información se requerirá desarrollar toda la aplicación desde el principio, producto del alto acoplamiento y las dependencias dentro de este gran sistema. Una aplicación que cubriría esta necesidad de integración sería capaz de garantizar la interoperabilidad mediante el bajo acoplamiento de las aplicaciones existentes que se comunicarían entre sí. Mediante la mensajería asincrónica sería capaz de coordinar las llamadas en pos de hacer la comunicación más fiable y gestionada. Una aplicación distribuida tiene como característica el alto acoplamiento, la comunicación se desarrolla de manera sincrónica, y por lo general tiene una gran dependencia de interacción humana, lo que proporciona desagrados al usuario ante lentos tiempos de respuesta [1].

Por otro lado, las aplicaciones integradas son independientes de las aplicaciones debido al bajo acoplamiento que permite que se ejecute independientemente de las aplicaciones que coexisten en la integración. Los desarrolladores de soluciones de integración han usado estilos como [1]:

- **Transferencia de archivos:** donde una aplicación escribe un archivo que es leído luego por otra. Para esto las aplicaciones deben estar de acuerdo en cuanto a formato de los datos, localización y nombre del archivo, el momento en que será leído o escrito y quién borrará el archivo.

- **Base de Datos compartida:** donde múltiples aplicaciones comparten una misma base de datos. Esto evita duplicar datos y tener que transferirlos de una aplicación a otra.
- **Invocación a proceso remoto:** donde una aplicación expone algunas de sus funcionalidades que pueden ser invocadas remotamente por otras aplicaciones como proceso remoto.
- **Mensajería:** una aplicación envía un mensaje a un canal de mensajería común. Otras aplicaciones pueden leer el mensaje desde el canal de mensajería en un momento posterior. Las aplicaciones deben estar de acuerdo con el canal de mensajería, así como con el formato del mensaje. La comunicación es asíncrona

Entre estos estilos se escoge la mensajería ya que propicia [1]:

- Bajo acoplamiento.
- Comunicación remota.
- Integración de múltiples plataformas/lenguajes.
- Comunicación asíncrona.
- Comunicación confiable.
- Operaciones sin conexión.
- Mediación.
- Procesamiento en paralelo.
- Evita la saturación.

1.1.1. Patrones de Integración

Dentro del estilo de integración basado en la mensajería se proponen una serie de patrones de integración que fueron agrupados según su funcionamiento o la posición que tienen en la comunicación entre los sistemas. Existen grupos como Destino de mensajería, Construcción de mensaje, Canal de mensajería, Ruteo de mensaje, Transformación de mensaje y Gestión del sistema.

Patrones para la transformación de mensajes propuestos por Hohpe y Woolf

En una organización en la que diferentes aplicaciones gestionan los procesos de negocio suele suceder que cada una de estas solo toma los atributos de las entidades que necesita para el desarrollo de sus propios procesos. Puede darse el caso que un mismo recurso se utilice en más de un sistema y tenga atributos que son significativos en uno, mientras que en otro tenga otras especificaciones importantes para el desarrollo de sus procesos propios, o que no estén definidos de la misma manera y formato. Esto es un

problema ante el intercambio de mensajes para la integración, por lo que se debe proceder a la transformación de los mismos para que las aplicaciones se puedan entender, dado que otra solución como cambiar el formato de las aplicaciones independientemente, sería costosa y difícil.

Un traductor de mensajes es un patrón que solucionaría problemas como estos, pues transforma el formato de datos de una aplicación a otro.

Para llevar a cabo la transformación existen los siguientes patrones:

- **Envelope Wrapper (Enrollador de sobre):** Es un patrón de integración que encapsula el mensaje en un paquete para su transporte. Este encapsulado se hace con el objetivo de que el formato del mensaje resultante coincida con el del sistema de mensajería, si sucediese que el mensaje original no cumpliera con sus especificaciones. En este proceso se pueden alterar campos en el encabezado del mensaje, así como adicionar nuevos campos que pueden ser extraídos del cuerpo del mensaje original, además de encriptar el cuerpo del mensaje o añadir credenciales de seguridad. Es además común encontrar encapsulaciones en cadena, donde en un paquete existan más de una encapsulación anidada. Una vez que el paquete ha llegado a su destino se hace la operación inversa de lo que se hizo para encapsularlo, obteniendo finalmente el mensaje original [1].
- **Content Enricher (Enriquecedor de contenido):** Cuando a un sistema destinatario le llega un mensaje en el que le faltan campos según su formato este patrón se encarga de solicitarlos en una fuente de datos externa [1].
- **Content Filter (Filtro de contenido):** Es un patrón que resuelve los problemas opuestos a los del patrón antes descrito (Content Enricher). Cuando el sistema destinatario recibe un mensaje con campos que no necesita o no están de acuerdo con su formato el Content Filter los elimina. Es además una solución cuando el sistema remitente no debe mostrar cierta información por problemas de seguridad, filtrándola del mensaje a enviar. También se usa no solo para filtrar campos, sino también para simplificar mensajes eliminando información redundante y dejando una lista de los elementos que pueden ser mejor entendidos y más útiles a los demás sistemas. Puede usarse también para dividir un mensaje en varios, con campos específicos obtenidos de original [1].

- **Claim Check (Chequeo de reclamo):** Al igual que el Content Filter permite eliminar información poco relevante del mensaje, solo que esta vez es de forma temporal, de forma que si se necesitaran nuevamente estos datos se pueden recuperar ya que estos se guardan en un almacén persistente que pudiera ser una base de datos o un fichero. Puede ser utilizado cuando el sistema de mensajería tiene un límite de tamaño para el mensaje y que este encaje en el sistema sin necesidad de que pierda la información que contiene. La recuperación de los datos se hace mediante una llave única generada que permita relacionar la información con el mensaje [1].
- **Normalizer (Normalizador):** Este patrón traduce los mensajes a un formato específico. Cuando llegan varios mensajes de formatos diferentes al aceptado, este los enruta a los traductores de mansajes que le corresponde a cada formato, pues debe existir un traductor para cada formato entrante. La salida, una vez que se han traducido todos los mensajes, son los mensajes traducidos al formato común [1].

- **Modelo de datos canónicos (Canonical data model, MDC):**
  El modelo de datos canónicos intenta minimizar las dependencias cuando se trata de integrar aplicaciones que tienen distintos formatos internos. Este patrón es independiente de las aplicaciones y resuelve el problema de la traducción de mensaje: se necesita uno para cada par de aplicaciones, de modo que si se quiere integrar otras aplicaciones el número de traductores crecería de manera exponencial. El modelo de datos canónicos es unidireccional pues provee un formato común con el que todas las aplicaciones deben estar de acuerdo. Por lo cual para enviar un mensaje, este debe pasar primero por el MDC, que será único en la integración. Es una solución muy eficaz cuando se integran muchas aplicaciones, no siendo así cuando incurren en la integración un número pequeño de aplicaciones. Para materializarlo se necesita de la implementación de otros patrones de transformación de mensajes que traduzcan los formatos al del modelo de datos [1].

  Uno de los beneficios del uso de un MDC es que este desacopla los sistemas a nivel de formatos de mensajes. Los sistemas no tienen que hacer suposiciones de tener que depender de los formatos de datos de los demás sistemas. Esto constituye un aspecto importante que tributa al débil acoplamiento. Otro beneficio es que la definición de formatos canónicos de mensajes crea la posibilidad de suministrar a la empresa un catálogo inequívoco de los mensajes disponibles sobre los eventos de negocios, que representan valiosos activos de negocio. Los eventos de negocios en

este catálogo son independientes de las fuentes que generan estos mensajes. Basadas en este catálogo las políticas pueden ser implementadas con respecto a la propiedad y el grado de libre disponibilidad de los datos que se intercambian entre dominios.

Por otro lado el uso de un modelo de datos canónico puede parecer excesivamente complicado si solo un pequeño número de aplicaciones participan en la solución de integración [1]. Además a veces puede requerir más esfuerzo de personalización que la construcción de un modelo desde cero [5].

Indicaciones para el diseño del MDC

El objetivo de un MDC es tener una representación estandarizada de las entidades de negocio en un entorno de integración. Con el fin de crear una estructura formal de representación de datos, los mensajes tienen que ser vistos como un conjunto de entidades comerciales y no meras representaciones textuales de la información necesaria para invocar un servicio [6].

Para la identificación de las entidades de negocio se debe tener en cuenta que solo se identificarán las entidades que serán representadas en el MDC, o sea, que estén involucradas en la integración. Por otra parte, el MDC intenta minimizar el impacto de los cambios. Para que esto se cumpla, se debe garantizar que los cambios en el MDC sean también mínimos. Solo deben adicionarse detalles acerca las entidades de negocio, pequeñas cantidades de atributos, sin poder eliminar ni renombrar atributos del MDC [6].

Existen ciertas características que deben tener los mensajes de un MDC, las que colaborarían a que el MDC sea más confiable, robusto y ágil [6]:

- **Identificable:** es importante los que los mensajes del MDC sean identificables unos con respecto a otros, pues puede ser necesario obtener información respecto a estos como la fecha de creación, fuente, destino, etc.
- **Trazable:** la comunicación asíncrona tiende a dejar un rastro confuso con los mensajes de respuesta si no es trazable con los mensajes de petición. Puede darse el caso en el que los mensajes son de gran tamaño por lo que existe la necesidad de dividirlo en varias partes, como pequeños mensajes. Cada uno será procesado en paralelo por el mismo proveedor o por diferentes proveedores. En situaciones como esta la respuesta de cada mensaje debe ser asociada al mensaje original.

- **Seguro para la Información:** el mensaje debe ser capaz de garantizarse él mismo la seguridad de los datos.
- **Enrutable:** hay ocasiones en que el destino de un mensaje no se conoce hasta el momento de enviarlo. Para manejar estas situaciones, los mensajes deben tener información de ruteo que garantice que el mensaje sea enrutado correctamente.
- **Resultados rastreables:** un mensaje puede ser procesado por diferentes sistemas antes de ser finalmente consumido. Es importante rastrear el resultado del procesamiento del mensaje en cada nodo, así se establece una forma de localizar el nodo en el que hubo error, en caso de haberlo.

1.2. Modelos que intervienen en el diseño de un MDC

Existe una interrelación entre los modelos que se usan para definir y analizar las estructuras de datos, la implementación de los servicios y su plataforma. Todo esto se hace a través de la especificación del modelo de datos canónicos pues no es más que la representación común de las entidades, sus atributos y relaciones basadas en los requerimientos de un proyecto SOA. Este se debe alinear con el glosario de negocio, los procesos, el modelo de servicio, el modelo de mensaje y el modelo de datos físicos. Esto asegura la semántica y la interoperabilidad estructural[3].

1.2.1. Modelo de datos conceptual

Es básicamente un modelo de datos canónicos visto desde el mayor punto de abstracción posible que maneja la especificación inicial del modelo. Representa los requerimientos de información estratégica de la empresa dentro del ámbito del proyecto SOA. Los principales elementos en la construcción de un modelo de datos conceptual son las entidades del negocio, pues representan la agrupación de los atributos alrededor de uno solo llamado llave, que resume el concepto de información. Este modelo se hace generalmente en la fase inicial del desarrollo e incluye solo las entidades del negocio y sus principales relaciones [3].

El desarrollo de un modelo de datos conceptual provee una representación compartida de los principales grupos de alto nivel de información usada en el análisis de negocio, modelamiento de caso de uso e identificación de servicios candidatos en un proyecto SOA. Provee además puntos de partida y límites para los próximos modelamientos dentro del proyecto. Este modelo permite al equipo de desarrollo

demostrar una comprensión de la estructura y el contenido de los requerimientos de información del cliente.

1.2.2. Modelo de Datos Lógicos

Está basado en el modelo de datos conceptual y adiciona nuevos detalles dentro del mismo modelo aunque se representa de manera separada. Su principal objetivo es proveer entradas al desarrollo de estructura de datos o las representaciones de datos a través de las capas de la arquitectura. Puede ser además usado en el análisis de servicio y ser expuesto a los consumidores de servicios como un modelo de proceso de negocio [3].

El diseño de un Modelo de Datos Lógicos que se adapte a las necesidades de la empresa es fundamental, no solo porque refleja el compromiso de tratar los datos como un activo real de la empresa, sino también porque permite el almacenamiento eficiente y eficaz de los datos a los que las empresas pueden acceder fácilmente para crear diversas informaciones y al conocimiento de los productos [7].

Las normas que rigen los dominios específicos de los valores permitidos, los rangos de los valores permitidos u otras restricciones sobre los valores lógicos de datos se especifican en este modelo, esto incluye restricciones cruzadas de los atributos. El hecho de que estas normas son parte del modelo de datos lógico, no se debe asumir que debe ser protegido por una base de datos. El cumplimiento de las normas de datos es responsabilidad del servicio que mantiene a estas entidades. El servicio puede usar un servicio de base de datos o uso de componentes de la aplicación para hacer cumplir estas reglas de datos lógicos. Las decisiones relativas a cómo las reglas se hacen cumplir y por cuál de los servicios son un aspecto clave de la gobernabilidad de datos [3].

1.2.3. Modelo de Análisis del Servicio

Representa la estructura de datos para la especificación de componentes y servicio. La estructura que se expresa dentro de este modelo no es más que la reflexión del modelo de datos conceptual o el de datos lógicos. El objetivo de un modelo de análisis del servicio es orientar a la especificación de componentes de software y servicios de manera que sea compatible con la arquitectura de datos global. En muchos casos un modelo de componentes puede ser definido desde los modelos de mensaje canónico y las especificaciones de componentes se derivan de estos. En este caso, el modelo conceptual aún guía la definición de los tipos de negocios dentro del componente y pasa a través de las interfaces de los

componentes. Esto conlleva los beneficios adicionales de la alineación de las estructuras de datos a través de la persistencia de datos, componentes de software y las definiciones de servicio[3].

1.2.4. Modelo de Mensaje Canónico

Representa el formato estandarizado con el que la información se intercambia en el bus de mensaje. Provee el formato de intercambio de datos por defecto por lo que las aplicaciones solo deberán conocer el formato propio y el que está definido en él, que está compartido en el bus de servicio. En él son definidos un conjunto de mensajes donde cada uno incluye un conjunto relacionado de tipos de datos previamente definidos, además de elementos y atributos estructurados para proveer un documento de negocio con especificaciones de la semántica y el contexto [3].

Para obtener los modelos de mensaje canónico existen varios métodos, que pueden ser usados en dependencia de las características de cada proyecto. En el método top-down (de arriba hacia abajo), el modelo de análisis de servicio guía el modelamiento y diseño del servicio para luego llegar a las especificaciones técnicas del servicio, especificaciones de componentes, y el modelo de clases a nivel de diseño, el que incluye los tipos de datos y las estructuras que componen el modelo de mensaje canónico. Los proyectos que se centran en la integración del ESB(del inglés Enterprise Service Bus), y en el método bottom-up (de abajo hacia arriba), el modelo de mensaje canónico puede ser creado directamente en el esquema XML (del inglés Extensible Markup Language) manualmente, o como resultado del diseño de flujo de mensajes o las herramientas de generación de código de servicio [3].

La forma más común de representar un Modelo de Mensaje Canónico es usar un conjunto de documentos XML, esto trae como ventaja que las definiciones de los mensajes se pueden reutilizar en el esquema WSDL (del inglés Web Services Description Language) que expone los servicios [3].

1.3. Enfoques para definir una forma canónica

A la hora de definir una forma canónica para un objeto se tienen en cuenta dos formas, de las que se encoge una: crear un gran conjunto de datos con la información del negocio, o solo representar el mínimo posible conjunto de información [8].

Una de las soluciones es utilizar toda la información del sistema fuente que tarde o temprano la aplicación pueda necesitar. Esta es evidentemente una solución que tendrá como consecuencia que la forma canónica será innecesariamente grande, conteniendo información irrelevante para otras aplicaciones. La

ventaja de esta solución es que una vez creada la forma canónica no será necesario modificarla en un futuro cercano, solo cuando aparezcan nuevas aplicaciones con nueva información [8].

Otra solución incluye no solo lo que se tiene en el sistema fuente, sino que también puede incluir la información que las aplicaciones puedan necesitar en un futuro. Al igual que la anterior se crea un conjunto de datos muy grande con información innecesaria, además de que se necesitaría un gran tiempo para determinar todo este set de datos que se utilizaría en un futuro. Esto último es poco probable, pues esta información es propensa a cambiar, y al haber diseñado todo un set de datos puede que después sea inútil y se haya invertido tiempo y esfuerzo en vano. Si se lograra definir y predecir toda la información de forma certera pues entonces no se tendrá que modificar la forma canónica en mucho tiempo [8].

Crear una forma canónica base, a partir de la que se vaya adicionando información según se vayan incorporando las aplicaciones, es otra solución que comienza con un set de datos pequeño pero que se convertirá en uno grande mientras más aplicaciones se incorporen [8].

La solución más simple de hacer es generar solo el mínimo de datos que identifica a cada objeto como único. Esto trae como consecuencia que los mensajes no contendrán toda la información que la aplicación necesita por lo que se tendrá que implementar un servicio extra que se encargue de recuperar estos datos [8].

Patrón de diseño Esquema Canónico propuesto por Thomas Erl

El Esquema Canónico (EC) no debe ser confundido con el MDC ya que este último asume que necesariamente para establecer la comunicación entre aplicaciones debe existir la traducción de mensajes. De manera contraria el EC asume que las aplicaciones están de acuerdo y se conforman con la forma canónica definida en un entorno, evitando la necesidad de la trasformación, asegurándose de que el servicio es compatible con los esquemas desde el principio de su diseño. Esto se logra mediante la aplicación de estándares de diseño de datos a los modelos de datos que se involucran con el servicio [9].

El EC es comúnmente aplicado a los servicios implementados como servicios web, ya que permite definir los modelos de datos en la estructura estándar XML. Las definiciones en los esquemas XML representan el mismo tipo de información o documentación que mantiene la alineación de los tipos de datos complejos y simples que recuerdan la sincronización a través de los diferentes servicios [9].

Una vez estandarizados los esquemas, este patrón es realizado mediante un proceso formal, a través del que los servicios son diseñados. Esto asegura que la aplicación sea consistente desde el principio del diseño del servicio estandarizado [9].

En grandes empresas la estandarización de modelos de datos puede necesitar ser limitada a los dominios individuales para hacer el esfuerzo de estandarización y las responsabilidades consecuentes de gobierno más manejables. Incluso este patrón propone que se hagan inventarios de dominio sobre el inventario de la empresa.

Patrones de transformación propuestos por Thomas Erl

Estos patrones resuelven básicamente los retos que tiene el logro de la interoperabilidad cuando se entregan servicios basados en las aplicaciones. Cada patrón ofrece una capa intermedia de procesamiento intermediario que tiene impacto similar en el desarrollo, complejidad del diseño y esfuerzo.

Transformación del Modelo de Datos

Los modelos de datos con los que una aplicación tiene definida su estructura de información generalmente no son iguales a los modelos de datos de otras aplicaciones con las que se quiera integrar. Este patrón soluciona las incompatibilidades de tipo de modelo de datos cuando los servicios están representados por diferentes esquemas con respecto a un mismo dato, impidiendo la interacción de los servicios [9].

Para darle solución a este problema se incorpora una tecnología a fin de convertir los datos entre sistemas con esquemas dispares, para lo que el mapeo de la lógica debe ser desarrollado y desplegado de acuerdo a un Modelo de Datos que puede ser dinámicamente convertido para cumplir con los diferentes modelos de datos, cumpliendo con los principios de reusabilidad y estandarización. Esto trae como consecuencia que se requiera un gran esfuerzo para el desarrollo, además de complejidad del diseño [9].

Transformación de Formato de Datos

Este patrón soluciona el problema de incompatibilidad del formato de datos que impide la comunicación. Por ejemplo: si un sistema está diseñado para recibir información en un formato estándar, será imposible para él entender cualquier otro formato. Para darle solución se necesita una transformación lógica intermediaria de los formatos de los datos, con el objetivo de traducir de un formato de datos a otro. Para ello se adicionan servicios lógicos internos, servicios agentes, o un servicio dedicado a la transformación, cumpliendo con los principios de bajo acoplamiento y estandarización. Al igual que la transformación de

los Modelos de Datos, la aplicación de este patrón incorpora grandes esfuerzos y complejidad del diseño [9].

Como la transformación del formato de datos afecta también al Modelo de Datos, este patrón debe ser aplicado en paridad con el de transformación de modelo de datos [9].

Puenteo de Protocolo

Este patrón soluciona el problema de incompatibilidad de protocolos de comunicación. Para darle solución un puenteo lógico es introducido para establecer la comunicación entre diferentes protocolos de comunicación, mediante la conversión dinámica de un protocolo a otro en tiempo real [9].

Una capa de puenteo de protocolo está compuesta por una serie de adaptadores que actúan sobre los protocolos de transporte dados. Cuando un puente de protocolo recibe un mensaje, este lo transforma de forma que cumpla con el protocolo destino [9].

1.4. Enfoques asociados a la integración de sistemas

La mensajería como estilo de integración introduce términos como bajo acoplamiento, comunicación asíncrona, servicio, entre otros.

Servicio

Un servicio se define como “una unidad de solución lógica la cual la orientación a servicios ha sido aplicada en un grado significativo. Es la aplicación de los principios de diseño orientado a servicios que distingue una unidad de la lógica como servicio en comparación con las unidades de la lógica que pueden existir simplemente como objetos o componentes.” [9]

Los servicios deben ser independientes de las aplicaciones, así como lo suficientemente atómicos como para ser reutilizables. Un servicio bien simple podría ser la suma de dos números, esto constituye una función útil en muchísimos sistemas y de estar implementada como servicio se puede aprovechar en los que los necesiten. Otros servicios más complejos se encargarían de otras funcionalidades más complejas en las que se podría implementar cualquier proceso del negocio o incluso orquestar otros servicios ya existentes.

1.4.1. Arquitectura Orientada a Servicios

En el ámbito empresarial se conoce como cambia rápidamente el mercado y por consiguiente los procesos de negocio. Si la empresa no es lo suficientemente flexible, corre el peligro de no ser lo suficientemente apta para subsistir. En los momentos actuales, cada empresa cuenta con un sistema informatizado que lleva el control de esta, automatiza actividades, realiza actividades de contabilidad y facilita de una manera u otra la gestión empresarial, pero siempre de manera que se ajuste fácilmente al constante cambio de los procesos. Por este motivo en muchos negocios se ha adoptado SOA como enfoque para sus sistemas, estilo arquitectónico que como su nombre lo indica, orienta el desarrollo sobre la línea de los servicios [4, 9, 10].

SOA representa un modelo arquitectural que tributa a que una empresa sea ágil y sus costos sean efectivos y controlados, mientras que reduce la carga de tecnología en la organización. Esto se lleva a cabo mediante el posicionamiento de servicios como el medio principal a través del cual se representa la lógica de la solución [9].

SOA permite que se mejoren sistemas integrados de forma que satisfagan las necesidades del negocio, además prevé la implementación de una plataforma de servicios que responden a los procesos de negocio que pueden ser combinados en diferentes soluciones, posibilitando que el negocio y las tecnologías de información se entiendan, así mismo logra flexibilidad ante las situaciones cambiantes de las empresas actuales [11].

Coincidentemente con las definiciones dadas se puede definir SOA como el paradigma arquitectural que introduce los servicios en el desarrollo de procesos empresariales a fin de lograr interoperabilidad, reusabilidad y atomicidad, haciendo del producto resultante una solución satisfactoria y tentadora ante los problemas de los cambiantes mercados actuales. Las soluciones SOA son flexibles y pueden contar con la capacidad de intercomunicarse con otros sistemas, características únicas que hacen de SOA una opción perfectamente viable para la solución de un problema de integración.

En una arquitectura SOA la integración se comporta transversal a la composición de procesos de negocio, los servicios y el acceso a datos y sistemas legados.

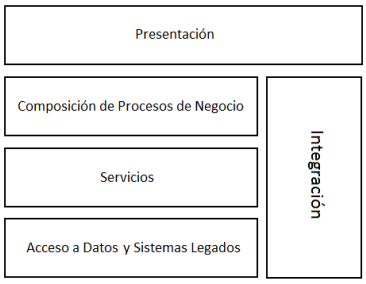


Figura 1. La capa integración en una Arquitectura SOA

Evento

Un evento es algo significante que sucede dentro del negocio. Puede significar un problema, un problema inminente, una oportunidad o una desviación. El término es usado para referirse tanto a la definición y especificación del evento como a cada instancia de este. Para que un evento sea significativo este debe ser descrito en los términos del negocio y no en los de la aplicación [12].

Cada ocurrencia de un evento tiene un encabezado y un cuerpo. El encabezado del evento contiene elementos que describen la instancia del evento, como pueden ser: identificadores, tipo de evento, nombre del evento, creador del evento, número de instancia de evento y otras. Estos elementos son consistentes sobre la especificación del evento [12].

El cuerpo del evento debe estar completamente descrito para que cualquier parte interesada pueda usar la información sin tener que volver a la fuente. Para asegurarse de que todos los eventos fueron correctamente entendidos por todos los consumidores se debe desarrollar un diccionario de términos de negocio o una ontología [12].

1.4.2. Arquitectura dirigida por eventos

La arquitectura dirigida por eventos (EDA, del inglés Event-Driven Architecture) define una metodología para el diseño e implementación de aplicaciones y sistemas en los que los eventos se transmiten entre componentes de software desacoplados y servicios. Esta arquitectura no reemplaza, sino que complementa a SOA. Mientras que SOA es más apropiado para el intercambio petición/respuesta, EDA introduce capacidades de procesos asíncronos de larga duración. Un nodo EDA anuncia los eventos y no depende de la disponibilidad de un servicio publicado. Está realmente desacoplado con los otros nodos. EDA es a veces llamada como “SOA dirigida por eventos” [13].

Por naturaleza una arquitectura dirigida por eventos está débilmente acoplada y altamente distribuida. La fuente solo conoce el evento en sí, y no las partes interesadas asociadas a este ni sus procesos consecuentes. La trazabilidad de un evento a través de una red de eventos dinámica puede ser difícil. Por lo tanto, la arquitectura dirigida por eventos es la más comúnmente usada para flujos de información y trabajo asíncronos [12].

Las principales características de la arquitectura dirigida por eventos son [14]:

- **Comunicación de difusión:** los sistemas participantes emiten eventos a cualquier parte interesada. Más de una parte puede escuchar el evento y procesarlo.
- **Asincronía:** la publicación de sistemas no espera por una respuesta para procesar los eventos.
- **Eventos finamente granulados:** las aplicaciones tienden a publicar eventos individuales como opuesto a un simple evento agregado.
- **Ontología:** los sistemas globales definen una nomenclatura para clasificar eventos, generalmente como una jerarquía. Al recibir mensajes se puede a menudo expresar interés en un evento en particular o en una categoría de eventos.
- **Procesamiento complejo de eventos:** el sistema entiende y monitorea las relaciones entre los eventos.

1.5. Lenguajes de Modelado

Un lenguaje de modelado es un conjunto estandarizado de símbolos y de modos de disponerlos para modelar un diseño de software. Específicamente para el desarrollo de soluciones de integración se usan generalmente los lenguajes de modelado UML (del inglés Unified Modeling Language) y XML.

1.5.1. UML

UML es un lenguaje gráfico, lo que permite documentar y especificar un sistema de software de manera estándar incluyendo aspectos conceptuales como los procesos de negocio y las funcionalidades del sistema. Cuenta con una notación estándar y semánticas esenciales para el modelado [15].

Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales, tales

como procesos de negocios y funciones del sistema, además de aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables [16].

Una de las metas principales de UML es avanzar en el estado de la integración institucional proporcionando herramientas de interoperabilidad para el modelado visual de objetos. Para ello se le definió a UML una semántica y una notación estándar [15].

1.5.2. XML

XML es una forma flexible de crear formatos de información común y compartir los datos y el formato en la red [17]. Consiste de una serie de reglas, pautas o convenciones para planificar formatos de texto para los datos, de manera que produzcan archivos que sean fácilmente generados y leídos por un ordenador, que sean inequívocos y que eviten los problemas más comunes como la falta de extensibilidad, la falta de interoperabilidad entre plataformas o la falta de soporte para universalizar su tratamiento [18].

XML tiene punteros a la estructura de los datos, lo que ahorra tiempo y simplifica el software de aplicación. Además no dispone de soporte para excepciones, por lo que cada etiqueta realiza siempre la misma función. Posee independencia de los navegadores y del sistema de objetos, porque en lugar de añadir etiquetas de presentación al documento se remite a una hoja de estilo realizada en XSL (Extensible Style Language). Es un lenguaje que permite jerarquizar y estructurar la información y describir los contenidos dentro del propio documento, así como la reutilización de partes del mismo [18].

1.6. Herramientas de Modelado

La herramienta de modelado que se seleccione para el desarrollo de una solución de integración debe ser capaz de soportar el diagrama de clases de UML así como tener la capacidad de generar ficheros XML. Además debe ser flexible a la hora de realizar un cambio en el diseño. Una herramienta ideal admitiría diseño desde el inicio hasta el fin del proyecto, así como diseño inverso, con esquemas amplios para documentar detalladamente los procesos [19].

1.6.1. Visual Paradigm para UML

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad. Permite modelar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y

generar documentación. Soporta las últimas versiones de UML y la Notación y Modelado de Procesos de Negocios (BPMN). Además provee el modelado de procesos de negocios (BPM) y un generador de mapeo de objetos-relacionales para los lenguajes de programación Java .NET y PHP [20].

Facilita a las organizaciones la diagramación visual y el diseño de sus proyectos de sistema que les brinda la posibilidad de integrar y desplegar sus aplicaciones empresariales y sus bases de datos. Proporciona el código y compatibilidad hasta con diez lenguajes y soporta un conjunto de lenguajes tanto en la generación del código como en la Ingeniería Inversa [20]. Soporta la importación y exportación de XML de versiones 1.0, 1.2 y 2.1 a fin de maximizar la interoperabilidad de los productos de Visual Paradigm para UML con otras aplicaciones.

1.6.2. Enterprise Architect

Es una herramienta que soporta UML 2.1 y se puede adquirir por un bajo coste de licencias. Se caracteriza por su alto rendimiento e interfaz intuitiva. Cubre el desarrollo de software desde el paso de los requerimientos a través de las etapas del análisis, modelos de diseño, pruebas y mantenimiento. Enterprise Architect es una herramienta multi-usuario, basada en Windows, diseñada para ayudar a construir software robusto y fácil de mantener. Ofrece salida de documentación flexible y de alta calidad. El manual de usuario está disponible en línea [21].

Provee generación de documentos y herramientas de reporte con un editor de plantilla. Soporta generación e ingeniería inversa de código fuente para muchos lenguajes, incluyendo C++, C#, Java, Delphi, VB.Net, Visual Basic y PHP. También hay componentes gratis para CORBA y Python. También vende puentes livianos para Eclipse o Visual Studio.Net, permitiendo modelar y saltar directamente al código fuente en el editor deseado. Las plantillas de generación de código permiten personalizar el código fuente generado a las especificaciones de requeridas [21].

1.6.3. Rational Rose

Es una poderosa herramienta para el modelado visual. Es usado para modelar cualquier sistema antes de que se escriba código alguno, de esta forma se puede asegurar una arquitectura robusta desde el principio del desarrollo del software. Soporta el modelado de procesos de negocio, ayudando a entender el negocio alrededor del sistema [22].

Genera códigos para aplicaciones Ada, ANSI C++, C++, CORBA, Java, J2EE, Visual C++ y Visual Basic a partir de los modelos. Es compatible con el lenguaje UML y es uno de los productos más completos de la familia Rational. Soporta patrones ANSI C++, Rose J y Visual C++, Enterprise JavaBeans 2.0, e ingeniería directa e inversa para algunas de las construcciones más comunes de Java. Es capaz de analizar la calidad del código y de generar código gracias a las capacidades de sincronización configurable entre el modelo y el código, además de una gestión más detallada y el uso de modelos con la función de componentes de modelos controlables por separado. Incluye un complemento de modelado web que proporciona la capacidad de visualización y el modelado, y herramientas para desarrollar aplicaciones web. Permite el modelado UML para diseñar bases de datos, con la posibilidad de representar la integración de los requisitos de datos y aplicaciones mediante diseños lógicos y físicos. Crea definiciones de tipos de documentos (DTD) XML para utilizarlas en la aplicación.

	Visual Paradigm para UML	**Rational Software**	**Enterprise Architect**
Licencia	Privativa (se tiene)	Privativa (no se tiene)	Privativa (no se tiene)
Experiencia de Uso en la UCI	Muy usada	Muy usada	Poco usada
Dificultad de Uso	Fácil	Medio	Medio

Tabla 1. Comparación entre herramientas de modelado.

1.7. Herramientas de Gestión Empresarial

1.7.1. Oracle SOA Suite

Es un producto completo que permite la creación, implementación y administración de soluciones orientadas a servicios incluyendo la integración de sistemas informáticos, el desarrollo y la administración de aplicaciones orientadas a servicios, así como del servicio propio como unidad básica del proceso de negocio. Aprovecha las inversiones existentes al ser modular, abierto y extensible [23]. Provee herramientas de fácil uso, promueve la reutilización de los activos, impulsa la colaboración del negocio y del desarrollador, brinda rendimiento, confiabilidad, gobierno y seguridad.

Entre los principales beneficios de su uso se encuentran la capacidad en común, inclusión de un solo modelo de administración e implementación, herramientas consistentes, seguridad integral y administración de metadatos unificados. Sin embargo, se trata de una herramienta con licencia privativa, por lo que no puede ser usada en el desarrollo de aplicaciones dentro de Cuba.

1.7.2. Suite JBoss

JBoss es un proyecto de código abierto basado en Java Platform Enterprise Edition (J2EE), y está implementado completamente en Java. Esto trae como ventaja que se puede utilizar en cualquier sistema operativo que soporte una máquina virtual de java. Es capaz de implementar cualquiera de los servicios de J2EE y está licenciado bajo la Licencia Pública General Reducida de GNU (LGPL), donde se plantea que puede ser libremente usada para fines comerciales y ser redistribuido. Dadas estas facilidades JBoss se convirtió en la plataforma más popular para desarrolladores, productores independientes y grandes empresas.

Esta herramienta es confiable a altos niveles de desarrollo como pudiera ser el software de una gran empresa y está orientado a SOA, por lo cual es flexible. Es una herramienta que por su bajo costo de adquisición y soporte se ha convertido en una de las más utilizadas por los desarrolladores en países subdesarrollados como Cuba.

1.7.3. Suite WSO2

WSO2 es una compañía que se dedica al desarrollo de aplicaciones de código abierto bajo la licencia Apache y está basada en SOA.

WSO2 Carbon, permite a los desarrolladores de manera rápida y simple la orquestación de los procesos de negocio. Está basado en OSGi (de inglés Open Services Gateway Initiative) e incluye más de 175 componentes que gestionan la mensajería, los datos, el negocio, la presentación, el monitoreo y los servicios.

	Oracle SOA Suite	**Suite JBoss**	**Suite WSO2**
Licencia	Privativa	Libre	Libre
Acceso a Soporte	No	No	Sí
Experiencia de Uso en la UCI	Poco usada	Poco usada	Poco usada, excepto en el CDAE.
Dificultad de Uso	Fácil	Fácil	Fácil

Tabla 2. Comparación de herramientas de gestión empresarial.

1.8. Bus de servicios empresariales

El ESB es un componente de infraestructura basado en XML que proporciona un escenario para la orquestación de servicios. Se considera que es la columna vertebral de todo SOA robusto [4]. Integra

tanto los sistemas externos como los internos de la empresa. Provee una capa no direccional que permite que los servicios sean adicionados, actualizados o eliminados mientras que minimiza el impacto en las aplicaciones clientes. Esto es posible porque las aplicaciones clientes envían mensajes al ESB en lugar de comunicarse directamente con los servicios. El ESB enruta los mensajes a los servicios apropiados en concordancia a reglas previamente definidas [24].

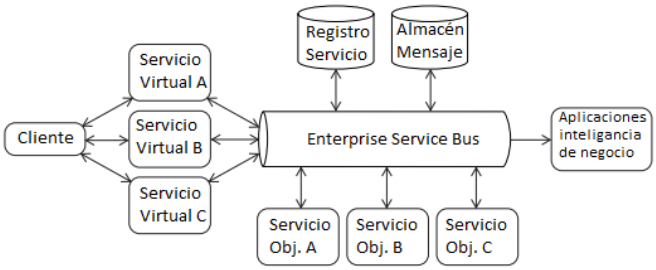


Figura 2. Funcionamiento ESB [24].

Muchas veces los consumidores de los servicios que se comunican con el ESB lo hacen mediante un conjunto de mensajes estándares, que no son más que un modelo de datos canónicos. La ventaja de esto es que no se necesitará crear servicios especializados diseñados para procesar todas las demandas específicas de las aplicaciones cliente. Ya que todos los clientes son forzados a usar la forma canónica, se crea un solo servicio que procese todas las peticiones [24].

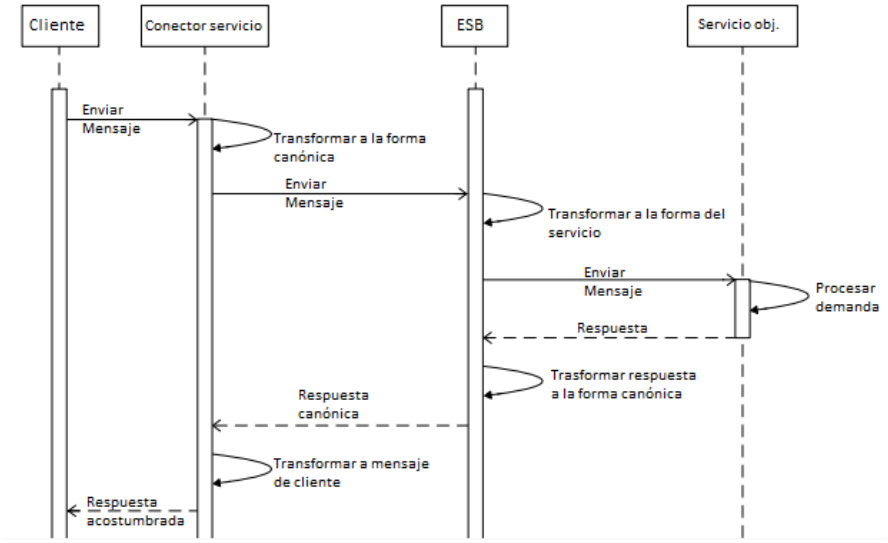


Figura 3. Funcionamiento del MDC en un ESB [24].

Como se describe en la Figura 3 un modelo de datos canónicos en un ESB transforma el mensaje enviado por el cliente a la forma canónica antes de que llegue al ESB, donde se transforma a la forma del servicio de manera que este pueda entender la demanda, procesarla y enviar la respuesta correspondiente. Esta respuesta es transformada en el ESB a la forma canónica. Luego, en el conector de servicio se transforma a la forma del cliente y este último recibe la respuesta en el formato propio.

Otra función realizada por el ESB es la traducción de protocolo y el mapeo de transporte. Esto puede emplearse cuando el servicio usa un protocolo o trasporte que es ajeno al del cliente. Los ESB sofisticados tienen otras funciones como la garantía de entrega de mensajes y autentificación [24].

Se listan a continuación las principales características y funciones de un ESB [2]:

- Direccionamiento y ruteo.
- Estilos sincrónicos y asincrónicos de comunicación.
- Transporte múltiple y encuadernaciones protocolares.
- Traducción y transformación del contenido.
- Orquestación de procesos de negocio.
- Adaptación a múltiples plataformas.
- Procesamiento de eventos.
- Integración de diseño, implementación y herramientas de despliegue.
- Gestión y monitoreo.
- Seguridad y persistencia.

Un ESB es una plataforma de integración basada en estándares que combina mensajes, servicios web, transformación de datos y enrutamiento inteligente para de forma fiable conectar y coordinar la interacción de un número significativo de aplicaciones a través de las empresas extendidas con integridad transaccional [25].

1.8.1. WSO2 ESB

Forma parte de la Suite WSO2 Carbon y ofrece una fácil configuración a través de una amigable interfaz gráfica. Permite el desarrollo de servicios proxy a fin de lograr la transparencia de ubicaciones de los servicios expuestos, además incluye transformación y ruteo de mensajes, monitoreo de las comunicaciones y gestiona la seguridad y las capacidades del gobierno SOA.

WSO2 ESB establece un bajo acoplamiento de los servicios, conectando sistemas de una manera gestionada y visualizada que permite a los administradores controlar y dirigir la comunicación sin interrumpir las aplicaciones existentes [18].

Con WSO2 ESB se puede:

- Desarrollar una potente lógica ESB.
- Gestionar las interacciones de los servicios.
- Crear servicios proxy para alojamiento virtual.
- Aplicar fácilmente seguridad a los mensajes.

WSO2 ESB es también compatible con las implementaciones del clúster, así como con el soporte para equilibrar la carga y el almacenamiento en caché del clúster. La integración con un registro de gobierno WSO2 integrado o un registro/repositorio externo, permite a WSO2 ESB usar recursos externos definidos por mediación, así como guardar esta configuración en un registro/repositorio externo a fin de lograr un mejor gobierno SOA.

1.8.2. JBoss ESB

El ESB de JBoss permite distribuir las instancias del servicio a través de muchos nodos, que pudieran ser máquinas virtuales o físicas que ejecutan instancias de JBoss ESB, además no pone restricciones en las estructuras de los servicios lo que permite a las implementaciones cambiar sin que los clientes o usuarios cambien [21].

El principal objetivo del ESB de JBoss es permitir comunicación entre sistemas dispares. Para lograrlo abstrae las diferencias entre los sistemas y trata a cada uno de ellos como un servicio lógico del ESB. JBoss ESB no provee una arquitectura orientada a servicios pero si da las herramientas para construir una pues provee bajo acoplamiento y mensajería asíncrona [26].

Entre sus características se encuentran que permite el monitoreo de procesos de negocio, entorno de desarrollo integrado, interfaz humana de flujo de trabajo del usuario, gestión de procesos de negocio, conectores, administrador de transacciones, seguridad, contenedor de aplicaciones, servicio de mensajería, repositorio de metadatos, nombrado servicio y arquitectura de computación distribuida [21].

1.8.3. Oracle ESB

El ESB de Oracle es un framework de aplicaciones de bajo acoplamiento que ofrece mayor flexibilidad, reusabilidad y capacidad de respuesta global en un entorno distribuido, heterogéneo y orientado a mensajes bajo los estándares de la industria que forma parte de la Suite SOA de Oracle [17].

Es un componente que provee la integración de aplicaciones en cuanto a conectividad, transformación de documentos, enrutamiento basado en contenido y encabezado [22].

Usa estándares abiertos para conectar, transformar y enrutar los documentos de negocio como mensajes XML a través de las aplicaciones involucradas. Oracle ESB ofrece soluciones de tipo SOA y EDA [22].

	Oracle ESB	**JBoss ESB**	**WSO2 ESB**
Licencia	Privativa	Libre	Libre
Acceso a Soporte	No	No	Sí
Experiencia de Uso en la UCI	Poco usada	Poco usada	Poco usada, excepto en el CDAE.
Dificultad de Uso	Fácil	Fácil	Fácil

Tabla 3. Comparación entre los ESB.

1.9. Lenguajes de desarrollo

1.9.1. XSLT

XSLT (del inglés Extensible Style sheet Language for Transformation) es un lenguaje estándar que permite poner estilos a un XML o convertirlo en otro. Se encarga de que cada nivel de cada pieza de datos XML sea accesible y reusable a través de las plataformas y del tiempo. Evita lo obsoleto en cuanto a datos o a diseño de arquitectura de información. Es fácil de usar, de entender y de enseñar. XSLT permite convertir de XML a HTML, otros XML o simplemente texto plano. Provee rápidas y fáciles soluciones para toda transformación XML.

“XSLT es un lenguaje que provee el mecanismo para manipular transformar los datos XML, los que están basados en el estándar de W3C (del inglés World Wide Web Consortium) y son la base para el intercambio de información estándar. XML provee de estructura a la información, y XSLT, junto con otro estándar relacionado, XPath (del inglés XML Path language), proveen de significado a la extracción, reestructuración y manipulación de la información.” [27]

1.9.2. XPath

Fue creado para ser usado bajo es estándar XSLT y es un lenguaje que permite crear expresiones regulares que recorren documentos XML teniendo en cuenta su estructura jerárquica. Con XPath podemos hacer referencia a cualquier elemento dentro de un documento XML, ya sea texto, atributos, o cualquier otra información que se encuentre dentro del fichero. Con este propósito se proveen facilidades para el manejo de valores numéricos, cadenas y booleanos.

"XPath utiliza una sintaxis compacta y no-XML para facilitar el uso de XPath dentro de URIs (del ingles Uniform Resource Identifier) y de valores de atributos XML. XPath opera sobre la estructura lógica abstracta de un documento XML, más que en su sintaxis superficial. XPath obtiene su denominación por el uso que hace de una notación de caminos, como en los URLs (de inglés Uniform Resource Locator), para navegar a través de la estructura jerárquica de un documento XML." [28]

XPath se basa en expresiones y la construcción de estas se fundamenta en la evaluación a fin de obtener un objeto que puede ser un nodo, una cadena, un numero o un valor booleano.

1.9.3. XQuery

Se trata de un lenguaje de consulta que trata documentos XML como si se tratase de una base de datos en sí. Esta semánticamente basado en SQL (del inglés Structured Query Language) pero maneja también algunas funciones de programación.

XQuery se encarga de proporcionar los medios para extraer y manipular información desde cualquier fichero XML o cualquier otra fuente de datos que se pueda representar mediante XML, pudieran ser bases de datos relacionales. XQuery usa expresiones basadas en XPath para acceder a los elementos del fichero aunque también incluye algunas expresiones SQL como son FOR, LET, WHERE, ORDER BY y RETURN. Por esta razón a estas expresiones se les conoce como FLWOR.

Permite también la construcción de documentos XML mediante el resultado de las consultas. Se puede utilizar una sintaxis XML si se conoce previamente la estructura, en caso contrario se hace mediante la construcción dinámica de nodos. Para ello los constructores se pueden definir dentro del lenguaje y ser anidados arbitrariamente según la conveniencia.

Es un lenguaje útil para desarrollar servicios web dado que extrae de las base de datos la información necesaria. Es capaz también de seleccionar y transformar datos XML a XHTML (del inglés eXtensible Hyper Text Markup Language) y de esta manera se puedan publicar en la web. Obtiene datos de diferentes fuentes a fin de integrar la información.

Conclusiones Parciales

Luego de realizar un estudio del marco teórico conceptual sobre las principales herramientas, patrones, lenguajes asociados con el MDC se llega a la conclusión de que a pesar de que internacionalmente los modelos datos canónicos sean una solución que ya cuenta con experiencia, en Cuba no es del mismo modo, pues lejos de esto, los sistemas que se gestionan en el país presentan las dificultades de la ausencia de interoperabilidad y comunicación entre los sistemas.

En cuanto a las herramientas a utilizar en el desarrollo del MDC como herramienta de modelado se usará Visual Paradigm para UML, para la implementación del modelo se usará la suite WSO2 ya que brinda diferentes plataformas para el desarrollo exitoso de la solución, como WSO2 ESB.

Capítulo 2. Propuesta de Solución

Introducción

En el presente capítulo se describe la solución propuesta que consiste en la definición de una estrategia para el desarrollo de un MDC en un entorno empresarial. Para el diseño de la solución se toman como referencia los conceptos y términos vistos durante el Capítulo 1. Se describirán las herramientas, actividades y tecnologías que forman parte de la estrategia de desarrollo para la solución propuesta.

2.1. Concepción de la estrategia para el desarrollo de un MDC

La estrategia para el desarrollo de un MDC que se presenta consta de actividades como estructura principal, además de descripción, roles y premisas.

Figura 4. Estructura de la Estrategia para el Desarrollo de un MDC

2.1.1. Descripción

La estrategia para el desarrollo de un MDC se ocupa de la creación de este, mas no de su gestión. Si se quisiera adicionar nueva información al modelo, así como para modificar o eliminar la que ya existe, se debe pasar por un proceso de gestión de cambio que es transversal al desarrollo del MDC y se mantiene después de su concepción para darle mantenimiento y gestión al MDC durante toda su vida útil. Para esto existen herramientas en el mercado que facilitan la gestión y soporte de un MDC en funcionamiento como por ejemplo IgniteXML.

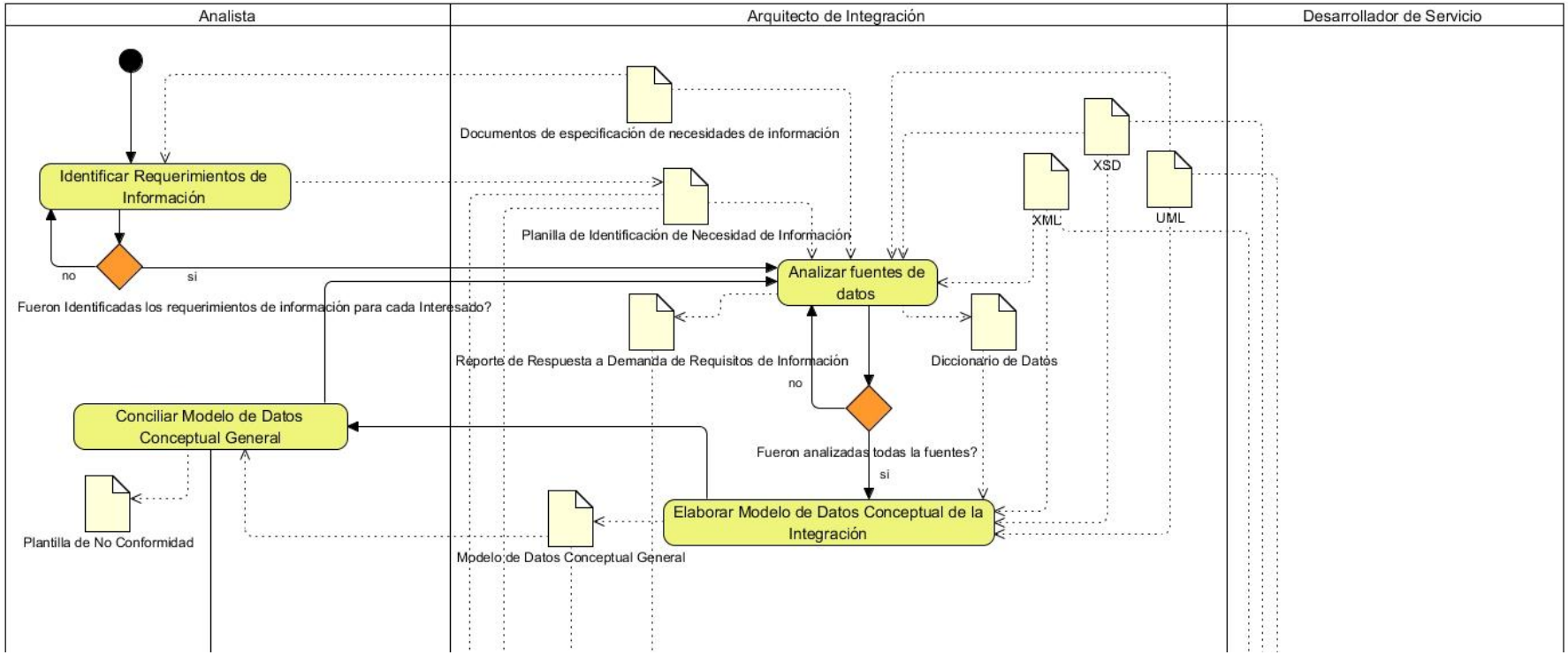


Figura 5. Flujo de actividades que describe la estrategia para el desarrollo de un MDC (parte 1)

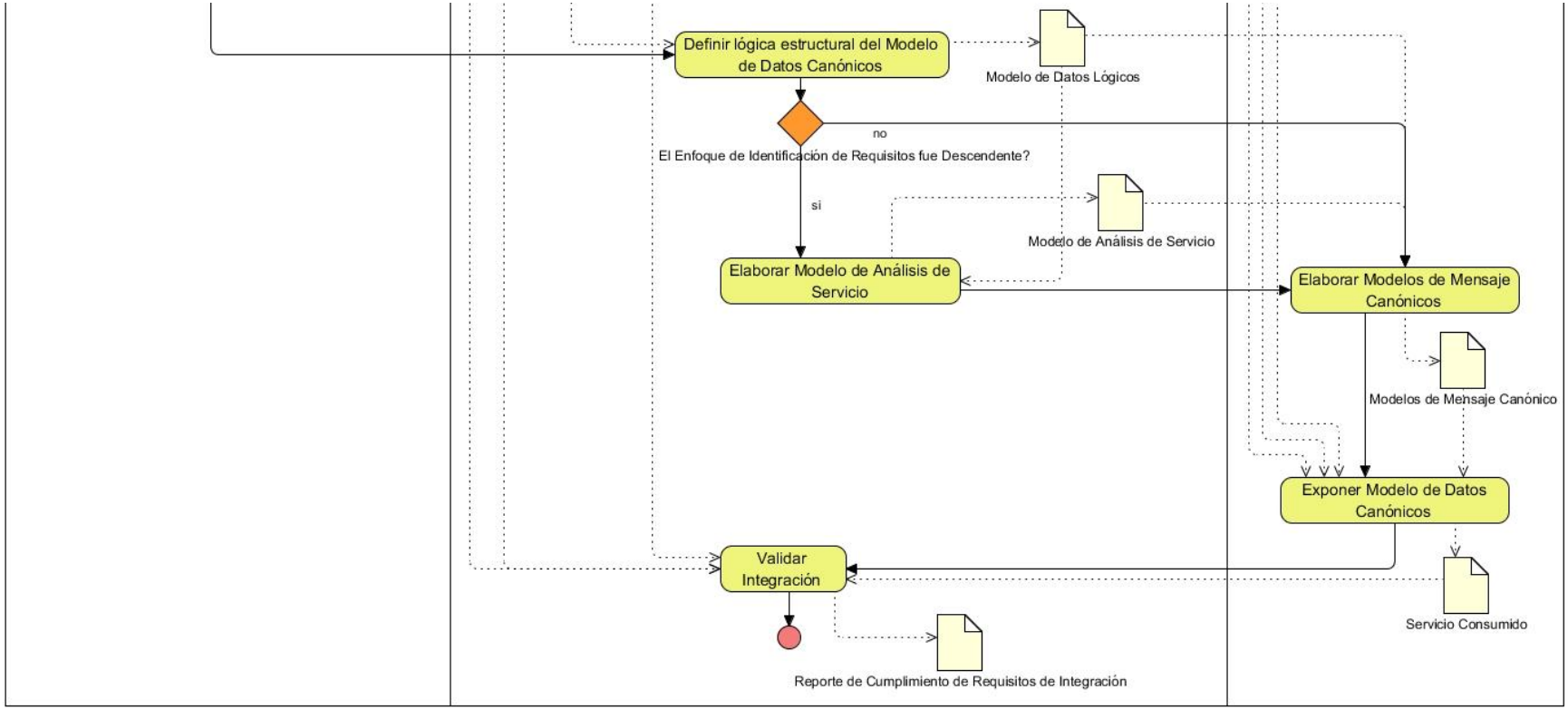


Figura 6. Flujo de actividades que describe la estrategia para el desarrollo de un MDC (parte 2)

La estrategia para el desarrollo de un MDC que se propone, ver Figura 5 y Figura 6, es aplicable a cualquier empresa que tenga necesidad de integración entre varias de sus aplicaciones. Por esta razón se tuvo en cuenta que no todas las organizaciones tienen las mismas condiciones tecnológicas para la aplicación de un MDC y se crearon artefactos simples y genéricos que reflejan las necesidades de información, las no conformidades, entre otros aspectos importantes para el desarrollo del MDC. La ejecución de las actividades depende de las condiciones organizacionales y de ahí se obtienen los resultados que produce cada una.

2.1.2. Roles

Un rol representa a una persona o grupo de ellas con responsabilidades comunes y nivel de competencia. Una persona puede desempeñar diversos roles, así como un mismo rol puede ser representado por varias personas. Sus responsabilidades abarcan tanto el llevar a cabo un conjunto de actividades como responder por la elaboración de un conjunto de artefactos. Además describen cómo los individuos deben comportarse en el contexto de un proyecto. Los principales roles que intervienen en la estrategia para el desarrollo de un MDC son:

- **Interesado:** Provee de la información básica para emprender el desarrollo: requisitos de información. Como premisa debe estar convencido de la necesidad de desarrollar el MDC. El interesado puede ser tanto un administrativo de la organización con especial interés y compromiso con el desarrollo de iniciativas de integración en la organización. Nótese que el interesado puede ser parte de alguna estructura o sistema que se desea integrar también.
- **Analista [29]:** Tiene a su cargo el entendimiento de las necesidades y oportunidades dentro del negocio, son traductores de lo técnico a lo “humano” y viceversa, sirviendo como puente entre el equipo de desarrollo y el Interesado. En dependencia del nivel de conocimiento que tenga el Analista puede ser Sénior o Junior.
- **Arquitecto de Integración [30]:** Se trata de la persona o grupo de ellas que elabora el MDC a partir de la solicitud del Interesado. Se responsabiliza por el buen funcionamiento de la integración y el intercambio de los datos demandados.
- **Desarrollador de Servicios:** Es el encargado de realizar la implementación de orquestación de servicios.

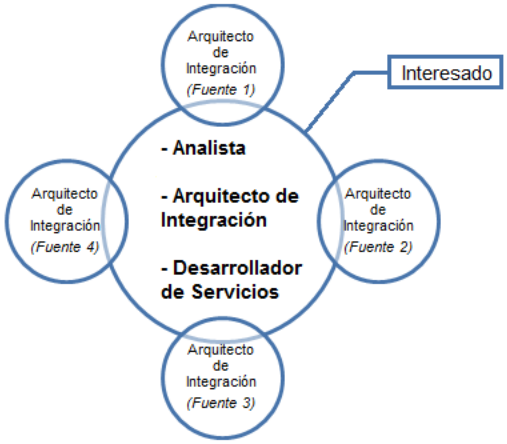


Figura 7. Relación entre roles.

En el esquema anterior se representa la relación que existen entre los roles. El Analista, el Arquitecto de Integración y el Desarrollador de Servicios intervienen directamente en el desarrollo del MDC. Puede haber Arquitectos de Integración comportándose de diferente forma:

- Del lado de la fuente de datos, los que responden por la información demandada que se encuentra en el sistema fuente que gestiona; existe al menos uno para cada sistema fuente de datos en el entorno de integración.
- Del lado del MDC desarrollando la solución sin un interés propio en la integración, sin necesidades propias de información, garantizando el cumplimento de las necesidades del Interesado.

Responsabilidad por Roles
Interesado
Define necesidades de información, sean de tipo gerencial (una persona necesita la información de una fuente) o a partir de una fuente (una aplicación necesita obtener información de otra).
Describe las especificaciones de la necesidad de información (fuente de datos, medio de la comunicación, fuente de información, justificación de la necesidad de información, estructura en la que necesita la información, medio actual por el que recibe la información, etc.).
Analista
Define, en conjunto con el Interesado, los requisitos de integración, basándose en las necesidades de información.
Modela y describe los procesos de negocio del Interesado y/o las características de su sistema.
Identifica necesidades de información a partir de los enfoques Ascendente y Descendente.
Arquitecto de Integración

Selecciona, gestiona y define soluciones de los escenarios arquitectónicos de integración.
Define y nominaliza los patrones y soluciones genéricas de integración, normaliza la comunicación, estableciendo cómo se va a llevar a cabo la interacción entre componentes.
Establece, norma y justifica cada uno de los elementos de integración.
Identifica los flujos de proceso existentes en la arquitectura de negocio, realiza la conversión de trazabilidad a la arquitectura del sistema, diseña y administra las soluciones de integración de procesos.
Identifica formatos y estándares que se aplicarán para los procesos de interoperabilidad e integración.
Formaliza los recursos de integración, los nodos de integración, los flujos de trabajos estáticos y dinámicos y formaliza el grupo de acciones dinámicas controladas.
Colabora en la comprensión de requerimientos funcionales y de calidad para las aplicaciones integradas.
Selecciona tecnologías de integración adecuadas que cumplan con los requerimientos de la aplicación
Analiza y evalúa la inversión o desarrollo de soluciones base tecnológica de integración, identifica las necesidades tecnológicas de la integración.
Analiza y diseña la solución de integración.
Desarrollador de Servicio
Implementa los servicios que consumen de las aplicaciones fuentes de información.
Realiza transformaciones de mensajes mediante la aplicación de patrones de integración empresarial.
Asegura que los servicios están ajustados a los estándares y buenas prácticas.
Estandariza los datos provenientes de las fuentes en caso de no ser entregadas en formato estándar de datos.
Garantiza la funcionalidad de las transformaciones mediante pruebas internas al código y al servicio consumido que se obtiene tras la exposición del MDC en el escenario de integración.

Tabla 4. Responsabilidades de los roles definidos para la Estrategia propuesta

2.1.3. Premisas

Las premisas de la estrategia para el desarrollo de un MDC definen elementos claves que se deben tener en cuenta a la hora de aplicarla. Principalmente establecen condiciones y/o restricciones de desarrollo que pueden condicionar la ejecución de la misma, según cada caso.

Las premisas que se definen para la Estrategia para el Desarrollo de un MDC son:

P1. Sistemas Implicados: Deben existir al menos dos sistemas para que exista intercambio de información. No tiene sentido establecer comunicación en un único sistema. La existencia de dos o más sistemas garantiza que existan sistemas fuentes y destinos en el intercambio de información.

P2. Requisitos reales de información: Deben existir necesariamente requisitos reales de información. No debe comenzarse una iniciativa de integración mediante el desarrollo de un MDC si estos requisitos no están claramente definidos en la organización o área a integrar. A fin de ahorrar en esfuerzos estos requisitos deben definirse previamente, por lo que forma parte de la propuesta de estrategia. Si no existiese un interesado real deberá entonces valorarse el desarrollo o no del MDC.

P3. Compromiso administrativo: Debe existir interés administrativo de todas las partes involucradas en el intercambio de información. Premisa altamente relacionada con la P2: Requisitos reales de información. La estrategia propuesta supone el interés o postura organizacional que favorezca el desarrollo del modelo, favoreciendo y facilitando el desarrollo del mismo, dígase la colaboración de todos los roles implicados en el desarrollo, lo cual permita el intercambio y acceso a la información oportunamente. Si no existe un claro interés para lograr la integración, tanto en la alta gerencia de la empresa como en niveles inferiores, es altamente probable que la iniciativa de integración fracase. De ahí que los Arquitectos de Integración internos de cada aplicación o fuente de datos y demás interesados deben estar de acuerdo con la necesidad de integración, para que de esta manera formen parte del proyecto de integración y se comprometan con el éxito común del mismo.

P4. No necesidad de adopción de SOA: El desarrollo de un MDC no precisa necesariamente de la adopción de SOA. El MDC no modifica las aplicaciones coexistentes en el entorno de integración sino que solamente las emplea para la captura de información relevante para desarrollar el modelo, por lo que la adopción de SOA sería un empeño a mayor plazo, que podría incluir o no el desarrollo de un MDC.

P5. Formato estándar de datos fuentes: La información que proveen las fuentes debe ser suministrada con formato estándar de datos (XML, XSD, etc.) que facilite tanto el intercambio como el manejo de los mismos. En el caso que la información sea suministrada mediante documentos o formatos no estándares de datos, el Desarrollador de Servicio será el responsable de extraer o transformar los datos hacia formatos estandarizados.

P6. Responsabilidad de la fuente: Los sistemas fuentes de datos son los responsables de la integridad, confiabilidad y disponibilidad de los datos brindados, así como su actualización. La estrategia propuesta se centra principalmente en el desarrollo del MDC y no así en validar los datos suministrados por cada fuente. La información y datos que se manejen e intercambien en todo caso

se realizará por canales seguros y previamente establecidos por la fuente y el Arquitecto de integración.

P7. Caducidad del proyecto de integración: Es importante señalar que la iniciativa de integración, en particular el desarrollo del MDC es un esfuerzo finito, que conlleva darle solución a cierto número de demandas o requisitos de información que solicita la organización o los interesados en la integración, de ahí que, como un proyecto de desarrollo de software tradicional, deba iniciarse y cerrarse una vez satisfechos estos requisitos. Se recomienda que los proyectos de integración no duren un largo período de tiempo pues es probable que se pierda el interés de las partes involucradas sin antes ver una solución de integración, o que los requisitos varíen con el tiempo y el proyecto entre en un ciclo en el que los requisitos nunca serán satisfechos.

P8. Responsables de la integración: El Arquitecto de Integración en conjunto con los interesados son los responsables de que se lleve a cabo exitosamente la integración y en caso contrario de la emisión de las no conformidades. Una vez que se acepta el proyecto de integración, el interesado o arquitecto de integración interno de cada fuente, acepta los requisitos a satisfacer, de ahí el Arquitecto de Integración está en la obligación de satisfacer dichas necesidades de información en el tiempo acordado y con la calidad que se perciba luego de los respectivos análisis de la información disponible.

2.1.4. Actividades

Las actividades constituyen el bloque principal de la estrategia para el desarrollo de un MDC, a través de las cuales se describen y especifican los pasos para adoptarla. Son unidades atómicas de trabajo que son realizadas por un rol. Por lo general requieren artefactos de entrada que contienen información útil para realizar dicho trabajo y poder obtener el resultado o artefacto de salida a la actividad. En su desarrollo se pueden utilizar o no herramientas y/o técnicas. Las actividades responden a un flujo de trabajo que guía la elaboración del MDC.

Cada actividad especifica:

- **Rol:** en este contexto representan un conjunto de responsabilidades que debe poseer un individuo o conjunto de individuos para realizar una actividad.
- **Descripción:** especifica la actividad, dando detalles de su realización y cumplimiento.

- **Entradas:** constituye un producto de trabajo tangible (documentos, modelos UML, etc.), que necesita una actividad para su ejecución.
- **Herramientas y Técnicas:** las primeras representan herramientas de software que sirven de apoyo al rol en la realización de una actividad y las segundas se constituyen mediante procedimientos predefinidos, enfoques que pueden ser utilizados para perfeccionar el desarrollo de la actividad.
- **Salidas:** constituye un producto de trabajo tangible (documentos, modelos UML, etc.), que la resolución de una actividad puede generar y que es necesitada por otra actividad para su ejecución.
- **Flujo:** Secuencia de pasos que especifican gráficamente el desarrollo de la actividad.

A continuación se describen las actividades propuestas:

2.4.1.1. Identificar Requerimientos de Información

♀ **Rol:** Analista

≡ **Descripción:** El interesado necesita obtener información que es gestionada en un sistema que es ajeno a él. El Analista, con su ayuda hace una solicitud formal de la integración identificando la necesidad real de información y especificando que datos se necesitan integrar. Esta solicitud se lleva a cabo mediante la elaboración de la Planilla de Identificación de Necesidad de Información, (ver Anexo 1). Esta actividad es desarrollada iterativamente por cada uno de los interesados en la integración y consiste un levantamiento de requisitos con la finalidad de obtener las necesidades de información del Interesado, que serán la forma de validar que fueron satisfechas al final del desarrollo del MDC.

→ **Entradas:**

- Documentos de especificación de necesidades de información previamente establecidos: Es posible que existan antecedentes de iniciativas de integración en la organización, fichas técnicas, especificaciones de requisitos de software u otros documentos técnicos donde se especifiquen requisitos de integración que aún estén en vigencia y no se hayan resuelto.

◘ **Herramientas y Técnicas:**

- Entrevista: es una técnica ampliamente usada para la recopilación de información, principalmente en las etapas tempranas del desarrollo cuándo aún no está claro que hay que

desarrollar. El Analista se reúne con el Interesado o Arquitecto de Integración de la entidad demandante a fin de extraer las necesidades de información del Interesado.

- Taller: los requisitos tienen a menudo implicaciones desconocidas para el Interesado y que a menudo no se descubren en las entrevistas o quedan incompletamente definidas durante la misma. Estas implicaciones pueden descubrirse realizando talleres facilitados por un Analista, donde los Interesados participan en discusiones para descubrir requisitos, analizan sus detalles y las implicaciones. Para esta actividad en específico este tipo de técnicas resultan útiles pues se crea un debate entre varios de los implicados en la integración y en el caso que la información o conocimiento informático de los involucrados sea escasa, se formulan en conjunto los requisitos de integración.
- Forma de contrato o Planilla de identificación de necesidades de integración: en lugar de una entrevista, se pueden llenar formularios o contratos indicando los requisitos directamente.

← **Salidas:**

- Planilla de Identificación de Necesidad de Información: en la que queda reflejada la necesidad real de información, así como la información demandada por el interesado y las fuentes de información de las que requiere los datos, en caso de conocerlas.

↓ **Flujo:**

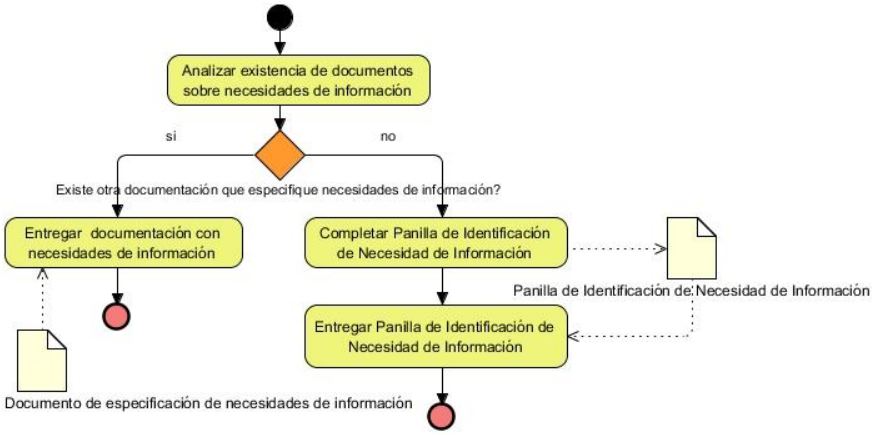


Figura 8. Actividad: Identificar de Requerimientos de Información

2.1.4.2. Analizar fuentes de datos.

♀ **Rol:** Arquitecto de Integración

≡ **Descripción:** El Arquitecto de Integración recibe la Planilla de Identificación de Necesidad de Información y basándose en esta analiza las necesidades de información del interesado refiriéndose a los sistemas que proveen la información demandada. Esta actividad se hace para cada sistema fuente de información. El Arquitecto de Integración debe contactar con los Arquitectos de Integración de cada aplicación a fin de solicitar la especificación de los datos que solicitó el interesado. La especificación debe ser entregada preferentemente en un formato estándar (XML, XSD, UML), en caso contrario el Desarrollador de Servicios transformará, en actividades posteriores, estos datos a un formato estándar usando patrones como el de Transformación de Formato de Datos propuesto por Thomas Erl, de forma que se garantice en futuras actividades el entendimiento de los sistemas. Esta transformación puede hacerse paralela al desarrollo siempre que sea antes de la actividad Exponer el MDC, donde serán necesitados los datos en formato estándar. En este caso se puede hacer una lectura del documento no estandarizado, extrayendo del mismo los requisitos de información. El Arquitecto de Integración es el encargado de elaborar de manera opcional el Diccionario de Datos (Anexo 2), documento que agrupa las especificaciones de los datos demandados y el Reporte de Respuesta a Demanda de Requisitos de Información (Anexo 3) que refleja las especificaciones de la disponibilidad de la información demandada y luego es enviada al Interesado para que refleje su conformidad. En ocasiones los sistemas fuentes cuentan con modelos conceptuales de sus activos de información, sería entonces beneficioso si el arquitecto pudiera adquirirlos, evitando tener que hacer el diccionario de datos. El objetivo de esta actividad es obtener los formatos y otras especificaciones de los datos de los sistemas fuentes y de esta manera definir el estado en el que se encuentra la información que necesita el interesado y principalmente, a partir de estas especificaciones desarrollar el MDC.

→ **Entradas:**

- Esquemas XML y XSD; Modelos UML que pueden ser provistos por los sistemas fuentes de los que se obtendrán las especificaciones de los datos demandados con el objetivo de ser usados en actividades posteriores.
- Planilla de Identificación de Necesidad de Información: en este caso se toman de ella específicamente las fuentes de datos que el interesado expone, en caso de conocerlas y los datos demandados. De no conocerse los sistemas fuentes que proveen la información, el Arquitecto de

Integración es responsable de hacer un estudio sobre cuáles sistemas pueden satisfacer estas necesidades de información para poder llevar a cabo la presente actividad.

- Documentos de especificación de necesidades de información previamente establecidos.

◘ **Herramientas y Técnicas:**

- Entrevista, para la obtención de la información referente a las fuentes de los datos demandados por el interesado.
- Herramienta CASE para el análisis de los documentos XML, XSD, y UML.

← **Salidas:**

- Reporte de Respuesta a Demanda de Requisitos de Información, en el que se establecerá detalladamente el porciento de cumplimiento de las demandas de información del interesado.
- Diccionario de datos que puede ser elaborado en caso que no se puedan brindar las especificaciones de la información demandada en un formato de datos estándar. Reflejará las principales características de los datos que son útiles en actividades posteriores.

↓ **Flujo:**

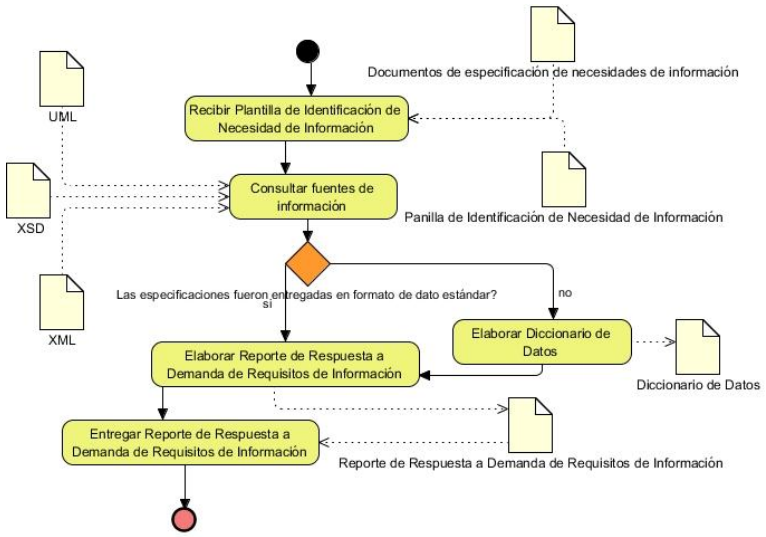


Figura 9. Actividad: Analizar fuentes de datos

2.1.4.3. Elaborar Modelo de Datos Conceptual de la Integración

♀ **Rol:** Arquitecto de Integración

≡ **Descripción:** A partir de los elementos agrupados en el Diccionario de Datos y esquemas XML, XSD y los modelos UML, el Arquitecto de Integración define el formato estándar de los datos que

serán compartidos. Para ello elabora el Modelo de Datos Conceptual General que agrupa los datos que se compartirán de manera gráfica. El objetivo de esta actividad es visualizar en un lenguaje gráfico y estándar, como UML, los formatos definidos que tendrán los datos. Este modelo es la guía del desarrollo ya que será utilizado en posteriores actividades para definir aspectos de la lógica y el diseño del MDC.

Para la definición del Modelo de Datos Conceptual General se deben incluir solo los datos que incurren en las necesidades de información, si se incluyen menos, no se satisfarán las necesidades de información del interesado; si se incluyen más, el MDC sería innecesariamente grande. Se reflejarán en este modelo todas las entidades y los atributos que constituyen necesidades de información, sin reflejar aspectos de la lógica o el diseño, solo las relaciones que existen entre ellos.

→ **Entradas:** Esquemas XML, XSD, Modelos UML, Diccionario de Datos: de los que se obtienen las especificaciones de los datos que ofrecen las fuentes.

◘ **Herramientas y Técnicas:** Herramienta CASE, se propone Visual Paradigm para UML

← **Salidas:** Modelo de Datos Conceptual General; refleja el formato estándar definido con el que se llevará a cabo la comunicación entre sistemas, así como el intercambio de información en un lenguaje estándar y gráfico, constituye la primera vista del MDC.

↓ **Flujo:**

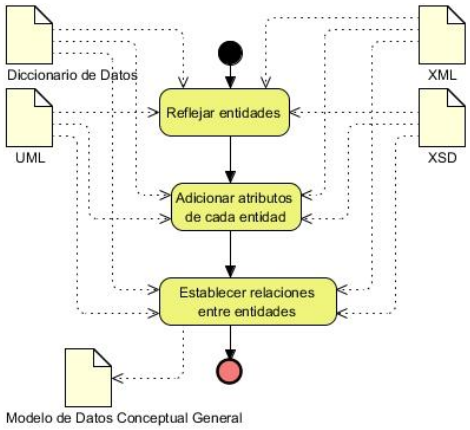


Figura 10. Flujo de actividades para Elaborar Modelo de Datos Conceptual de la Integración

2.1.4.4. Conciliar Modelo de Datos Conceptual General

♀ **Rol:** Analista

≡ **Descripción:** El Analista muestra el Modelo de Datos Conceptual General al Interesado y es revisado a fin de ver si fueron cumplidas sus necesidades de información hasta este punto del desarrollo. Si no está conforme el Analista, a partir de las inquietudes del Interesado llena la Planilla de No Conformidad (ver Anexo 4) y se la entrega al Arquitecto de Integración que intentará solucionar la inconformidad del Interesado. Este consenso debe hacerse personalmente, de forma que el Interesado exponga al Analista sus inquietudes y este último pueda definir cuáles pueden o no ser atendidas. Esta actividad tiene como objetivo afianzar el Modelo de Datos Conceptual General con el Interesado de forma que se puedan rectificar a tiempo errores o actualizar las necesidades de información del interesado sin afectar etapas posteriores del desarrollo. Es importante que el Interesado tenga presente que esta es la última vez que se le consulta sobre los requisitos de información, por lo tanto debe tener bien definido lo que necesita. Si cambiasen sus necesidades de información luego de haber conciliado el Modelo de Datos Conceptual General, esto se gestionaría en un proceso de gestión del cambio (que no se incluye en esta estrategia).

→ **Entradas:** Modelo de Datos Conceptual General.

◘ **Herramientas y Técnicas:** Entrevista, Taller.

← **Salidas:** Planilla de No Conformidad, en la que se refleja cuáles fueron los aspectos con los que el interesado no estuvo de acuerdo, de forma que se rectifiquen a tiempo, antes de avanzar más en el desarrollo.

↓ **Flujo:**

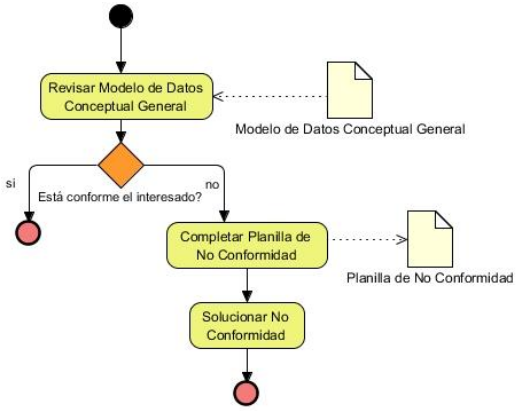


Figura 11. Flujo de actividades para Conciliar Modelo de Datos Conceptual General

2.1.4.5. Definir lógica estructural del Modelo de Datos Canónicos

♀ **Rol:** Arquitecto de Integración

≡ **Descripción:** Recibiendo el Modelo de Datos Conceptual General el Arquitecto de Integración define la lógica de la estructura del MDC en el Modelo de Datos Lógicos. Para ello define llaves primarias, foráneas, cardinalidad, restricciones en los datos y normaliza el modelo; obtiene una vista más detallada de las especificaciones de los datos. Esta actividad tiene como finalidad establecer los aspectos lógicos del MDC, establecer una vista clara de las relaciones entre entidades y los atributos que las identifican.

→ **Entradas:** Modelo de Datos Conceptual General, del que se toma la estandarización de datos definida.

◘ **Herramientas y Técnicas:** Herramienta CASE, se recomienda Visual Paradigm para UML

← **Salidas:** Modelo de Datos Lógicos, en el que se describe de forma gráfica la lógica del MDC, estableciendo las especificaciones de las relaciones entre entidades, así como llaves primarias y otros aspectos lógicos.

↓ **Flujo:**

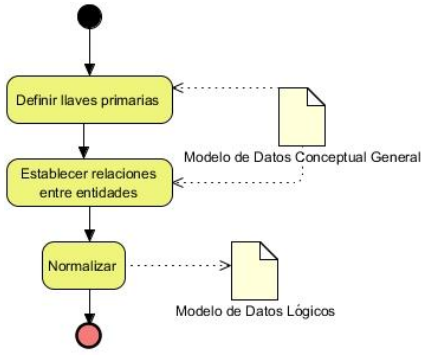


Figura 12. Flujo de actividades para Definir lógica estructural del Modelo de Datos Canónicos

2.1.4.6. Elaborar Modelo de Análisis de Servicio

♀ **Rol:** Arquitecto de Integración

≡ **Descripción:** Si el levantamiento de requerimientos de información se realiza mediante el enfoque descendente, el Arquitecto de integración debe elaborar el Modelo de Análisis de Servicio a fin de describir componentes y alinear el desarrollo del MDC con la arquitectura global. Esta actividad trae como beneficio la alineación de la capa de acceso a datos, la de componentes y la de las definiciones de los servicios.

→ **Entradas:** Modelo de Datos Conceptual General, Modelo de Datos Lógicos; cualquiera de estos modelos puede ser utilizado para la elaboración del Modelo de Análisis de Servicio.

◘ **Herramientas y Técnicas:** Herramienta CASE, se recomienda Visual Paradigm para UML.

← **Salidas:** Modelo de Análisis de Servicios

↓ **Flujo:**

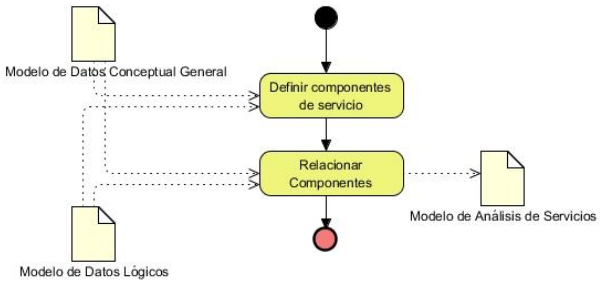


Figura 13. Flujo de Actividades para Elaborar Modelo de Análisis de Servicios

2.1.4.7. Elaborar Modelos de Mensaje Canónico

♀ **Rol:** Desarrollador de Servicio

≡ **Descripción:** A partir del Modelo de Datos Lógicos, se generan documentos XML que contienen las especificaciones de los datos solicitados de manera estándar, en la forma que tendrán los mensajes que se intercambiarán durante la comunicación en la forma canónica. Este esquema constituye pequeñas vistas no gráficas del MDC, las que pueden ser utilizadas en los esquemas WSDL que exponen los servicios [31]. Si se desarrolla mediante el enfoque descendente los Modelos de Mensaje Canónicos son elaborados basados en el Modelo de Análisis de Servicio. Esta actividad pretende definir pequeños extractos del MDC a fin de que sean utilizados directamente para la comunicación de los sistemas de la forma estándar definida mediante el MDC. Cada sistema fuente o destino tendrá su propio Modelo de Mensaje Canónico con las especificaciones de los datos que intercambia, en el formato estándar que propone el MDC.

→ **Entradas:**

- Modelo de Datos Lógicos, del que se obtendrán las formas estandarizadas, como base para la creación de los Modelos de Mensaje Canónico, respetando la lógica definida en el modelo.

- Modelo de Análisis de Servicio, se extraerán los componentes definidos y se convertirán es pequeños esquemas XML que respondan a estos componentes. Esta forma también respeta el formato propuesto por el MDC y apoya la transformación de los mensajes.

◘ **Herramientas y Técnicas:** Herramienta de Desarrollo, se recomienda XML Spy y/o Eclipse

← **Salidas:** Modelos de Mensaje Canónico, en los que se refleja la estructura de los mensajes en la comunicación entre el MDC y los sistemas fuentes y/o destino.

↓ **Flujo:**

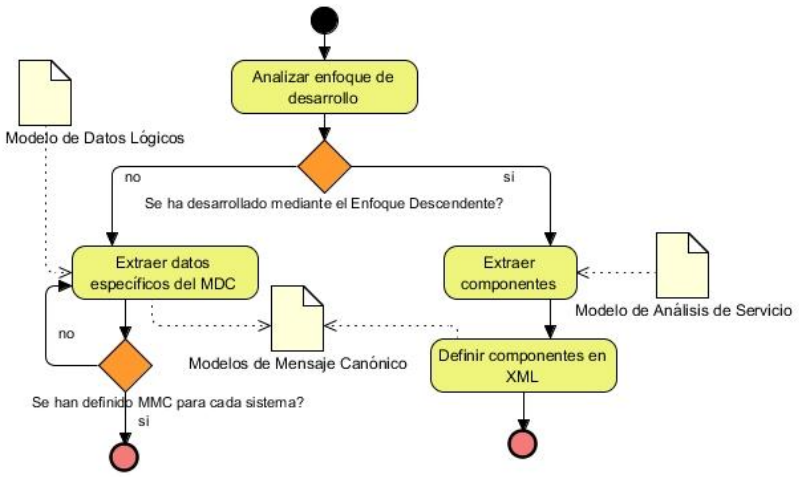


Figura 14. Flujo de actividades para Elaborar Modelos de Mensaje Canónicos

2.1.4.8. Exponer Modelo de Datos Canónicos

♀ **Rol:** Desarrollador de Servicio

≡ **Descripción:** Los esquemas XML obtenidos de la actividad anterior, son expuestos en el ESB. El objetivo de esta actividad es garantizar el intercambio de información entre los sistemas mediante la transformación y ruteo de mensajes en el ESB. Para ello se deben aplicar otros patrones de integración que se refieran a la transformación de los mensajes, de forma que sean traducidos de la forma canónica a la forma de los sistemas y viceversa. Es en esta actividad donde se implementan las transformaciones necesarias para garantizar el entendimiento entre los sistemas. Una vez garantizada la transformación, se consumen los servicios con el objetivo comprobar la efectividad de lo implementado en las trasformaciones y que los sistemas se comuniquen de manera fluida.

→ **Entradas:**

- Modelos de Mensaje Canónico, los que constituyen el medio de comunicación entre los sistemas y el MDC.
- Esquemas XML, XSD, UML, entregadas por los sistemas fuentes y resultados de la transformación de formatos, los que constituirán el destino y/o origen de las comunicaciones.

◘ **Herramientas y Técnicas:** Herramienta de Gestión Empresarial, se propone el ESB de la Suite WSO2.

← **Salidas:** Servicios Consumidos: los que contienen las especificaciones de las transformaciones de los mensajes.

↓ **Flujo:**

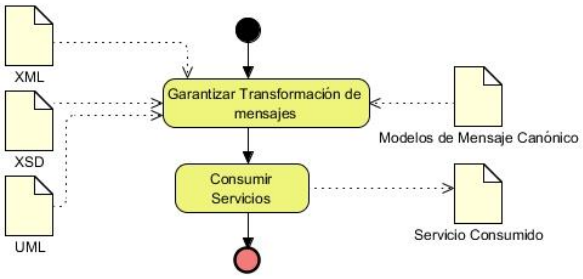


Figura 15. Flujo de actividades para Exponer MDC

2.1.4.9. Validar requisitos de integración

♀ **Rol:** Arquitecto de Integración

≡ **Descripción:** Los objetivos de esta actividad son: validar el cumplimiento y la satisfacción de las necesidades de información que plantea el Interesado en la primera actividad y probar la Integración mediante el MDC. Para el primero se validan los requerimientos de información planteados por el Interesado teniendo en cuenta el Reporte de Respuesta a Demanda de Requisitos de Información, el documento donde se reflejen las necesidades de información del Interesado y el servicio consumido tras la exposición de MDC en el ESB. Se realiza una comparación y se decide si se han cumplido todas las necesidades del Interesado, según los datos atendidos en el Reporte de Respuesta a Demanda de Requisitos de Información. Se genera entonces el Reporte de Cumplimiento de Requisitos de Integración, ver Anexo 5, en el que se refleja este análisis, así como la conformidad del Interesado, estableciendo el fin del proyecto. Para

probar los servicios que satisfacen los requisitos de integración se prueba internamente el MDC, de forma que se valide el buen funcionamiento del mismo.

→ **Entradas:**

- Reporte de Respuesta a Demanda de Requisitos de Información, del que se tomará el porciento de atención a las demandas de información del Interesado.
- Planilla de Identificación de Necesidad de Información, Documentos de especificación de necesidades de información previamente establecidos; de los que se tomarán las necesidades de información del Interesado a fin de compararlas con las satisfechas en el Servicio Consumido, teniendo en cuenta los datos sin atender descritos en el Reporte de Respuesta a Demanda de Requisitos de Información.
- Servicio Consumido, serán comparadas las necesidades satisfechas expuestas en el Servicio Consumido con las expuestas en la Planilla de Identificación de Necesidad de Información, teniendo en cuenta los datos sin atender descritos en el Reporte de Respuesta a Demanda de Requisitos de Información.
- Modelos de Mensaje Canónico, sufrirá pruebas internas a fin de validar el buen funcionamiento de la integración y se garantice la comunicación entre sistemas.

◘ **Herramientas y Técnicas:** Herramienta de Gestión Empresarial, se recomienda el ESB de la Suite WSO2 Carbon.

← **Salidas:** Reporte de Cumplimiento de Requisitos de Integración

↓ **Flujo:**

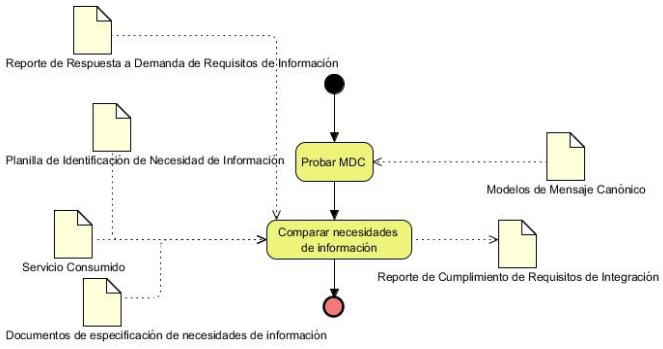


Figura 16. Flujo de actividades para Validar Requisitos de Integración

Conclusiones Parciales

En el presente capítulo se propuso la Estrategia para el Desarrollo de un MDC que cubre el ciclo completo de un proyecto de integración centrado en el desarrollo de un MDC partir de la descripción de roles, premisas a tener en cuenta para el desarrollo y actividades, especificando que rol las llevará a cabo, sus artefactos de entrada y salida, así como herramientas y técnicas para la ejecución de las mismas. Se propone además para cada actividad un flujo de actividades que especifica y describe el desarrollo de la actividad. Se especifica el flujo de los artefactos y proponiendo en algunos casos las herramientas para el desarrollo de cada actividad.

Capítulo 3. Validación y Aplicación Práctica de la Propuesta

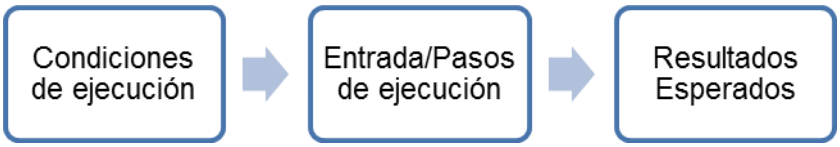
Introducción

En este capítulo se aplica y evalúa la solución propuesta mediante su aplicación práctica en un caso de estudio, comprobando la efectividad de las transformaciones de mensajes como método para la comunicación entre aplicaciones y mediante el Método Experto, contribuyendo este último en la calificación de aspectos cualitativos como la calidad de la estrategia, o sea, características como completitud, facilidad de implementación, correspondencia entre actividades, roles, artefactos y otras. El objetivo de esta validación es analizar si la estrategia propuesta contribuyó en la creación de un entorno de integración que favoreciera la interoperabilidad en la UCI.

3.1. Aplicación práctica de la propuesta mediante un caso de estudio

Los casos de estudio son útiles para probar soluciones con enfoque centrado tanto en el proceso como en el problema y contribuyen al perfeccionamiento de la solución para adaptarlas a la realidad lidiando con los factores inesperados que pueden incidir los resultados finales. Se puede decir entonces, que el objetivo de los casos de estudio es comprobar la solución en un proyecto real de la organización y capturar lecciones aprendidas y resultados para refinar la solución con vistas a su posterior instalación [32].

La validación mediante el caso de estudio será ejecutada teniendo en cuenta las siguientes etapas:



El caso de estudio que se elabora es un acercamiento a la integración en la UCI. Para ello no se cuenta con necesidades reales de integración debidamente especificadas, por lo que se asume su existencia mediante la obtención de los servicios del sistema Assets, encargado de la gestión de los trabajadores de la universidad, Personal, que gestiona las áreas tercerizadas dentro de la universidad y Akademos, que se encarga de la gestión académica, expuestos en la UDDI (del inglés Universal Description, Discovery and Integration) de la universidad disponibles en la dirección https://uddi.uci.cu. La ejecución del caso de estudio no es con un caso real de integración, pero es efectivo para la validación de la estrategia

propuesta ya que se trabaja en un escenario real en el que las aplicaciones intercambian información real entre ellas.

3.1.1. Condiciones de ejecución.

Teniendo en cuenta las premisas expuestas en el Capítulo 2, se puede decir que las condiciones de ejecución del presente caso de estudio se cumplen en las premisas: P1: Sistemas Implicados, P4: No necesidad de adopción de SOA, P5: Formato estándar de datos fuentes, P6: Responsabilidad de la fuente, P7: Caducidad del proyecto de integración, P8: Responsables de la integración. Las premisas P2: Requisitos reales de información y P3: Compromiso administrativo quedan sin cumplimiento, pues a pesar que se reconoce la necesidad de integración no existe un compromiso claro a nivel institucional donde se definan las responsabilidades y por lo tanto, no están especificadas las necesidades reales de integración.

3.1.2. Pasos de ejecución

En esta etapa de la elaboración del caso de estudio se da cumplimiento a la estrategia trazada para el desarrollo de un MDC. La primera actividad a ejecutar es la Identificación de Requerimientos de Información. El cumplimiento de esta actividad se ve afectado pues no se cuenta con un interesado real que solicite la integración, por lo que para este caso se completa la Planilla de Identificación de Necesidad de Información que contiene necesidades de integración irreales pero necesarias para la Identificación de Requisitos de Información en este caso de estudio y constituyen el punto de partida para el mismo.

En este caso el interesado pudiera ser cualquier persona ajena a los sistemas de Assets, Akademos y Personal que necesite información que ellos pueden ofrecer, como por ejemplo el rector de la universidad o cualquier directivo, o incluso un sistema que para su funcionamiento necesite de la información de estos servicios.

Una vez que completada la Planilla de Identificación de Necesidad de Información queda de la siguiente forma, nótese que las especificaciones expuestas son hipotéticas:

Solicitante			
Requerimiento del proceso que necesita la información		Se necesita la información para la complementación de procesos de negocio en los que se involucra el conocimiento de aspectos específicos de estudiantes y trabajadores.	
Funcionalidades			
Nombre de la funcionalidad	**Parámetros de**	**Parámetros de salida**	**Descripción**

	entrada		
ObtenerDatosDocentesEstudianteDadoSolapin	solapín	Nombres(string), apellidos(string), grupo(string) y estado(string)	La funcionalidad devuelve el nombre completo, estado y el grupo del estudiante con el número de solapín entrado.
ObtenerDatosNoDocentesDadoCI	ci	nombres(string), apellidos(string), solapin(string), genero(string), municipio(string), provincia(string)	La funcionalidad devuelve el nombre completo, el número de solapín, el género del estudiante, el municipio y la provincia en la que reside el estudiante con el carnet de identidad entrado.
ObtenerTrabajadorDadoCI	ci	Nombres(string), apellidos(string), ci(string), solapín(string), docente(string), genero(string), militante(string), cargo(string), area(string)	La funcionalidad devuelve el nombre completo, el carnet de identidad, solapín, si es docente, el género, si es militante, el cargo que desempeña y el área del trabajador con el carnet de identidad entrado.
ObtenerNombre_Cargo_AreaDadoSolapín	solapín	Nombres(string), apellidos(string), cargo(string), area(string)	La funcionalidad devuelve el nombre completo, el número de solapín, el cargo que desempeña y el área del trabajador con el número de solapín entrado
Sistema(s) fuente		Gestión Universitaria, Assets y Personal.	
Medio actual para la comunicación		Correo electrónico	
Limitantes en la obtención de la Información		El correo independientemente de que ha resuelto el problema temporalmente la información solicitada no siempre llega en el tiempo requerido. Además la disponibilidad de la información depende del personal autorizado al acceso a esta.	

Figura 17. Planilla de Identificación de Necesidad de Información completada por el interesado

Una vez que el Arquitecto de Integración cuenta con esta planilla puede pasar a la siguiente actividad que se basa en el análisis de las fuentes de la información solicitada por el interesado. Esta actividad genera los esquemas XML de los servicios de los sistemas fuentes y el Reporte de Respuesta a Demanda de Requisitos de Información que se elabora para entregárselo al interesado de forma que refleje los datos que presentan dificultades para su tratamiento, si se diera el caso, y la justificación.

Nótese que el interesado en este caso conoce los sistemas fuentes que contiene la información que necesita, por lo que el Arquitecto de Información consulta los sistemas fuentes en busca de características de la información demandada como son la disponibilidad y el formato. En caso contrario el Arquitecto de Integración debe investigar la disponibilidad de los datos solicitados por el interesado en los sistemas que existen en el entorno informático en busca de sistemas fuentes que puedan satisfacer las necesidades de información del interesado.

En este caso se entregan los datos demandados en formato WSDL, que no es más que un formato XML que se utiliza para describir servicios web. A continuación se exponen fragmentos de los documentos adquiridos de los sistemas fuentes, consultados de la UDDI de la universidad.

```
<message name="ObtenerEstudianteDadoCI">
      <part name="CI" type="xsd:string" />
</message>
<message name="ObtenerEstudianteDadoCIResponse">
      <part name="ObtenerEstudianteDadoCIReturn" type="xsd:anyType" />
</message>
<message name="ObtenerEstudianteDadoIdExpediente">
      <part name="IdExpediente" type="xsd:string" />
</message>
<message name="ObtenerEstudianteDadoIdExpedienteResponse">
      <part name="ObtenerEstudianteDadoIdExpedienteReturn" type="xsd:anyType" />
</message>
```

Figura 18. Fragmento del WSDL del servicio Akademos

```
<xsd:complexType name="Persona">
      <xsd:all>
            <xsd:element name="Apellidos" type="xsd:string" />
            <xsd:element name="Area" type="typens:Area" />
            <xsd:element name="Cargo" type="typens:Cargo" />
            <xsd:element name="CI" type="xsd:string" />
            <xsd:element name="Direccion" type="xsd:string" />
            <xsd:element name="EsAlta" type="xsd:boolean" />
            <xsd:element name="EsBaja" type="xsd:boolean" />
            <xsd:element name="EsDocente" type="xsd:boolean" />
            <xsd:element name="FechaAlta" type="xsd:dateTime" />
            <xsd:element name="FechaNacimiento" type="xsd:dateTime" />
            <xsd:element name="IdExpediente" type="xsd:string" />
            <xsd:element name="Militante" type="xsd:integer" />
            <xsd:element name="Municipio" type="typens:Municipio" />
            <xsd:element name="NivelEscolaridad" type="xsd:string" />
            <xsd:element name="Nombres" type="xsd:string" />
            <xsd:element name="Sexo" type="xsd:boolean" />
            <xsd:element name="Telefono" type="xsd:string" />
      </xsd:all>
</xsd:complexType>
```

Figura 19. Fragmento del WSDL del servicio Assets

```
        <operation name="ObtenerPersonasDadoldCategoria">
            <soap:operation soapAction="urn:PersonalWSAction" />
            <input>
                <soap:body namespace="urn:PersonalWS" use="encoded"
                encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
            </input>
            <output>
                <soap:body namespace="urn:PersonalWS" use="encoded"
                encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
            </output>
        </operation>
    </binding>
    <service name="PersonalWSService">
        <port name="PersonalWSPort" binding="typens:PersonalWSBinding">
            <soap:address location="http://personal.uci.cu/servicios/v4/PersonalWS.php" />
        </port>
    </service>
```

Figura 20. Fragmento del WDSL del servicio Personal

Existen datos que solicitó el interesado que no se encuentran en ciertos sistemas fuentes porque esta información no se encuentra disponible en las especificaciones de los servicios de la UDDI, quizás por razones de seguridad o disponibilidad. En cualquier caso se debe elaborar el Reporte de Respuesta a Demanda de Requisitos de Información que refleja que datos no pueden ser reflejados en el MDC, y permite que el interesado conozca esto expresando su conformidad**.**

Interesado:		
Dato No Atendido	**Fuente**	**Causa**
Docente	Personal	No está disponible en el sistema o la especificación del servicio
Militancia	Personal	No está disponible en el sistema o la especificación del servicio
Porciento de Atención: 93% **Firma de Conformidad:**		

Figura 21. Reporte de Respuesta a Demanda de Requisitos de Información mostrando incidencias en la obtención de los datos en los sistemas fuentes.

Basándose en la información agrupada en los documentos XML entregados por los sistemas fuentes de información se elabora el Modelo de Datos Conceptual General en el que se define el formato estándar con el que la comunicación tendrá lugar. Para ello se agrupa la información referente a una persona, que es el objeto del que se quiere intercambiar información.

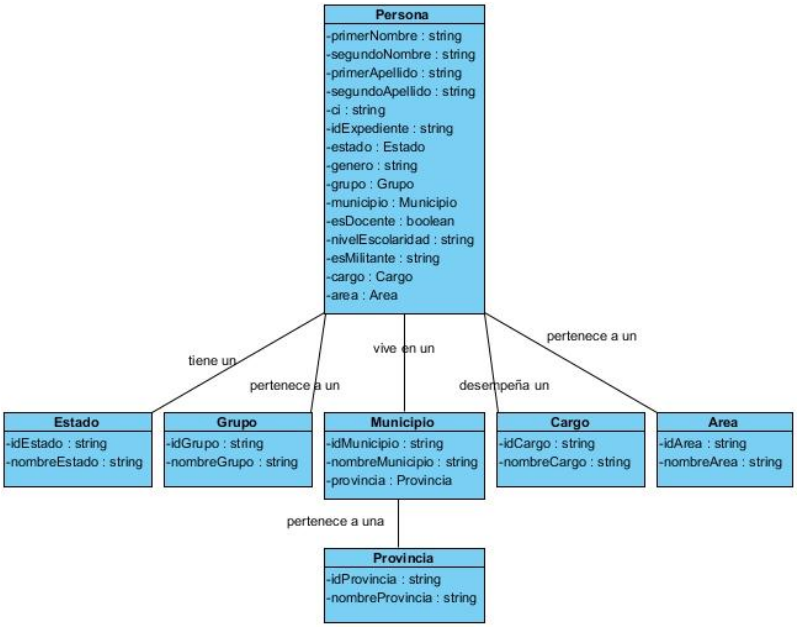


Figura 22. Artefacto: Modelo de Datos Conceptual General para Caso de Estudio UCI

La siguiente actividad a la elaboración del Modelo de Datos Conceptual General es la conciliación del mismo con los interesados. Esta actividad será omitida del caso de estudio pues no existe tal interesado. El caso de estudio se elabora con datos reales, pero no ha sido solicitada la aplicación del mismo.

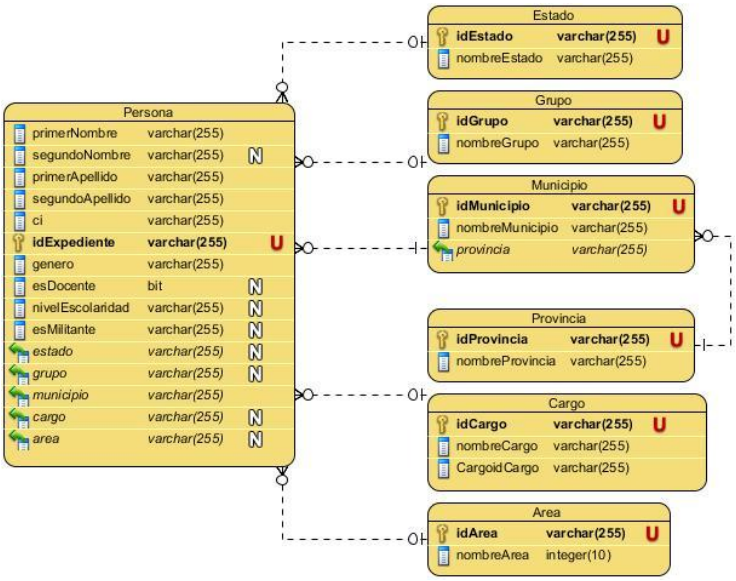


Figura 23. Artefacto: Modelo de Datos Lógicos para el Caso de Estudio UCI

A partir de la conformidad que expresó el interesado queda confirmado el avance del desarrollo y basándose en el Modelo de Datos Conceptual General se define la lógica del MDC mediante la elaboración del Modelo de Datos Lógicos, ver Figura 23.

Una vez definida la lógica del MDC se generan los Modelos de Mensaje Canónicos, ver Figura 24, Figura 25 y Figura 26, que contienen las especificaciones de los formatos de los datos de manera estándar para cada aplicación fuente y/o destino. En este caso específico se expusieron las descripciones de los servicios (WSDL) correspondientes a cada aplicación.

```
<xsi:element name="ObtenerDatosNoDocentesDadoCIResponse">
    <xsi:complexType>
        <xsi:sequence>
            <xsi:element name="datosNoDocentes" type="tns:DatosNoDocentes"/>
        </xsi:sequence>
    </xsi:complexType>
</xsi:element>
<xsi:complexType name="DatosDocentes">
    <xsi:sequence>
        <xsi:element name="primerNombre" type="xs:string"/>
        <xsi:element name="segundoNombre" type="xs:string" minOccurs="0"/>
        <xsi:element name="primerApellido" type="xs:string"/>
        <xsi:element name="segundoApellido" type="xs:string"/>
        <xsi:element name="estado" type="xs:string"/>
        <xsi:element name="grupo" type="xs:string"/>
        <xsi:element name="idExpediente" type="xs:string"/>
    </xsi:sequence>
</xsi:complexType>
```

Figura 24. Fragmento del Modelo de Mensaje Canónicos para Akademos

```
<wsdl:message name="ObtenerTrabajadorDadoCIRequest">
    <wsdl:part name="ci" element="tns:ObtenerTrabajadorDadoCI"/>
</wsdl:message>
<wsdl:message name="ObtenerTrabajadorDadoCIResponse">
    <wsdl:part name="trabajador" element="tns:ObtenerTrabajadorDadoCIResponse"/>
</wsdl:message>
<wsdl:message name="ObtenerNombre_Cargo_AreaDadoSolapinRequest">
    <wsdl:part name="solapin" element="tns:ObtenerNombre_Cargo_AreaDadoSolapin"/>
</wsdl:message>
<wsdl:message name="ObtenerNombre_Cargo_AreaDadoSolapinResponse">
    <wsdl:part name="trabajadorCorto" element="tns:ObtenerNombre_Cargo_AreaDadoSolapinResponse"/>
</wsdl:message>
```

Figura 25. Fragmento de Modelo de Mensaje Canónico para Assets

```
<xs:complexType name="Trabajador">
    <xs:sequence>
        <xs:element name="primerNombre" type="xs:string"/>
        <xs:element name="segundoNombre" type="xs:string" minOccurs="0"/>
        <xs:element name="primerApellido" type="xs:string"/>
        <xs:element name="segundoApellido" type="xs:string"/>
        <xs:element name="ci" type="xs:string"/>
        <xs:element name="idExpediente" type="xs:string"/>
        <xs:element name="genero" type="xs:string"/>
        <xs:element name="cargo" type="xs:string"/>
        <xs:element name="area" type="xs:string"/>
    </xs:sequence>
</xs:complexType>
<xs:element name="ObtenerTrabajadorDadoCI">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="ci" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
```

Figura 26. Fragmento de Modelo de Mensaje Canónico para Personal

Con los Modelos de Mensaje Canónico definidos se expone el MDC en el ESB, garantizando la comunicación entre las aplicaciones y con el Interesado. Para materializar esto, se deben transformar los mensajes que intervienen en la comunicación. Para ello se hace uso de los lenguajes XSLT y XPath, descritos en el Capítulo 1. A continuación se muestran fragmentos de la trasformación realizada para cada aplicación:

```
<xsl:template match="/">
    <xsl:choose>
    <xsl:when test="local-name(/child::node()[last()]) = 'ObtenerDatosDocentesEstudianteDadoSolapin'">
        <urn:ObtenerEstudianteDadoIdExpediente xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="h
            <IdExpediente>
                <xsl:value-of select="new:ObtenerDatosDocentesEstudianteDadoSolapin/new:idExpediente"/>
            </IdExpediente>
        </urn:ObtenerEstudianteDadoIdExpediente>
    </xsl:when>
    <xsl:when test="local-name(/child::node()[last()]) = 'ObtenerDatosNoDocentesDadoCI'">
        <urn:ObtenerEstudianteDadoCI xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.
            <CI>
                <xsl:value-of select="new:ObtenerDatosNoDocentesDadoCI/new:ci"/>
            </CI>
        </urn:ObtenerEstudianteDadoCI>
    </xsl:when>
    </xsl:choose>
</xsl:template>
```

Figura 27. Fragmento de la transformación de los parámetros de entrada de Akademos

```
<xsl:when test="string-length(substring-before(ns1:ObtenerPersonaDadoIdExpedienteResponse/ObtenerPersonaDadoIdEx
    <new:primerNombre>
        <xsl:value-of select="ns1:ObtenerPersonaDadoIdExpedienteResponse/ObtenerPersonaDadoIdExpedienteReturn/No
    </new:primerNombre>
    <new:segundoNombre/>
</xsl:when>

<xsl:otherwise>
    <new:primerNombre>
        <xsl:value-of select="substring-before(ns1:ObtenerPersonaDadoIdExpedienteResponse/ObtenerPersonaDadoIdEx
    </new:primerNombre>
    <new:segundoNombre>
        <xsl:value-of select="substring-after(ns1:ObtenerPersonaDadoIdExpedienteResponse/ObtenerPersonaDadoIdExp
    </new:segundoNombre>
</xsl:otherwise>
```

Figura 28. Fragmento de la transformación de los parámetros de salida de Assets

```
<xsl:template match="/">
    <xsl:choose>
    <xsl:when test="local-name(/child::node()[last()]) = 'ObtenerNombre_Cargo_AreaDadoSolapin'">
        <urn:ObtenerPersonaDadoIdExpediente xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xm
        <pIdExpediente>
            <xsl:value-of select="new:ObtenerNombre_Cargo_AreaDadoSolapin/new:solapin"/>
        </pIdExpediente>
    </urn:ObtenerPersonaDadoIdExpediente>
    </xsl:when>
    <xsl:when test="local-name(/child::node()[last()]) = 'ObtenerTrabajadorDadoCI'">
        <urn:ObtenerPersonaDadoCI xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="h
        <pCI>
            <xsl:value-of select="new:ObtenerTrabajadorDadoCI/new:ci"/>
        </pCI>
    </urn:ObtenerPersonaDadoCI>
    </xsl:when>
    </xsl:choose>
</xsl:template>
```

Figura 29. Fragmento de la transformación de los parámetros de entrada de Personal

Para comprobar la validez y efectividad de las transformaciones implementadas se debe consumir los servicios generados. Para ello se hace uso de la funcionalidad Probar Servicio del ESB, resultando las siguientes pantallas correspondientes a una operación de cada aplicación:

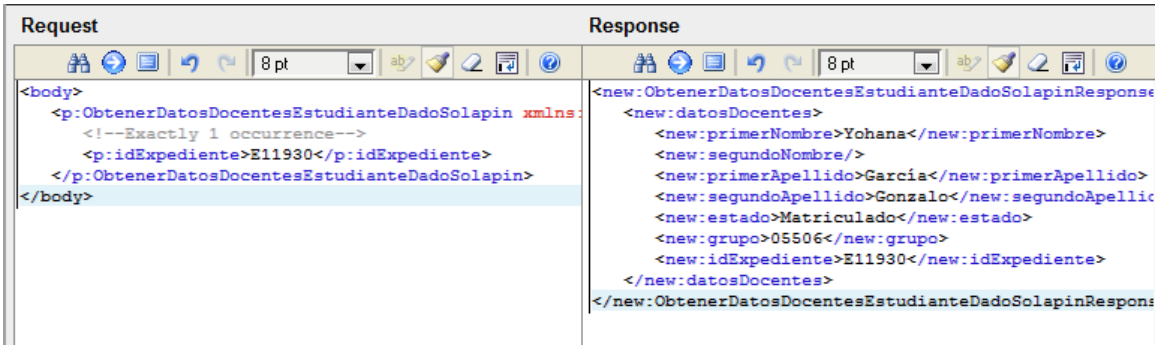


Figura 30. Servicio Akademos Consumido en operación ObtenerDatosDocentesEstudianteDadoSolapin

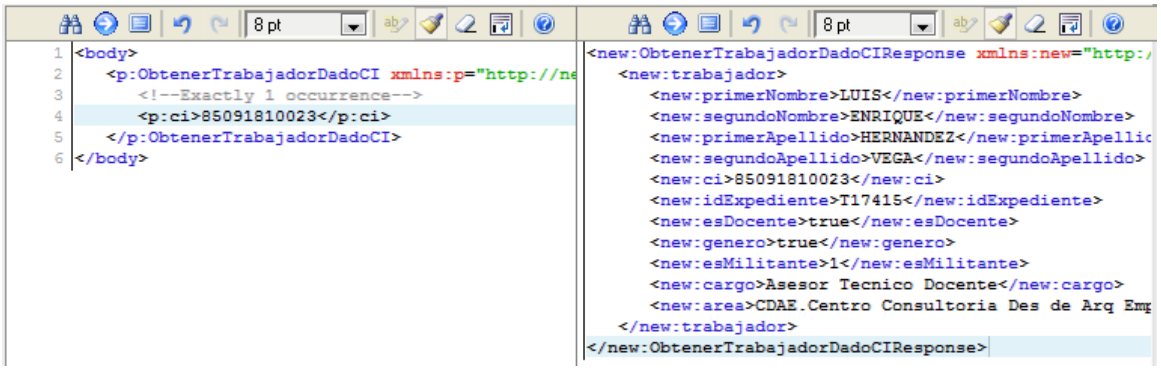


Figura 31. Servicio Assets Consumido en operación ObtenerTrabajadorDadoCI

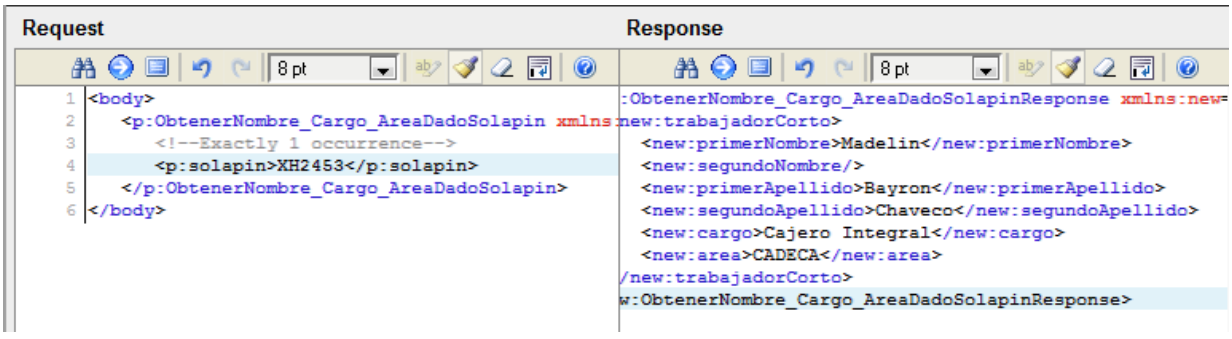


Figura 32. Servicio Personal Consumido en operación ObtenerNombre_Cargo_AreaDadoSolapin

Una vez obtenidas estas pantallas se puede decir que el MDC es funcional y garantiza las transformaciones a la forma canónica y viceversa. Para finalizar el proceso de desarrollo se debe validar el MDC a fin de establecer formalmente el fin del proyecto y verificar la conformidad del Interesado con lo obtenido. Para ello se verifica el cumplimiento de los Requerimientos de información obtenidos en la

primera actividad, teniendo en cuenta el Reporte de Respuesta a Demanda de Requisitos donde se reflejan los datos que no pudieron ser atendidos, comparando con los servicios consumidos obtenidos tras la exposición del MDC en el ESB.

Reporte de Cumplimiento de Requisitos Integración	
Interesado:	
Centro: Centro de Consultoría y Desarrollo de Arquitecturas Empresariales	
Porciento de Atención de necesidades: 93%	
Porciento de Satisfacción de necesidades: 100%	
Firma Interesado:	**Firma Centro:**

Figura 33. Reporte de Cumplimiento de Requisitos de Integración para el presente caso de estudio

3.1.3. Resultados esperados

Teniendo en cuenta el caso de estudio anteriormente elaborado, se puede decir que se cumplieron las expectativas del desarrollo, pues la comunicación entre los sistemas fue exitosa, ya que se garantizaron las transformaciones pertinentes a cada mensaje entre las aplicaciones y el MDC. Finalmente los servicios fueron consumidos mostrando los resultados esperados.

3.2. Validación de la calidad de la propuesta de solución mediante el Método Experto

El Método Experto usa como fuente de información y evaluación un conjunto de personas seleccionadas por su conocimiento en un área determinada, en este caso el de las iniciativas de soluciones de integración, así como a los patrones de integración empresarial.

Se usa este método de validación ya que no es suficiente con la elaboración del caso de estudio anteriormente expuesto para determinar ciertos aspectos como la calidad: completitud y facilidad de implementación; dado que no existe un interesado real que pueda establecer una conformidad con el desarrollo del MDC. Para este caso en específico se utiliza el Método Delphi, contemplado dentro del Método Experto.

Para la elaboración del Método Experto se transitan las etapas de Selección de expertos, Ejecución y procesamiento de la encuesta y Análisis conclusivo de resultados.

3.2.1. Proceso de selección de expertos

El método Delphi no cuenta con un valor exacto que determine el valor óptimo para el número de expertos que se debe seleccionar [33]. En este caso se seleccionaron 7 de 10 propuestas iniciales.

Para la selección de estos posibles expertos se siguieron una serie de criterios de selección entre los que se encuentran ser graduado del nivel Superior, conocimientos sobre Arquitectura Orientada a Servicios, experiencia laboral, capacidad de análisis y pensamiento lógico y disposición para participar en la validación.

Se realizó una encuesta (ver Anexo 6) individualmente a cada posible experto, midiendo los criterios de selección anteriores.

3.2.1.1. Cálculo del Coeficiente de Competencia

Para valoración de las competencias de los expertos se debe calcular el coeficiente de competencia (k), que se calcula de acuerdo con la opinión del experto sobre su nivel de conocimiento (kc) especificado en la encuesta, multiplicado por 0.1, de forma que en una escala de 0 a 1 se obtenga el nivel de conocimiento de cada experto; y con el coeficiente de argumentación (ka), establecido por la sumatoria de los valores correspondientes a las fuentes que le permiten argumentar sus criterios, ver Anexo 7.

$$k = \frac{1}{2}(ka + kc)$$

Una vez calculado el coeficiente de competencia, se determina el nivel de competencia de acuerdo a la siguiente escala:

- Si 0,8 < k < 1,0 el Coeficiente de Competencia es alto.
- Si 0,5 < k < 0,8 el Coeficiente de Competencia es medio.
- Si k < 0,5 el Coeficiente de Competencia es bajo.

Los expertos que se seleccionaron presentaron nivel de competencia alto o medio en la distribución expresada en la siguiente gráfica, basada en lo obtenido en el cálculo del coeficiente de competencia descrito en el Anexo 8.

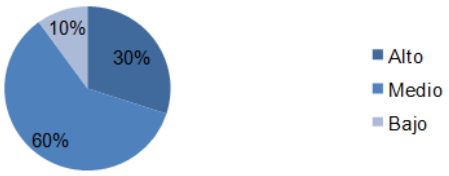


Figura 34. Grado de competencia de los expertos expresado en porcientos

3.2.2. Elaboración del cuestionario para la validación de la propuesta

A los expertos seleccionados se les hizo una segunda encuesta en la se reflejan los aspectos no medidos en el caso de estudio elaborado. Para ello se les facilitó un extracto de la propuesta de solución para que la estudiasen y pudieran emitir un criterio certero sobre la misma mediante el cuestionario. Con la elaboración de este segundo cuestionario se pretende validar el alcance de la estrategia para el desarrollo de un MDC y la calidad de la estrategia para el desarrollo de un MDC teniendo en cuenta:

1.1. Completitud.

1.2. Correspondencia entre artefactos, actividades y roles.

1.3. Facilidad de implementación.

3.2.3. Análisis conclusivo de los resultados

Una vez terminadas las encuestas a los 7 expertos seleccionados se resumen los resultados obtenidos. En el Anexo 9 son reflejadas las respuestas de las preguntas (P) de la encuesta para cada experto encuestado (E).

El 100% de los Expertos considera que el alcance de la estrategia está bien definido. El 85% de los expertos califica de satisfactorio el grado de correspondencia entre los artefactos y las actividades.

En el caso de la definición de los roles para el desarrollo de un MDC, el 100% de los expertos opinan que son los necesarios para esta tarea. Mientras que el nivel de completitud de la estrategia es catalogado positivamente por el 85% de los expertos encuestados.

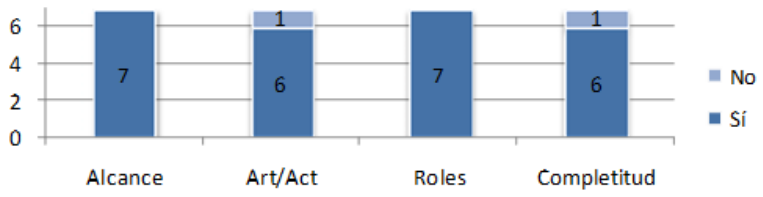


Figura 35. Calificación de indicadores de la estrategia por los expertos.

En el caso de la facilidad de implementación de la estrategia para el desarrollo de un MDC propuesta, el 71% de los encuestados considera que el nivel de dificultad de la implementación de la estrategia en Medio, el resto lo califica de Fácil, ningún experto lo considera Difícil.

Figura 36. Diagrama de la opinión de los expertos acerca la dificultad de implementación

Conclusiones Parciales

En el presente capítulo se valida la propuesta de solución mediante la utilización de un caso práctico en un entorno real de integración y a través de la utilización del Método Experto para calificar aspectos cualitativos como la calidad de la solución propuesta. Se comprueba finalmente que un MDC es un elemento factible a la hora de integrar aplicaciones, sobre todo en aquellos escenarios que las aplicaciones no deban sufrir alteraciones, ya que este modelo tiene como principio de comunicación la transformación de los mensajes a los formatos definidos en el MDC.

Por otra parte en cuanto a la calidad de la estrategia propuesta se comprobó mediante la realización de encuestas a expertos en el tema, que:

- La propuesta tiene un nivel medio de facilidad de implementación
- Tiene bien definido su alcance, el grado de correspondencia entre los artefactos y las actividades es satisfactorio.
- Los roles definidos son los necesarios para el desarrollo de un MDC.
- La estrategia propuesta tiene un nivel de completitud adecuado correspondiente para el desarrollo de una solución de integración.

Conclusiones

Después de desarrollar el presente trabajo diploma se concluye que:

1. Para la elaboración de la estrategia propuesta se investigaron las terminologías referentes al área de conocimiento pertinente a las soluciones de integración, así como las herramientas y lenguajes de desarrollo para cumplimentarla. Se escogió para el modelado la herramienta Visual Paradigm para UML y como herramienta de gestión empresarial, la Suite WSO2 Carbon. Para las transformaciones de los mensajes se utilizó el lenguaje XSLT y XPath.
2. La estrategia propuesta fue descrita mediante Actividades, Roles y Premisas, siendo las actividades la columna vertebral de la estrategia. Estas fueron definidas mediante el apoyo de un flujo de trabajo individual, así como el rol que la lleva a cabo, los artefactos de entrada y salida de cada una, así como las herramientas y técnicas a utilizar.
3. Una vez definida la estrategia para el desarrollo de un MDC, se valida la misma mediante un caso de estudio en el escenario de la UCI y mediante el Método Delphi. Se comprueba finalmente que las transformaciones de los mensajes ocurren desde las formas de las aplicaciones a la forma canónica y viceversa, garantizando la comunicación entre los interesados y las aplicaciones fuentes de información. Se establece también que la estrategia propuesta cuenta con buena calidad, ya que los expertos afirman que aspectos como la completitud, el alcance, la relación entre actividades y artefactos, los roles y la facilidad de implementación son positivos.

Recomendaciones

- Agregar a la estrategia procedimientos que garanticen la gestión del MDC.
- Definir un subproceso de pruebas al MDC, de forma que se especifiquen detalladamente las pruebas que se le harían al MDC.

Referencias Bibliográficas

1. Gregor Hohpe, B.W., *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions* 2003: Addison Wesley. 736.
2. Binildas C. A., *Service Oriented Java Business Integration*. 1ra ed. 2008: Packt Publishing Ltd. 433.
3. Brian Byrne, J.K., David McCarty, Dr. Guenter Sauter, Peter Worcester, *The information perspective of SOA design, Part 4: The value of applying the canonical modeling pattern in SOA.* developerWorks, 2008.
4. Miko Matsumura, B.B., Jignesh Shah, *Adopción de SOA para Dummies®, edición especial de Software AG*. 2009: Wiley Publishing, Inc. 99.
5. Evelson, B., *Pros and cons of using a vendor provided analytical data model in your BI implementation*, in *Forrester Blogs*. 2010.
6. Kapoor, H., *Canonical Data Model, Case For Canonical Data Model*, in *Harshit Kapoor, A techie's take….* 2010.
7. Srikant, S. *Logical Data Modeling: A Key to Successful Enterprise Data Warehouse Implementations*. 2006 [cited 2012 9/5/2012]; Available from: http://www.information-management.com/issues/20060901/1062071-1.html.
8. Judson, C. (2007) *How to create a canonical data model?* Real Life Service Oriented Architecture.
9. Thomas Erl, *SOA Design Patterns*. 2009. 856.
10. Mónica Canto, D.P., Andrés Segurola, *Service Oriented Architecture (SOA)*, Facultad de Ingeniería, Universidad de la República: Montevideo. p. 6.
11. Norbert Bieberstein, S.B., Marc Fiammante, Keith Jones, Rawn Shah, *Service-Oriented Architecture Compass: Business Value, Planning, and Enterprise Roadmap*. 2005: IBM Press. 272.
12. Michelson, B.M., *Event-Driven Architecture Overview*, in *Event-Driven SOA Is Just Part of the EDA Story*. 2006, Patricia Seybold Group. p. 9.
13. Maréchaux, J.-L. *Combining Service-Oriented Architecture and Event-Driven Architecture using an Enterprise Service Bus*. 2006 [cited 2012 13/2/2012].
14. Hohpe, G., *Programming Without a Call Stack –*

Event-driven Architectures. 2006.

15. Cornejo, J.E.G. *¿Qué es UML?* [cited 2012 10/2/2012]; Available from: http://www.docirs.cl/uml.htm.
16. Rodríguez, Y.G. and Y.R.D. Cobas, *Sistema de autenticación por tarjetas de códigos de barras y códigos de identificación*

personal para el proyecto Seguridad del Centro de Informática Industrial. . 2010, Universidad de las Ciencias Informáticas: La Habana. p. 89.

17. *Introduction to Oracle Enterprise Service Bus*. 2008 [cited 2011 20/11/2011]; Available from: http://download.oracle.com/docs/cd/E12524_01/doc.1013/e12638/esb_intro.htm
18. Lapuente, M.J.L. *Hipertexto: El nuevo concepto de documento en la cultura de la imagen.* 2005 [cited 2012 12/2/2012]; Available from: http://www.hipertexto.info/documentos/xml.htm.
19. Cornejo, J.E.G. *¿Cuáles son las características que debe tener una herramienta UML?* . [cited 2012 12/2/2012]; Available from: http://www.docirs.cl/caracteristica_herramienta_uml.htm.
20. Sierra, D. *Visual Paradigm For Uml*. 2005 [cited 2012 12/2/2012]; Available from: http://www.slideshare.net/vanquishdarkenigma/visual-paradigm-for-uml.

21. *JBoss ESB tutorial* 2012 [cited 2012 9/2/2012]; Available from: http://www.mastertheboss.com/jboss-esb/78-jboss-esb.html.
22. Boggs, W. and M. Boggs, *Mastering UML with Rational Rose 2002*, ed. D. Crossman. 2002: SYBEX. 714.
23. ORACLE, *ORACLE SOA SUITE* p. 3.
24. Daigneau, R., *Service Design Patterns*, in *Fundamental desing solutions for SOAP/WSDL and REST ful Web Services*, M. Fowler and I. Robinson, Editors. 2011, Addison-Wesley: Westford, Massachusetts. p. 354.
25. David A. Chappell, *Enterprise Service Bus*. 1ra ed, ed. M. Hendrickson. 2004: O'Relly Media.
26. Conner, K., et al., *JBoss ESB* 2012, Packt Publishing. p. 289.
27. Gardner, J.R. and Z.L. Rendon, *XSLT & XPATH A guide to XML transformations*. 2002: Prentice Hall PTR.
28. *Lenguaje de Caminos XML (XPath)*

Versión 1.0. 1999 [cited 2011 18/11/2011]; Available from: http://www.sidar.org/recur/desdi/traduc/es/xml/xpath.html.
29. .
30. Ochoa, R.L., *Modelo de referencia para el desarrollo arquitectónico de sistemas de software en dominios de gestión*. 2011, Universidad de Ciencias Informáticas: La Habana. p. 97.
31. Selvage, M.Y., et al. *Four approaches to implementing a canonical message model in an ESB*. Exploring the Enterprise Service Bus 2008 [cited 2012 14/03/2012].
32. León, Y.G., *Proceso de diseño de servicios para proyectos SOA.* 2010, Univesisdad de las Ciencias Informáticas: La Habana. p. 125.
33. Salgado, C.S. and L.M. Almaguer, *Propuesta de Infraestructura Tecnológica para el desarrollo y despliegue de una iniciativa con Arquitectura Orientada a Servicios*. 2010, Universidad de las Ciencias Informáticas: La Habana. p. 107.

Anexos

Anexo 1. Planilla para la Identificación de Necesidad de Información.

Identificación de Necesidad de Información			
Solicitante	<<Nombre de la persona o sistema que solicita la información>>		
Requerimiento del proceso que necesita la información	<<Justificación de la necesidad de información>>		
Funcionalidades <<Enumeración de las funcionalidades requeridas>>			
Nombre de la funcionalidad	**Parámetro de entrada**	**Parámetro de salida**	**Descripción**
		<<Se especifica además el formato en el que se desea la información>>	
Sistema fuente	<<Nombre del sistema que contiene la información demandada>>		
Medio actual para la comunicación	<<Especificación del medio, si existe, por el que se obtiene actualmente la información demandada >>		
Limitantes en la obtención de la Información	<<Problemas por los que no se puede obtener la información o no se obtiene correctamente>>		

Anexo 2. Diccionario de Datos

Diccionario de Datos			
Interesado: <<Nombre de la persona o del sistema Interesado>>			
Sistema Fuente	**Nombre Dato**	**Tipo Dato**	**Descripción**

Anexo 3. Reporte de Respuesta a Demanda de Requisitos de Información.

Reporte de Respuesta a Demanda de Requisitos de Información		
Interesado: << Nombre de la persona o del sistema Interesado >>		
Dato No Atendido	**Fuente**	**Causa**
Porciento de Atención: <<Se calcula el porciento de atención>> **Firma de Conformidad:** <<El interesado firma demostrando su conformidad>>		

Anexo 4. Planilla de No Conformidad

Planilla de No Conformidad
Interesado: <<Nombre la persona o sistema interesado>>

Fecha: <<Para definir una Planilla de otra>>
Descripción de la No Conformidad: <<Descripción de los aspectos que con los que no se está conforme y por qué >>

Anexo 5. Reporte de Cumplimiento de Requisitos Integración

Reporte de Cumplimiento de Requisitos Integración	
Interesado: << Nombre la persona o sistema interesado >>	
Centro: << Nombre del centro que se encarga de garantizar la Integración >>	
Porciento de Atención de necesidades: <<porciento reflejado en el Reporte de Respuesta a Demanda de Requisitos de Información >>	
Porciento de Satisfacción de necesidades: <<porciento de satisfacción en el Servicio Consumido con respecto al total de atención disponible en el Reporte de Respuesta a Demanda de Requisitos de Información>>	
Firma Interesado: <<El interesado firma demostrando su conformidad>>	**Firma Centro:** <<El Arquitecto de Integración firma estableciendo el fin del proyecto>>

Anexo 6. Encuesta para la selección de expertos

Estimado compañero usted ha sido propuesto como posible experto por su experiencia y calificación profesional a los efectos de valorar la pertinencia y calidad del aporte que se propone con esta investigación, por ello se le solicita su cooperación. La aplicación del criterio de expertos requiere de algunos datos, los cuales se recopilarán a través de las siguientes preguntas, antes de someter a consulta la propuesta del autor.

1. Datos generales:

   Nombre y Apellidos: _____
   Título universitario: _____
   Categoría docente: _____
   Grado académico y/o científico: _____
   Centro de trabajo: _____
   Cargo que desempeña: _____

2. Marque con una cruz en la siguiente tabla, el valor que considere se corresponda con el grado de conocimiento que posee sobre iniciativas de soluciones de integración y patrones de integración empresarial. Tenga en cuenta el carácter ascendente de la escala que se utiliza.

1	2	3	4	5	6	7	8	9	10

3. Valore el grado de influencia que han tenido diversas fuentes, en el nivel de conocimiento que posee usted sobre iniciativas de soluciones de integración y patrones de integración empresarial. Marque con una cruz el grado de influencia en Alto (A), Medio (M) y Bajo (B) de cada una de las fuentes reflejadas en la siguiente tabla:

Fuentes de Argumentación	Grado de Influencia de Cada Fuente		
	Alto (A)	Medio (M)	Bajo (B)
Análisis teóricos realizados por usted.			
Experiencia obtenida realizada con el tema de la investigación.			
Trabajo de autores nacionales.			
Trabajo de autores extranjeros.			
Su propio conocimiento del estado del problema en el centro.			
Su intuición.			

Anexo 7. Valores correspondientes a las fuentes de argumentación para el Método Delphi

Fuentes de Argumentación	Grado de Influencia de Cada Fuente		
	Alto	Medio	Bajo
Análisis teóricos realizados por usted.	0.3	0.2	0.1
Experiencia obtenida realizada con el tema de la investigación.	0.5	0.4	0.2
Trabajo de autores nacionales.	0.05	0.05	0.05
Trabajo de autores extranjeros.	0.05	0.05	0.05
Su propio conocimiento del estado del problema en el extranjero	0.05	0.05	0.05
Su intuición.	0.05	0.05	0.05
Totales	**1.0**	**0.8**	**0.5**

Anexo 8. Determinación de nivel de competencias de los posibles expertos

Número de los Expertos	Kc	Ka	K	Competencia
E1	0.8	0.9	0.85	Alto
E2	0.1	0.5	0.3	Bajo
E3	0.6	0.6	0.6	Medio
E4	0.8	0.9	0.85	Alto
E5	0.7	0.9	0.8	Alto
E6	0.7	0.8	0.75	Medio
E7	0.6	0.6	0.6	Medio
E8	0.4	0.6	0.5	Medio
E9	0.7	0.8	0.75	Medio

E10	0.6	0.8	0.7	Medio

Anexo 9. Encuesta para la validación de aspectos no medibles en el caso de estudio

Encuesta para la tesis "Propuesta de Estrategia para el Desarrollo de un Modelo de Datos Canónicos"

Compañero (a): _____

El presente trabajo de diploma se propone entre sus objetivos la proposición de una estrategia para desarrollar un Modelo de Datos Canónicos. Con este fin se solicita su valiosa colaboración para evaluar la estrategia planteada. Por lo que se solicita que usted responda el conjunto de preguntas que se encuentran a continuación:

1. ¿Considera usted que el alcance de la estrategia está bien definido?
   Sí ____ No____ ¿Por qué? _____

   _____
2. ¿Considera usted que el grado de correspondencia que se establece entre los artefactos y las actividades planteadas es satisfactorio?
   Sí ____ No____ ¿Por qué? _____
   _____
3. ¿Considera usted que los roles definidos son los necesarios para desarrollar un Modelo de Datos Canónicos?
   Sí ____ No____ ¿Por qué? _____
   _____
4. ¿Cree usted que el nivel de completitud de este grupo de indicadores es el más adecuado?
   Sí ____ No____ ¿Por qué? _____
   _____
5. ¿Cómo valora usted la facilidad de implementación de la estrategia?
   Fácil ____ Medio _____ Difícil_____

Otras consideraciones:

1. Evaluación personal del encuestado sobre la estrategia propuesta: En este punto el encuestado emite una breve evaluación personal sobre el proceso.

2. Fallas o problemas hallados por el encuestado en la estrategia propuesta: En este punto el encuestado manifiesta los problemas o deficiencias hallados en el proceso.

3. Recomendaciones del encuestado a la estrategia propuesta: En este punto el encuestado expresa algunas recomendaciones para el proceso.

Anexo 9. Resultados de la encuesta para la validación de la propuesta de solución

	E1	E2	E3	E4	E5	E6	E7
P1	Sí	Sí	Sí	Sí	Sí	Sí	Sí
P2	Sí	Sí	Sí	Sí	No	Sí	Sí
P3	Sí	Sí	Sí	Sí	Sí	Sí	Sí
P4	Sí	Sí	Sí	Sí	Sí	Sí	No
P5	Fácil	Medio	Medio	Medio	Fácil	Medio	Medio

Glosario de Términos

-B-

Back-Bone: se refiere a las principales conexiones troncales de Internet.

-C-

Clúster: El término clúster se aplica a los conjuntos o conglomerados de computadoras construidos mediante la utilización de hardwares comunes y que se comportan como si fuesen una única computadora.

Caché: Es un conjunto de datos duplicados de otros originales, con la propiedad de que los datos originales son costosos de acceder, normalmente en tiempo, respecto a la copia en la caché. Cuando se accede por primera vez a un dato, se hace una copia en el caché; los accesos siguientes se realizan a dicha copia, haciendo que el tiempo de acceso medio al dato sea menor.

-M-

Metadato: Los metadatos son datos altamente estructurados que describen información, describen el contenido, la calidad, la condición y otras características de los datos. Es "Información sobre información" o "datos sobre los datos".