RISI 7(2), 2010 (9-19)
Revista de Investigación de Sistemas e Informática
Facultad de Ingeniería de Sistemas e Informática
Universidad Nacional Mayor de San Marcos

ISSN 1815-0268 (versión impresa)
ISSN 1816-3823 (versión electrónica)

Conceptos fundamentales de Ingeniería dirigida por Modelos y Modelos de Dominio Específico

Fundamental concepts of engineering headed by models and models of specific domain

John Ledgard Trujillo Trejo, Armando David Espinoza Robles

Universidad Nacional Mayor de San Marcos

Facultad de Ingeniería de Sistemas e Informática

jtrujillot@unmsm.edu.pe, davidespinozarobles@yahoo.com.mx

RESUMEN

La inclusión de los paradigmas de SOA y MDE en metodologías y procesos de desarrollo de software ha cobrado gran importancia en los últimos años. La definición de servicios en las organizaciones es un área que viene siendo tratada por los constructores de software empresarial, existiendo actualmente varias propuestas metodológicas y técnicas para abordar el problema. La ingeniería dirigida por modelos tiene como principal objetivo especificar y explicitar los términos del negocio en modelos. Los modelos no solo al inicio del proceso de desarrollo de software, sino en todo el ciclo de vida a través de transformaciones. Con las transformaciones se le está ofreciendo a los desarrolladores de SOA la posibilidad de poder realizar la automatización de sus procesos llevando a un nivel de abstracción mayor la obtención de los artefactos.

Palabras clave: modelo, servicios, SOA, MDE, transformación

ABSTRACT

The inclusion of the paradigms of SOA and MOU in methodologies and processes of software development, has gained considerable importance in recent years. The definition of services in the organizations, is an area that is being treated by the builders of business software, exist now several methodological proposals and techniques to address the problem. The engineering headed by models has as its main objective specify and clarify the terms of the business models. The models not only to the top of the process of software development, but throughout the life cycle through transformations. With the transformations is being offered to the developers of SOA the possibility of able to carry out the automation of their processes leading to a level of abstraction increased the collection of artifacts.

Keywords: model, services, SOA, MDE, processing

1. INTRODUCCIÓN

El término Orientado a Servicio representa una manera diferente de separar la lógica del negocio en las organizaciones. La lógica para resolver un problema complejo se podría construir, desarrollar y administrar descomponiendo en una colección de "piezas" más pequeñas pero interrelacionadas, donde cada una de estas "piezas" se centrará en resolver una parte específica del problema. En las empresas los servicios se diseñan para ser independientes, autónomos y para interconectarse adecuadamente, pueden combinarse y recombinarse con suma facilidad en procesos complejos que respondan a las necesidades de cada momento en el seno de una organización.

Las Arquitecturas Orientadas a Servicios (SOA: Service Oriented Architecture) se encuentran en proceso de implantación en las principales empresas de mercado. Estas arquitecturas se caracterizan por estar compuestas de unidades funcionales, servicios, que proporcionan servicios de negocio.

Aunque inicialmente la definición SOA comprendía únicamente Servicios Web, la realidad de las empresas muchas veces implica ampliar esta definición para incluir otros servicios que comprenden desarrollos adhoc (transacciones Cobol / CICS, Visual Basic, Fortan, C...) y paquetes de software (SAP, Documentum, Staffware...).

Ingeniería dirigida por modelos (MDE) es un enfoque fortalecido en muchas de las técnicas exitosas aplicadas en la ingeniería del software. Se caracteriza por: a) poner al frente el nivel de abstracción ocultando los detalles específicos en la plataforma; b) Aprovechar el uso de los modelos en todas las fases de desarrollo de software para mejorar la comprensión; c) desarrollar Framework y lenguajes específicos para lograr la comprensión del dominio; y d) obtener provecho de las transformaciones para automatizar trabajo repetitivo y mejorar la calidad del software.

Los modelos son los artefactos claves de MDE. Por esa razón el término modelo es la primera definición. Para MDE "Un modelo es una descripción de un (parte de) sistema escrito en un lenguaje bien definido. Un lenguaje bien definido es un lenguaje con formas (sintaxis) y significados (semántica) bien definidos, los cuales pueden ser interpretados automáticamente por una computadora".

SOA y MDE facilitan y mejoran el desarrollo de software en un entorno muy cambiante para las empresas. El principal objetivo para la aplicación conjunta de los paradigmas SOA y MDE es brindar soporte al ciclo de vida del desarrollo de software de las organizaciones, aportando a su mejora continua y permitiendo una rápida reacción a los cambios necesarios en los mismos, originados tanto por causas internas o externas a la misma.

2. FUNDAMENTACIÓN TEÓRICA

2.1. Service-Oriented Architectures (SOA)

Contexto histórico de SOA

No existe precisión alguna de quién fue el primer personaje que acuño el término de SOA, pero en los primeros informes acerca de SOA en 1996, los analistas de Gartner Roy W. Schulte y Yefim V. NATIS expresaron que: "Alexander Pasik, acuñó el término de SOA en una exposición de "middleware" en 1994. Pasik estaba trabajando en XML o servicios web y creyó necesario crear el concepto de SOA porque la arquitectura cliente/servidor había perdido su clásico significado. Para evitar la confusión entre los nuevos y viejos significados de cliente / servidor, Pasik destacó "orientación de servicios" que alentó a los desarrolladores a diseñar aplicaciones de negocio de SOA [1].

SOA es la convergencia de diferentes tecnologías y es un ambiente de integración sin exclusiones. SOA es el producto de una evolución de las siguientes tecnologías:

XML (eXtensible Markup Language). El origen de SOA es consecuencia del nacimiento del XML (padre de múltiples tecnologías) estándar de facto para la descripción de datos a ser intercambiados sobre la web.

RPC (Remote Procedure Call). Protocolo que permite a un programa de ordenador ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambos.

Web Services. Son servicios utilizados para transmitir y recibir datos por aplicaciones heterogéneas de diferentes empresas u organizaciones vía web. Todo esto se realiza con un conjunto de protocolos y estándares: Web Services Protocol Stack, SOAP (Simple Object Access Protocol), XML-RPC (XML Remote Procedure Call), HTTP (Hypertext Transfer Protocol), FTP (File Transfer Protocol), o SMTP (Simple Mail Transfer Protocol), WSDL (Web Services Description Languages),

UDDI (Universal Description, Discovery and Integration), WS-Security (Web Service Security).

SOA no es un concepto nuevo. Los ingenieros de software entendieron sus principios a mediados de los 80, cuando llegó al mercado la computación distribuida y las llamadas a procedimientos remotos. Las Arquitecturas de Computación Distribuida de los 90 no alcanzaron la aceptación esperada: Open Software Foundation's (OSF's), Distributed Computing Environment (DCE), Object Management Group's (OMG's) y Common Object Request Broker Architecture (CORBA) [2].

2.2. Definición de SOA

La Arquitectura Orientada a Servicios es un paradigma, es un modelo de arquitectura de software basado en la definición de servicios reutilizables, con interfaces públicas bien definidas, donde los proveedores y consumidores de servicios interactúan en forma desacoplada para realizar los procesos de negocio.

SOA proporciona una metodología y un marco de trabajo basado en servicios. Además, está muy orientada y alineada con el negocio, por lo que normalmente se habla de SOA como un modelo de arquitectura tanto de Tecnología de Información (TI) como corporativo.

La mayoría de las organizaciones para ser exitosas tratan de reducir los costos y maximizar la utilización de la tecnología existente; para lograrlo las empresas intentan ser más competitivas y avanzan en sus prioridades estratégicas. Para conseguir estos propósitos se trazan dos líneas: la heterogeneidad y el cambio. Muchas empresas tienen diferentes sistemas, aplicaciones y arquitecturas de diferentes tecnologías y algunas bastante antiguas. Integrar productos de diferentes proveedores y de diferentes tecnologías es una ardua tarea, por eso cada vez más las aplicaciones están orientadas al servicio [3].

Figura 1. Componentes de SOA [4].

SOA está conformada por proveedores y consumidores de servicios, así como por un directorio de los mismos. Estos roles se grafican en la Figura 1. En SOA existen tres operaciones principales: publish, find y bind [4].

En un SOA, los recursos de software son empaquetados en forma de "servicios", el cual están bien definidas, auto-contenida en módulos que proporcionan la funcionalidad estándar de negocios y son independientes de la situación o el contexto de otros servicios [5].

2.3. Elementos de SOA

Una Arquitectura Orientada a Servicio está compuesta por elementos funcionales y elementos relacionados con la calidad de servicio [3].

Elementos funcionales:

- Transporte: para llevar las peticiones de servicios y respuestas entre el proveedor y el consumidor del servicio.
- Protocolo de comunicación del servicio: se establece entre el proveedor y el consumidor del servicio.
- Descripción del servicio: describe servicio, cómo debe invocarse y que datos son requeridos para la invocación.
- Servicio: Describe un servicio que está disponible para utilizarse.
- Proceso de negocio: conjunto de servicios, invocados de una manera específica, con una determinada secuencia y con ciertas reglas particulares para llevar a cabo la funcionalidad de negocio requerida.
- Registro de Servicio: repositorio de servicios y las descripciones de las mismas.

Aspectos de la calidad del servicio:

- Política: reglas o condiciones entre el proveedor y consumidor de los servicios.
- Seguridad: Conjunto de reglas aplicados para la identificación, autorización y el control de acceso de los consumidores de servicios.
- Transacción: Conjunto de atributos que pueden

ser aplicados a un grupo de servicios para conseguir un resultado consistente.

- Gestión: Conjunto de atributos que se aplican para manejar a los proveedores del servicio o a los consumidores.

Servicio

En esencia, un servicio de TI es una representación de funcionalidad de algunas empresas [1].

Un servicio es una unidad bien definida del trabajo realizado por un componente y empaquetados para facilitar el acceso. La idea es implementar la funcionalidad una sola vez, hacerlo bien y hacerlo ampliamente accesible [6].

La arquitectura orientada a servicio debe ser dinámica y para que funcione como tal, tiene que cumplir las siguientes características [7]:

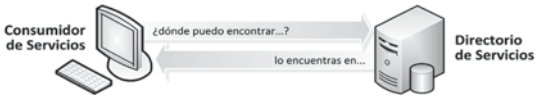

Figura N°. 2. Detalle de la operación publish [4].

- Los servicios tienen que ser modulares.
- Los servicios tienen que soportar la interoperabilidad.
- Los servicios tienen que tener la descripción perfectamente establecida.
- Los servicios tienen que ser transparentes a la localización.
- Los servicios tienen que ser independientes del lenguaje de implementación.
- Los servicios tienen que ser transparentes al protocolo de comunicación.

Figura N°. 3. Detalle de la operación find [4].

- Los servicios tienen que ser independientes del Sistema Operativo y del hardware utilizado.

Directorio y roles en un servicio de SOA

El directorio de servicios es un intermediario entre proveedores y consumidores [4], con su uso es posible: una gran escalabilidad en el número de servicios, desacoplar a los consumidores de los proveedores, permitir actualizaciones de servicios y proveer un servicio de búsqueda a los consumidores. Todo esto en tiempo de ejecución.

Un rol refleja un tipo de participante en un SOA [5]. Los roles de una Arquitectura Orientada a Servicio son [3]:

- **Consumidor de servicio:** Es una aplicación, un módulo de software u otro servicio que requiere un servicio.
- **Proveedor de servicios:** Es una entidad a la que se puede acceder a través de la red y que acepta y ejecuta peticiones de los consumidores. Publica las interfaces de los servicios en el registro de servicios para que los consumidores puedan descubrirlos y puedan acceder a ellos.
- **Registro de servicios:** Es el que permite que los servicios puedan ser descubiertos.

El funcionamiento de los roles en la arquitectura es el siguiente [27]: inicialmente, un proveedor de servicios necesita registrar en el directorio el servicio que está listo a proveer. Para ello utiliza la operación publish (publicación) (Figura 2).

Una vez registrado, los consumidores potenciales del servicio, utilizando el directorio, pueden localizar dinámicamente al servicio mediante la operación find (localización) (Figura 2.3).

Cuando el servicio ya fue identificado por el consumidor, el directorio le provee información sobre la forma en cómo puede contactar al proveedor y entonces el consumidor es capaz de preparar la comunicación utilizando la operación bind (enlazar e invocar) (Figura 2.4), para posteriormente utilizar el servicio en sus necesidades.

Figura N°. 4. Detalle de la operación bind y del uso del servicio [4].

Beneficios de una Arquitectura SOA

Con SOA se pueden lograr beneficios para que las organizaciones puedan conseguir procesos de negocios exitosos y dinámicos [3]: solucionar los problemas existentes de TI en la organización, facilitar la integración y gestión de la complejidad de las funcionalidades heterogéneas del negocio, aceleración de la puesta en producción de los sistemas, reducción de costes y aumento de la reutilización y estar preparados para el cambio.

Las arquitecturas SOA aportan un gran valor a las organizaciones, pero hay que analizar bien para migrar a SOA, ya que no es una tarea trivial.

También se pueden ver los beneficios de una Arquitectura SOA desde el punto de vista empresarial [3]: eficiencia en transformación de los procesos de negocio en servicios, capacidad de respuesta de los servicios y adaptabilidad ante los de cambios de TI y los modelos de negocio.

2.4. Model-Driven Engineering (MDE)

Contexto histórico de MDE

Desde los inicios de la informática, los investigadores y desarrolladores de software han creado abstracciones que ayuden a programar en términos de su diseño, y no pensando en el entorno computacional -por ejemplo, CPU, memoria y dispositivos de red- y protegerse de la complejidad de estos entornos.

Estas abstracciones incluyeron a los lenguajes de programación y las tecnologías de plataforma. Por ejemplo, assembler y Fortran, protegieron a los desarrolladores de la complejidad de la programación con código de máquina. Los sistemas operativos, como OS/360 y Unix, protegieron a los desarrolladores de las complejidades de la programación directamente al hardware.

En los 80s se desarrollaron los CASE Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador), que se centraron en los métodos y herramientas de desarrollo de software, permitiéndoles expresar sus diseños en términos de representaciones gráficas de propósito general para la programación, tales como máquinas de estado, diagramas de estructura, y diagramas de flujo de datos. Uno de los objetivos de CASE era permitir un análisis más minucioso de programas gráficos que tengan menos complejidad que los lenguajes de programación de propósito general convencionales. Otro objetivo era sintetizar la aplicación de las representaciones gráficas de los artefactos para reducir el esfuerzo manual de codificación, depuración y portabilidad de programas.

El CASE no fue ampliamente adoptado en la práctica. Hubo una serie de problemas como: las representaciones gráficas de los lenguajes de propósito general no eran portables a diferentes plataformas (DOS, OS/2 o Windows), carecían de apoyo importante para la calidad del servicio (QoS), propiedades, tales como la distribución transparente, tolerancia a fallos, y la seguridad.

Otro problema con CASE fue la incapacidad de manejar el desarrollo de aplicaciones a gran escala y de sistemas complejos. En general, las herramientas CASE no apoyaron la ingeniería concurrente, limitándose a la escritura de programas por una sola persona o por un equipo que serializó su acceso a los archivos utilizados por estas herramientas. Por otra parte, debido a la falta de potentes plataformas de middleware común, las herramientas CASE apuntaron a entornos de ejecución propietarios lo que hizo difícil integrar el código que se generaba con otros lenguajes y tecnologías de software de plataforma. Las Herramientas CASE no lograron apoyar a muchos ámbitos del desarrollo de aplicaciones de manera efectiva, debido a que todas las representaciones gráficas fueron demasiado genéricas y no customizables.

Como resultado de ello, el CASE tuvo poco impacto en el desarrollo de software comercial durante los años 1980 y 1990, centrándose principalmente en algunos ámbitos, como el procesamiento de llamadas de telecomunicaciones. En la medida en que se aplicaron herramientas CASE, en la práctica, se limitaron en gran medida a un subconjunto de herramientas que permitieron a los diseñadores trazar los diagramas de arquitecturas de software y documentar decisiones de diseño, que los programadores utilizaban para ayudar a guiar la creación y evolución de sus aplicaciones hechas a mano, ya que no había relación directa entre los diagramas y la aplicación a desarrollar, y que rara vez se sincronizaba el código durante etapas posteriores de los proyectos [8].

Definición de MDE

La Ingeniería Dirigida por Modelos (MDE) es una metodología de desarrollo de software que se centra en la creación de modelos, o abstracciones, más cerca de

algunos conceptos de dominio en lugar de conceptos de la informática (o algorítmica). Todas las formas de ingeniería se basan en modelos de diseño de sistemas del mundo real. Los modelos se utilizan en muchos sentidos: para entender aspectos específicos del sistema, predecir cualidades del sistema, la razón sobre el impacto de los cambios, y comunicar las principales características del sistema a las partes interesadas. En un enfoque Dirigido por los modelos, es decir, el sistema de modelos tiene suficiente detalle como para permitir la generación de un sistema completo de aplicación de los modelos propios. De hecho, "el modelo es el código", es decir, la atención se centra en el modelamiento y el código es generado mecánicamente a partir de modelos [9].

Realizando una comparación de MDE con la orientación a objetos: en MDE "todo es un modelo", y en la orientación a objetos "todo es un objeto" [10].

MDE es un enfoque abierto e integrador que abarca muchos otros espacios tecnológicos de manera uniforme. Un espacio tecnológico de trabajo es un contexto con una serie de conceptos relacionados, el cuerpo de conocimientos, herramientas, conocimientos necesarios y posibilidades [11]. Ejemplos de espacios tecnológicos son lenguajes de programación con sintaxis concreta y abstracta, ontología de ingeniería basadas en XML, Lenguajes de Sistemas de Gestión de Bases de Datos (DBMS: Database Management System) y la Aquitectura Dirigida por Modelos (MDA: Model-Driven Architecture).

La tecnología de Ingeniería Dirigida por Modelos ofrece un enfoque prometedor para hacer frente a la incapacidad de los lenguajes de tercera generación para aliviar la complejidad de las plataformas y expresar conceptos de dominio de manera eficaz [8].

Objetivos de MDE

El objetivo primordial de la mayoría de los vendedores de herramientas es entregar beneficios a los usuarios:

- La necesidad de separar de manera muy clara la lógica de negocio y la tecnología utilizada.
- La separación de las preocupaciones (separation of concerns[1]), considerado como uno de los principios fundamentales en la ingeniería de software. Este principio afirma que un determinado problema implica diferentes tipos de preocupaciones, que deben ser identificadas y separadas para hacer frente a la complejidad, y para lograr los factores de calidad de ingeniería tales como la robustez, adaptabilidad, mantenibilidad y reutilización.
- La necesidad de modelizar y especificar la parte de la lógica de negocios a un nivel abstracto, la plataforma de implementación y de proyectar el nivel abstracto sobre la plataforma.
- Generar software nuevo a partir de modelos.
- Apoyar a los desarrolladores en su productividad.
- Realizar el proceso de construcción de software a través de modelos durante todo del ciclo de vida del software.
- Generar los cambios en las partes del modelo.

Beneficios de MDE

El MDE está destinado a aumentar la productividad al máximo, la compatibilidad entre sistemas, simplificando el proceso de diseño, y promoviendo la comunicación entre los individuos y los equipos que trabajan en el sistema.

MDE tiene por objeto aumentar la rentabilidad de una empresa derivado del esfuerzo de desarrollo de software. Este beneficio se entrega en dos formas básicas [12]: mediante la mejora de la productividad a corto plazo de los desarrolladores (incrementar valor de los primeros artefactos de software) y mediante la mejora de la productividad a largo plazo de los desarrolladores (evitar la rápida obsolescencia de los objetos primarios de software)

Un segundo aspecto del MDE, y estratégicamente más importante, es la reducción de la sensibilidad de los primeros objetos de arte al cambio. Indican cuatro formas fundamentales de los cambios que son de particular importancia [12]:

- **Personal.** El desarrollo del conocimiento vital no debe ser sólo almacenado en la mente de los desarrolladores, esta información se perderá en el caso demasiado frecuente de las fluctuaciones de personal. Por lo tanto, esta información debe ser de fácil acceso para cualquier persona que no sean los creadores del artefacto software.

1 Este término de "separation of concerns" fue probablemente acuñado por Edsger W. Dijkstra, en su documento de 1974 "On the role of scientific thought".

- **Requerimientos.** Los cambios de los requerimientos es un problema conocido en la ingeniería de software. En el actual entorno empresarial aparecen nuevas funcionalidades más rápido que nunca. Todos estos cambios deben tener un bajo impacto sobre los sistemas existentes en términos de mantenimiento y no perturbar los sistemas en línea.
- **Plataformas de desarrollo.** Están en un estado de constante evolución. Para disociar la vida de un artefacto de software de la herramienta de desarrollo utilizado para su creación inicial, es necesario disociar el objeto o el modelo que representa el objeto de la herramienta de desarrollo.
- **El despliegue de plataformas.** Para aumentar la vida útil de los objetos de software es necesario protegerlos contra los cambios en el despliegue de la plataforma.

3. REVISIÓN DEL ESTADO DEL ARTE

Estado del arte de metodologías de SOA

Un conjunto de metodologías han surgido para atender la enorme demanda de orientación de procesos y las mejores prácticas en proyectos de SOA. Sin embargo, un estudio sobre estas metodologías y un análisis de sus propiedades es que la mayoría son tratados desde un punto de vista general, sin referirse a propuestas específicas. Arsanjani de IBM [13] en general clasifica a SOA en seis categorías de criterios: los procesos dirigidos por negocios, herramienta basada en MDA, envoltura (Interfaces) heredadas, componentes heredados, dirigido por datos, y dirigido por mensajes. Papazoglou proporciona una guía de investigación, donde entre otras cosas, estudia brevemente el estado de la técnica y algunos grandes desafíos en ingeniería orientada a servicios. Zimmermann discute sobre el análisis y técnicas de diseño orientadas a los servicios para el desarrollo y la integración con IBM SOMA como método a modo de ejemplo [14].

IBM Analisis y Diseño Orientado a Servicios (Service-Oriented Analysis and Design, SOAD) [15]: SOAD propone elementos que deberían formar parte de un servicio orientado hacia el análisis y la metodología de diseño; en resumen, es un famework en lugar de una metodología holística. SOAD se basa en las técnicas existentes, tales como Object Oriented Analysis and Design (OOAD), el Computing Business Driven (CDB), y Business Process Management (BPM). SOAD También introduce técnicas específicas de SOA, como la conceptualización de servicios, clasificación y agregación del servicio, las políticas y los aspectos, se reúnen en medio del proceso, semántica de intermediación, y el servicio de recolección.

IBM Modelado y Arquitectura Orientada a los Servicios (Service Oriented Modeling and Architecture, SOMA) [16]. SOMA es una verdadera metodología de modelado por IBM que consta de tres pasos: identificación, especificación, y la realización de los servicios, los flujos (los procesos de negocio), y componentes de la realización de servicios. El proceso es muy iterativo e incremental. Sin embargo, debido a que SOMA es propiedad de IBM, su especificación completa no está disponible. Se ha anunciado recientemente que el Rational Unified Process se ha combinado con SOMA, el resultado es lo que se denomina IBM RUP para SOMA.

SOA Repetible Calidad (Repeatable Quality RQ) [17]. RQ SOA es una metodología propietaria de Sun Microsystems que se basa en un RUP parecido, con proceso iterativo e incremental que consta de cinco fases: creación, elaboración, construcción, transición, y la concepción. Compatible con artefactos UML se utilizan para documentar las diversas prestaciones de estas fases.

Proceso CBDI-SAE [18]: El Foro CBDI está desarrollando actualmente una metodología SOA como parte de CBDI-SAE SOA Reference Framework (RF). Las cuatro grandes áreas de la disciplina del proceso son: consumir, proporcionar, administrar y habilitar. Cada zona de grupos de disciplinas similares se desglosan a unidades de proceso y, a continuación, a las tareas. Esta metodología pretende que la empresa de TI a través de la integración de arriba abajo el análisis de los requerimientos del negocio, así como de abajo hacia arriba de la integración de los sistemas heredados. El proceso de CBDI-SAE tiene como objetivo abarcar todo el ciclo de vida de SOA, incluido el despliegue, la supervisión y las actividades de governance.

Service Oriented Architecture Framework (SOAF) [14]. SOAF consta de cinco fases principales: elicitación de información, servicio de identificación, definición del servicio, realización del servicio y plan de trabajo y planificación. Se basa en dos tipos de actividades de modelado simultáneamente: “Para-ser” de modelado, que

es enfoque de la empresa de arriba a abajo orientado a describir los procesos de negocio, y el modelado "tal y como son", que es el enfoque ascendente que describe como los procesos de negocio actuales se forman por las aplicaciones existentes.

Service Oriented Unified Process (SOUP) [19]. Como su nombre indica, este enfoque por el Sr. K. Mittal se basa principalmente en el Rational Unified Process. Su ciclo de vida consta de seis fases: incepción, definir, diseñar, construir, implementar y soporte. Sin embargo, carece de la documentación detallada SOPA y deja margen para la adaptación. Se utiliza en dos variantes ligeramente diferentes: una inicial para la adopción de proyectos SOA RUP y la otra la adopción de una mezcla de RUP y XP para el mantenimiento de SOA existentes.

Metodología [20]. En su documento, Papazoglou examina metodología de desarrollo de unos servicios desde el punto de vista de los proveedores y los consumidores, que intenta cubrir la totalidad del ciclo de vida de SOA. Se basa en parte en los bien establecidos modelos de desarrollo, como el RUP, el CDB, y BPM. La metodología utiliza un proceso iterativo e incremental que comprende una preparatoria y ocho distintas fases principales.

Thomas Erl [21]. La metodología de análisis y diseño orientado a servicios documentado en el libro de Thomas Erl Esta metodología es una guía paso a paso a través de dos fases principales: análisis y diseño. Las actividades en la fase de análisis es de arriba hacia abajo en la visión empresarial donde se identifican los candidatos de servicio. Estos sirven como entrada para la próxima fase, orientada hacia el servicio de diseño, donde el servicio de los candidatos se especifica en detalle y, posteriormente, se realizan como servicios web.

Estado del arte de métodos MDE en SOA

Las empresas necesitan una mayor comunicación y colaboración entre la TI y el negocio. Los software de las empresas se han vuelto demasiado complejos para ser eficaces y cumplir con los objetivos del negocio. Una enorme cantidad de dinero se gasta para mantener los sistemas existentes en funcionamiento.

La solución a este gran problema es SOA (Service-Oriented Architecture). Al utilizar el enfoque modular de SOA se produce la separación de funciones en un nivel superior, reduciendo la complejidad o, y la estructuración de la misma. Al separar las aplicaciones y la orquestación de servicios, el sistema resultante puede ser fácilmente modificado por la organización de los servicios de otra manera o simplemente por el uso de otros servicios en una determinada organización.

La globalización en la que estamos inmersos lleva a las organizaciones a determinar la mejor manera de diseñar y desarrollar sus Sistemas de Información con objeto de que sean fácilmente integrables y portables. Para conseguir ambos objetivos, se hace necesario trabajar con modelos conceptuales que recojan fielmente la semántica del negocio y que, por medio de herramientas automáticas (o semiautomáticas) de transformación de modelos, obtengan el modelo que se implante físicamente en la plataforma que corresponda. Pero, para realizar con éxito esta tarea, es conveniente trabajar dentro de un marco metodológico que oriente y guíe al informático en el proceso, por lo que se considera que una propuesta dirigida por modelos es la más adecuada, tanto técnica como económicamente, para que las organizaciones puedan fácilmente adaptarse a los cambios tecnológicos que se producen en cada momento [22].

MDD4SOA: Model-Driven Service Orchestration [23]. MDD4SOA se desarrolló como parte del proyecto de la UE SENSORIA, modeliza los aspectos estructurales de los servicios, orquestaciones de servicios, políticas y otras propiedades no funcionales. Los modelos diseñados en el enfoque de MDD4SOA generan código en multiples lenguajes, entre ellos Lenguaje de Ejecución de Procesos de Negocio (BPEL: Business Process Execution Language), y Lenguaje de Descripción de Servicios Web (WSDL: Web Services Description Language,) Java, y el lenguaje formal Jolie. Usa una extensión de UML para SOA –denominado perfil UML4SOA– que es un alto nivel de lenguaje de dominio especifico para la modelización de orquestación de servicios.

Model Driven Development of a Service Oriented Architecture (SOA) Using Colored Petri Nets [24]. Método que usa Colored Petri Nets[2] (CPN) para el desarrollo dirigido por modelos y la evaluación de la calidad de un software SOA para el Departamento de De-

2 Colored Petri Nets (CPN) es un lenguaje orientado a gráficos para el diseño, especificación, simulación y verificación de los sistemas.. En particular, es muy apropiada para los sistemas que consiste en una serie de procesos que se comunican y sincronizar. Los ejemplos típicos de áreas de aplicación son los protocolos de comunicación, sistemas distribuidos, sistemas de producción automatizados, análisis de flujo de trabajo y los chips VLSI.

fensa. El alto grado de concurrencia y comunicaciones sincrónicas y asincrónicas inherentes a SOA le hace un buen candidato.

El Departamento de Defensa y la Agencia de Sistemas de Información de Defensa (DISA) han definido un conjunto básico de servicios empresariales para su uso en defensa relacionados con sistemas SOA, como parte de una iniciativa llamada Servicios empresariales centrado en red (NESI: Red-Centric Enterprise Solutions de interoperabilidad). Estos Net-Centric Enterprise Services (NCES) definida por NESI se definen por un conjunto de redes centradas en los servicios, los nodos y los servicios públicos del Departamento de Defensa.

Una propuesta de solución general es disponer de una arquitectura responsable con un mediador para el descubrimiento dinámico, la sensibilización y el equilibrio de carga. Para permitir la interoperabilidad de múltiples tipos de conexión se requiere un interior bien definido, el formato y el protocolo, bien definidas las interfaces externas al interior y los mecanismos de transporte interno y buffering. Llamamos a esta arquitectura de Defensa de la Arquitectura Orientada a Servicios (SODA).

El enfoque es integrar un modelo basado en este enfoque de desarrollo de software que pueden ser utilizados para guiar la aplicación y evaluar la fiabilidad, la escalabilidad y el rendimiento del producto SODA utilizando simulación, verificación y validación. Además, la investigación es proporcionar, mediante el modelado y análisis, retroalimentación para el desarrollo del Departamento de Defensa y las comunidades para mejorar su comprensión de capacidades y limitaciones, la influencia de arquitectura y técnicas de toma de decisiones proceso, y establecer las expectativas de la tecnología de red centrada en la arquitectura comportamientos.

Las redes de Petri Proporcionan un lenguaje de modelado (o anotación) adecuado para sistemas distribuidos en los que la comunicación, la sincronización y la distribución de recursos juegan un papel importante. CPNs combinan los puntos fuertes de las redes de Petri ordinarias con la fuerza de un alto nivel de lenguaje de programación junto con un riguroso mecanismo de abstracción. Redes de Petri proporcionar primitivas para el proceso de interacción, mientras que el lenguaje de programación que proporciona las primitivas para la definición de tipos de datos y la manipulación de datos valores.

A Model-driven Approach to Describe and Predict the Performance of Composite Services [25]. Enfoque dirigido por modelos para la predicción del rendimiento de la integración en procesos de composición de servicios llevados a cabo por el uso de BPEL (Lenguaje de Ejecución de Procesos de Negocios para la Web Servicios). El enfoque propuesto se basa en P-WSDL (Performance habilitado WSDL), una extensión orientado al rendimiento, el lenguaje para describir la información acerca de capacidades de servicio y los mecanismos de invocación. P-WSDL es una ligera extensión de WSDL para la descripción de las características de un servicio web.

Model-driven synthesis of SOA Solutions [26]. Enfoque promovido por la División de Investigación de IBM, Model-Driven Business Transformation (MDBT) utiliza un software síntesis dirigido por tecnología de modelos para generar producción automáticamente – calidad de componentes de servicio de negocios para un nivel alto de modelo de procesos de negocios. Se presenta la Business Entity Life Cycle Analysis (BELA), técnica para la realización de integración en el modelado de soluciones SOA basado en MDBT y Service-oriented modeling and architecture (SOMA) el método end-to-end para el Aplicación y solución de SOA, desarrollo de soluciones y aplicaciones SOA. BELA se centra en las actividades y entidades, identifica y especifica entidades de negocios y descompone procesos de negocios, genera interacción de componentes, servicios de entidades de negocios y sus almacenes de datos y definiciones de interfaces de servicios. Este método reduce en 40% el costo del proyecto y reducción en un 20% en el tiempo del ciclo en comparación con los enfoques convencionales de SOA.

4. CONCLUSIONES

En este trabajo se presenta una visión general de las metodologías de SOA y MDE para relacionar los conceptos de modelos y servicios, que sirve para facilitar la comprensión y entendimiento del desarrollo de software de negocios y las relaciones involucrados en ambas áreas, y cómo se relacionan los conceptos de un área con los de la otra.

Por lo tanto, se puede concluir que es interesante contar con un método que relacione SOA y MDE obtener un modelo del negocio y a partir de allí con las transformaciones en las distintas fases del desarrollo de

software llegar al modelo de los servicios y finalmente al código.

Esta línea de trabajo presentada aquí, continúa con la ampliación de la taxonomía y creación del método, la descripción de los Metamodelos, y los algoritmos de transformación.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] Nicolai M. Josuttis, SOA in Practice: The art distributed system design, ed. O'Reilly Media, August 2007: First Edition.

[2] Gartner (Roy W. Schulte). "Predicts 2003: SOA Is Changing Software" , 2002 ID Number: AV-18-9758. http://www.gartner.com/resources/111900/111987/111987.pdf

[3] Otón Tortosa, Salvador. "Propuesta de una arquitectura software basada en servicios para la implementación de repositorios de objetos de aprendizaje distribuidos". Tesis doctoral. Universidad de Alcalá. Alcalá de Henares. España. 2006

[4] Edgardo Aviles López, "Arquitectura Orientada a Servicios para Redes Inalámbricas de Sensores", tesis para obtener el grado de maestro en ciencias en ciencias de la computación. centro de investigación científica y de educación superior de ensenada, Ensenada, Baja California. Noviembre de 2006.

[5] Papazoglou, M. P. and Heuvel, W. Service oriented architectures: approaches, technologies and research issues. The VLDB Journal 16, 3 (Jul. 2007), 389-415. DOI= http://dx.doi.org/10.1007/s00778-007-0044-3

[6] Paul C. Brown, Implementing SOA: Total Architecture in Practice, Ed. Addison Wesley Professional, April 09, 2008, pages: 736.

[7] James McGovern, Sameer Tyagi, Michael Stevens, Sunil Mathew. "Java Web Services Architecture". Chapter 2, Service Oriented Architecture, 2003.

[8] Douglas C. Schmidt. "Model-Driven Engineering", IEEE Computer, vol. 39, no. 2, pp. 25- 31, Feb., 2006.

[9] Flavio Oquendo. p-Method: A Model-Driven Formal Method for Architecture-Centric Software Engineering. ACM SIGSOFT Software Engineering Notes. Volume 31 Number 3. May 2006.

[10] Bézivin, J. In Search of a Basic Principle for Model Driven Engineering. UPGRADE , Vol. V (No. 2), 21-24, 2004.

[11] Kurtev, I., Bézivin, J., & Aksit, M. Technological Spaces: an Initial Appraisal. CoopIS, DOA'2002, Industrial track, 2002.

[12] Atkinson, C., & Kühne, T. Model-Driven Development: A Metamodeling Foundation. IEEE Softw, 20 (5), 36-41, 2002.

[13] A. Arsanjani, "Toward a pattern language for Service-Oriented Architecture and Integration, Part 1: Build a service eco-system", IBM Corporation, available from http://www.ibm.com/developerworks/webservices/library/ws-soa-soi/, July 2005.

[14] E Ramollari, D Dranidis and A J H Simons, A survey of service-oriented development methodologies, Proc. 2nd Young Researchers' Workshop on Service Oriented Computing, 11-12 June, eds. S Gorton, M Solanki and S Reiff-Marganiec, (Leicester: University of Leicester, 2007), 1-6.

[15] Olaf Zimmermann, Pal Krogdahl y Clive Gee. "Elements of Service-Oriented Analysis and Design", IBM Corporation, available from http://www.ibm.com/developerworks/webservices/library/ws-soad1/ , June 2004.

[16] A. Arsanjani, "Service-oriented modeling and architecture", IBM Corporation, available from http://www.ibm.com/developerworks/library/ws-soa-design1/, November 2004.

[17] SUN Microsystems, "SOA RQ methodology - A pragmatic approach", available from http://www.sun.com/products/soa/soa methodology.pdf.

[18] P. Allen, "The service oriented process", in CBDi Journal, February 2007, http://www.cbdiforum.com/report summary.php3?page=/secure/interact/2007-02/service oriented process.php&area=silver.

[19] M. P. Papazoglou and W. J. van den Heuvel, "Service-oriented design and development methodology", International Journal of Web Engineering and Technology (IJWET), 2006.

[20] Thomas, Earl, Service-Oriented Architecture: Concepts, Technology, and Design, Upper Saddle River: Prentice Hall PTR, 2005.

[21] Marta Zorrilla, Belén Vela, Esperanza Marcos, PhD. A Model Driven Approach to Design and Build XML

Schemas: A Case Study. revista AVANCES EN SISTEMAS E INFORMÁTICA. Universidad Nacional de Colombia Sede-Medellín. Vol 3 2007.

[22] P. Mayer, A. Schroeder, and N. Koch, "A Model-Driven Approach to Service Orchestration". Intl. Conference on Services Computing (SCC), Honolulu, USA, 2008.

[23] Vijay Gehlot, Thomas Way, Robert Beck and Peter DePasquale. "Model Driven Development of a Service Oriented Architecture (SOA) Using Colored Petri Nets." First Workshop on Quality in Modeling, ACM/IEEE 9th International Conference on Model Driven Engineering Languages and Systems, pages 63-77, October, 2006.

[24] D'Ambrogio, A. and Bocciarelli, P. 2007. A model-driven approach to describe and predict the performance of composite services. In Proceedings of the 6th international Workshop on Software and Performance (Buenos Aires, Argentina, February 5-8, 2007). WOSP '07. ACM, New York, NY, 78-89. DOI= http://doi.acm.org/10.1145/1216993.1217008

[25] Strosnider, J. K., Nandi, P., Kumaran, S., Ghosh, S., and Arsanjani, A. 2008. Model-driven synthesis of SOA solutions. IBM Syst. J. 47, 3 (Jul. 2008), 415-432.

[26] David Chappell, Tyler Jewell. Java Web Services. Publisher: O'Reilly. First Edition March 2002, 276 pages.