Propuesta metodológica para la implementación de una arquitectura orientada a servicios en entornos de Sistemas de Fabricación Distribuida

Medina Bulo, I. [(1)(*)]; García Domínguez, A. [(1)]; Aguayo, F. [(2)]; Sevilla, L. [(3)]; Marcos, M. [(4)]

(1) Departamento de Lenguajes y Sistemas Informáticos. Universidad de Cádiz. Escuela Superior de Ingeniería. c/ Chile 1, 11002, Cádiz.
(*) Autor al que debe enviársele la correspondencia. e-mail: inmaculada.medina@uca.es
(2) Escuela Universitaria Politécnica. Universidad de Sevilla. c/ Virgen de África, 7. Sevilla
(3) Departamento de Ingeniería Civil, Materiales y Fabricación. Universidad de Málaga. ETSII. Plaza el Ejido s/n, Málaga.
(4) Departamento de Ingeniería Mecánica y Diseño Industrial. Universidad de Cádiz. Escuela Superior de Ingeniería. c/ Chile 1, 11002, Cádiz.

RESUMEN

Los Sistemas de Fabricación de Siguiente Generación, concebidos por la organización IMS (Intelligent Manufacturing Systems), cubrirán la demanda de un mercado cada vez más cambiante y exigente, reuniendo de forma dinámica las funciones de múltiples sistemas internos y externos a la propia empresa de fabricación en lo que conocemos por una empresa extendida. Sin embargo, resulta difícil estructurar estos sistemas para conseguir la flexibilidad deseada y la interoperabilidad con los de las demás organizaciones. Además, un defecto en el sistema tendría un fuerte impacto: afectaría no sólo a la empresa propietaria del sistema, sino también a sus colaboradores. Por estas razones, creemos que una buena decisión sería emplear una arquitectura orientada a servicios diseñada siguiendo una metodología que adopte los servicios como concepto central, en vez de como un detalle de implementación. Para que la arquitectura sea fiable en su conjunto, tendrá que ayudar a localizar errores antes de que sucedan en un entorno de producción. En este artículo proponemos el uso de técnicas de pruebas específicas para este tipo de sistemas, realizamos una comparación de las metodologías existentes para el desarrollo de arquitecturas orientadas a servicios y esbozamos una serie de extensiones sobre una de las existentes para integrar técnicas de prueba.

KEYWORDS

Service-oriented architectures, web services, intelligent manufacturing systems, model-driven engineering, testing.

ENGLISH ABSTRACT

As envisioned by the IMS (Intelligent Manufacturing Systems), Next Generation Manufacturing Systems will satisfy the needs of an increasingly fast-paced and demanding market by dynamically integrating systems from inside and outside the manufacturing firm itself into a so-called extended enterprise. However, organizing these systems to ensure the maximum flexibility and interoperability with those from other organizations is difficult. Additionally, a defect in the system would have a great impact: it would affect not only its owner, but also its partners. For these reasons, we argue that a service-oriented architecture (SOA) would be a good candidate. It should be designed following a methodology where services play a central role, instead of being an implementation detail. In order for the architecture to be reliable enough as a whole, the methodology will need to help find errors before they arise in a production environment. In this paper we propose using SOA-specific testing techniques, compare some of the existing SOA methodologies and outline several extensions upon one of them to integrate testing techniques.

INTRODUCCION

Hoy en día, las empresas se hallan bajo la necesidad de competir en un mercado en el que los ciclos de vida de los productos son cada vez más cortos y se exigen mayores niveles de flexibilidad y calidad a menor coste. Para ello, necesitan ser capaces de reorientar y mejorar continuamente sus procesos de negocio en función de la situación y de forma rentable. Sin embargo, las plataformas centralizadas normalmente usadas en las empresas de hoy en día no pueden cambiarse tan rápidamente como las situaciones lo requerirían, y finalmente son éstas las que definen las prácticas a seguir, más que la propia situación del mercado.

Se necesita, por lo tanto, un enfoque distinto para estructurar los sistemas de información en la así llamada Siguiente Generación de Empresas de Fabricación (SGEF). Actualmente, se admite [1] a nivel conceptual la necesidad de distribuir las actividades a lo largo de varios sistemas de información especializados y de posteriormente integrarlas en lo que se conoce como una empresa extendida. Entre los protomodelos de empresa distribuida más conocidos [2], se encuentran las organizaciones holónicas, llamadas así por constituirse por una serie de actores semiautónomos que se interrelacionan entre sí a varios niveles, conocidos como holones.

Una vez establecido el marco conceptual sobre el que modelar a la organización, resulta evidente la necesidad de plasmarlo en un sistema de información. Afortunadamente, desde hace varios años, una forma de estructurar los sistemas de información que se ajusta bien a estas ideas ha ido recibiendo cada vez más atención: las arquitecturas orientadas a servicios o *service-oriented architectures* (SOA). La idea central detrás de ellas es organizar los sistemas de información no como sistemas integrados unidad a unidad de negocio o proyecto a proyecto, como se ha hecho comúnmente, sino como servicios individuales que pueden ser reutilizados a lo largo de la organización, o incluso por

otras organizaciones con las que se tiene relación. Posteriormente, estos servicios individuales pueden ser integrados en servicios de nivel superior con su propio valor añadido, modelando procesos de negocio completos en vez de operaciones individuales. A nivel de implementación, estas arquitecturas suelen implementarse utilizando servicios web (*web services* o WS), para así aprovechar las numerosas tecnologías, herramientas e infraestructuras ya implementadas para la World Wide Web y reducir las barreras tecnológicas.

Este enfoque y su implementación mediante WS ofrecen grandes ventajas a nivel tecnológico y de negocios, pero no dejan de tener sus propias dificultades. La máxima flexibilidad se conseguiría con servicios lo más granulares posible, pero implantar cada servicio tiene un coste adicional respecto a la simple implementación de la funcionalidad, por lo que en términos de coste de desarrollo a corto plazo, interesa tener pocos servicios. El óptimo global se halla en una granularidad media de los servicios, pero es difícil de localizar. A esto se le añade la dificultad inherente en el desarrollo de cualquier sistema distribuido de esta magnitud.

Para hacer frente a este nivel de complejidad, han surgido una serie de metodologías [3-5] orientadas a SOA que tratan de reducir el nivel de dificultad y asegurar resultados más controlables y repetibles en su implantación. Algunas de estas metodologías son revisiones de metodologías de desarrollo tradicional de software orientado a objetos, y otras han sido creadas desde cero alrededor del concepto de "servicio" tal y como se entiende en SOA. Nuestra opinión es que estas nuevas metodologías son las que pueden dar los mejores resultados, ya que disponen de un nivel mayor de abstracción y reflejan de forma más directa los procesos de negocio a implementar.

Otro problema, independientemente de la metodología que desarrollemos, es el hecho de que el mayor grado de integración y la mayor visibilidad de los servicios de la organización implican también una mayor dependencia en su correcto funcionamiento, y un mayor grado de impacto en caso de fallo.

En este artículo proponemos extender una de las metodologías existentes con los aspectos necesarios para integrar técnicas de prueba específicas para las arquitecturas orientadas a servicios, y esbozamos algunas de esas extensiones con el fin de ayudar a localizar fallos antes de que aparezcan en el entorno final de producción de los sistemas de información de las empresas de fabricación. Este artículo se estructura de la siguiente forma: tras introducir los conceptos esenciales detrás de las ideas ofrecidas en este artículo, realizaremos una comparación de las metodologías existentes y seleccionaremos una de ellas como base de nuestra nueva metodología. Posteriormente, concluiremos el artículo ofreciendo una relación de trabajos relacionados y resultados conseguidos.

CONCEPTOS PREVIOS

En esta sección revisaremos algunos conceptos básicos sobre los que se apoya el resto del texto de este artículo. Comenzaremos por un recorrido general sobre el tratamiento del problema, y continuaremos con las ideas centrales a nuestra propuesta: las arquitecturas orientadas a servicios y las metodologías de desarrollo dirigido por modelos.

ORGANIZACIONES VIRTUALES: LA EMPRESA EXTENDIDA

Ninguna empresa vive en un vacío: ha de relacionarse con el ecosistema formado por sus proveedores, diseñadores, fabricantes, subcontratistas, clientes y otras empresas competidoras. Para tener éxito, ha de añadir más valor a su producto final que los competidores, y esto se traduce, normalmente, en usar la información de forma que se pueda responder más rápidamente a las demandas del mercado.

Sin embargo, hoy en día las empresas se ven obligadas a interactuar más allá del ámbito geográfico local al que se encuentran habituadas. El uso de sistemas de información a través de Internet es por lo tanto un paso crucial, así como la estandarización de las prácticas de cada participante y la resolución de diversos problemas de logística. Una vez se ha completado dicha integración, la organización resultante se conoce con el nombre de *empresa extendida* [6]. Existe una amplia variedad de modelos para representar la estructura de una empresa extendida. Muchos de ellos pueden englobarse en una de tres categorías [2]: modelos biónicos, fractales u holónicos.

Los modelos biónicos se hallan inspirados en sistemas biológicos. La empresa consiste en una serie de tejidos (correspondientes a procesos, productos o servicios), formados a su vez por células que realizan diversas operaciones y reciben y producen artefactos codificados de forma genética (partes y productos, por ejemplo). Manteniendo la analogía, una célula se autorregula mediante la secreción de enzimas (información interna de control), y puede influenciar o ser influenciada por las hormonas (información del entorno y otras unidades) que se hallen en el entorno. Ante situaciones urgentes, un sistema nervioso puede responder rápidamente.

Por otro lado, los modelos fractales parten de las formas geométricas del mismo nombre. Se centran en la construcción de la empresa a partir de entidades similares entre sí a distintos niveles de abstracción. Se construye un fractal sobre otro cuando el fractal de nivel inferior no puede acometer todas las tareas que deben ser realizadas. De forma inversa, las metas de la organización descienden desde el fractal principal que modela a la organización completa, y se van concretando nivel a nivel. Cada fractal tiene un cierto grado de autonomía y capacidad de reorganización.

Finalmente, los modelos holónicos se inspiran en los estudios de los sistemas jerarquizados de Koestler, quien define el concepto de holón como una entidad que es a la vez un todo formado por agentes de nivel inferior, y una parte de varias

holarquías (jerarquías de holones) de nivel superior. Dichos holones combinan cierta autonomía con un grado de dependencia en otros holones del mismo nivel o del nivel inmediatamente superior.

Estos tres modelos poseen distintos enfoques, pero tienen en común la conceptualización de la empresa como una red dinámica de agentes con cierta autonomía y capacidad de colaboración. A nivel de implementación de estos modelos, uno de los mayores problemas de las empresas extendidas es el establecimiento de los canales de comunicación necesarios. Los sistemas de información suelen diseñarse e implementarse con planes a corto plazo específicos de cada unidad de cada organización, y en otras ocasiones existen aplicaciones cruciales que se diseñaron en una etapa anterior a la Web y resulta demasiado arriesgado reemplazar. Todo esto resulta en dificultades a la hora de establecer colaboraciones puntuales con empresas de otros países y aprovechar los nuevos procesos y otras ventajas que ofrecen.

ARQUITECTURAS ORIENTADAS A SERVICIOS

Normalmente, los sistemas de información utilizados por las empresas suelen dividirse en una serie de capas implementadas a lo largo de varias máquinas: una base de datos recoge toda la información, que es manipulada por una capa con la lógica de cada área de negocio y finalmente visualizada y manipulada por el usuario mediante una capa de presentación. Esta arquitectura ha sido utilizada con éxito para definir un número considerable de sistemas de gran envergadura de hoy en día. Encajan de forma natural con las empresas centralizadas, jerarquizadas y estables.

Sin embargo, no están exentos de problemas. Estos sistemas suelen estar desarrollados para satisfacer las necesidades de partes individuales de la organización a corto plazo. No tienen en cuenta la necesidad de cambiar las prácticas de negocio a medio o largo plazo: la lógica de negocio suele hallarse unida al diseño y la implementación. Tampoco se suelen diseñar con vistas a colaborar en el futuro con otras compañías originalmente no previstas. Con el tiempo, las empresas o bien rodean al sistema para innovar, o pierden la capacidad de reaccionar ante nuevas situaciones.

Por estas razones, un nuevo enfoque basado en arquitecturas orientadas a servicios ha recibido un gran nivel de atención en los últimos años. Más que un conjunto de tecnologías, es una forma distinta de organizar los sistemas [7] de información de una empresa. Éstos dejan de ser estructuras rígidas y centralizadas para convertirse en una serie de servicios reutilizables y recombinables independientemente para definir nuevos procesos de negocio y/o refinar los existentes. Las ideas subyacentes son muy similares a las de los modelos holónicos o basados en agentes: un servicio de nivel inferior puede formar parte de servicios de nivel superior o procesos de negocio, posiblemente colaborando con otros servicios, y con un cierto nivel de autonomía. Dado esto, consideramos muy adecuado implementar una arquitectura orientada a servicios en el contexto de los sistemas de fabricación distribuida.

Las tecnologías más utilizadas hoy en día para implementar SOA son las de los servicios web. Aprovechan muchas de las herramientas existentes y se basan en estándares abiertos para reducir las barreras tecnológicas.

Un fallo común a la hora de adoptar SOA es pensar que basta con adquirir y utilizar la última versión de la plataforma SOA escogida. Sólo se consiguen sus verdaderos beneficios una vez se hayan aplicado debidamente sus conceptos fundamentales, se haya definido un catálogo de servicios base de alto rendimiento y fiabilidad y exista una visión global de los servicios y procesos definidos desde los niveles más altos de la organización. Este proceso no es sencillo, por lo que surge la necesidad de definir metodologías que guíen su ejecución, al igual que cuando se desarrolla cualquier otro tipo de software. En una sección posterior haremos una comparativa de algunas de ellas.

INGENIERÍA DIRIGIDA POR MODELOS

En la actualidad, un problema al desarrollar software es el hecho de que existe un vínculo muy débil entre los modelos de alto nivel (es decir, más cercanos a los humanos) que se crean de los sistemas y el código que los implementan. En la práctica, lo que ocurre es que los lenguajes de modelado se usan solamente como vías de comunicación entre desarrolladores, y los modelos son de usar y tirar, ya que rápidamente quedan obsoletos al cambiar el código o los requisitos del sistema. Esto se traduce en una serie de problemas en varias actividades usuales: comprobar si se han implementado los requisitos pedidos por el cliente, optimizar un diseño ante nuevas situaciones, aprovechar las últimas tecnologías existentes o verificar si el sistema cumple determinadas condiciones.

Una nueva perspectiva en el desarrollo de software conocida como ingeniería dirigida por modelos (*model driven engineering* o MDE) y defendida [8] por el Object Management Group (OMG) bajo el nombre de arquitectura dirigida por modelos (*model driven architecture* o MDA) trata de dar la vuelta a esta situación. Proponen implementar el sistema a partir de una serie de modelos cada vez más detallados entre los cuales se puedan hacer correspondencias a distintos niveles de automatización. Así, volveríamos a elevar el nivel de abstracción al que se implementan los sistemas, como se consiguió con los lenguajes de alto nivel frente al código ensamblador.

En particular, la propuesta del OMG define tres tipos de modelos para cualquier sistema. Los modelos independientes de computación (*computation-independent models* o CIM) describen el negocio sin preocuparse de cómo se implementará el sistema. Los modelos independientes de la plataforma (*platform-independent models* o PIM) dicen cómo cumplir los requisitos de negocio con el sistema, pero sin entrar en detalles de cómo se implementará en una plataforma software y hardware determinada. Por último, los modelos específicos de la plataforma (*platform-specific* models o PSM) completan el resto de los detalles, para así simplificar la traducción final a código.

METODOLOGÍAS PARA ARQUITECTURAS ORIENTADAS A SERVICIOS

En esta sección realizaremos una revisión de algunas de las metodologías disponibles para desarrollar sistemas de información basados en arquitecturas orientadas a servicios. Descartaremos aquellas especificaciones y metodologías que sólo cubren pasos específicos del proceso de construcción del software, como la BPM (*Business Process Modeling Notation*) [9], que se ocupa únicamente de modelar los procesos de negocio, sin entrar en cómo se implementarán.

ANTECEDENTES EN EL DESARROLLO BASADO EN COMPONENTES

Ya se ha mencionado anteriormente que SOA no es una revolución, sino una evolución de las lecciones aprendidas en el campo de la Ingeniería del Software. Pueden establecerse varios paralelismos entre SOA y el paso conceptual inmediatamente anterior: desarrollo basado en componentes (*component based development* o CBD) [3]. Existe una gran variedad de definiciones del concepto de "componente", pero en general podemos decir que se trata de una "caja negra" con entidad propia de la que sabemos *qué* servicio provee, pero no *cómo* lo hace, y que podemos ensamblar con otros componentes. La principal diferencia entre SOA y CBD es cómo se explotan esos servicios: con CBD, el componente pasa a formar parte de nuestro programa, mientras que con SOA sigue siendo un ente independiente, con el que nos comunicamos mediante mensajes. Además, un servicio de SOA ha de implementar una funcionalidad a nivel de negocio, mientras que un componente no tiene por qué, pudiendo ser algo más sencillo.

METODOLOGÍA SOMA

La metodología SOMA (Service Oriented Modeling and Architecture), desarrollada por IBM [4], define un enfoque integrado sobre el proceso de desarrollo SOA que cubre desde su concepción hasta su monitorización y mantenimiento. Consiste en un ciclo iterativo de refinamiento dividido en varias fases.

En primer lugar, se define un modelo del negocio y una serie de plantillas para los distintos tipos de soluciones de integración posibles. A continuación se identifican los servicios que formarán parte de la arquitectura, combinando varias fuentes: las metas de la empresa, un modelo conceptual del entorno, y los sistemas informáticos ya implantados.

Posteriormente se estructurarán, racionalizarán y especificarán todos los servicios especificados en una arquitectura coherente. Puede que se haya identificado un gran número de servicios, por lo que se seleccionará el subconjunto que reporte el mayor retorno en inversión y se pospondrá el resto. Finalmente, se implementarán, depurarán e implantarán los servicios, tras los cuales se someterán a monitorización.

La metodología se halla implementada como una serie de extensiones al Proceso Unificado de Rational (*Rational Unified Process* o RUP), y utiliza el estándar Lenguaje Unificado de Modelado (*Unified Modeling Language* o UML) [10] con algunas extensiones para modelado. Ello implica una serie de ventajas y desventajas: aunque se halla validada a través de múltiples proyectos y se apoya sobre un proceso conocido, se trata de una metodología compleja en la que se elabora un gran número de documentos intermedios y se utilizan numerosas herramientas. Por ello, esta metodología puede no ser la mejor opción para empresas pequeñas y medianas de fabricación, con menos recursos para realizar este tipo de modelado extensivo. Una metodología más ligera sería más efectiva en estos casos.

METODOLOGÍA SOD-M

La metodología SOD-M (Service Oriented Development Method) [5,11] es también dirigida por modelos y centrada en servicios. Cubre los tres puntos de vista del enfoque MDA del OMG con perspectivas sobre el negocio (nivel CIM) y sobre el sistema (niveles PIM y PSM). La perspectiva del negocio incluye un modelo de la organización y su entorno especificado sobre intercambios de valor [12], y un modelo de los procesos de negocio seguidos, utilizando diagramas UML de actividad. Estos diagramas son parecidos a los conocidos diagramas de flujo.

La definición de la perspectiva del sistema (nivel PIM), por otro lado, comienza por establecer modelos de casos de uso del sistema de información, que después se extienden y convierten a modelos de procesos de servicio del sistema. Estos modelos detallan las actividades necesarias para su realización y las relaciones con los demás procesos. Posteriormente, esas actividades se reparten entre los participantes en modelos de composición de servicio. Estos dos últimos tipos de modelos también usan diagramas de actividad de UML.

Finalmente, en el nivel PSM de la perspectiva del sistema se decide en el modelo de composición de servicio extendido qué actividades del modelo de composición de servicios serán servicios web, y se definen sus interfaces.

Esta metodología resulta ser más ligera que en el caso de SOMA, y algunas de las correspondencias entre los distintos modelos llegan a estar automatizadas parcialmente [11], a diferencia de otros métodos, como [3]. Puede verse un esquema de las relaciones entre los distintos modelos en la figura 1. Utiliza notaciones más sencillas, facilitando la comunicación entre clientes y desarrolladores. Sin embargo, a diferencia de SOMA, no incluye todos los pasos del proceso de desarrollo, y en particular no integra actividades de realización de pruebas sobre el software.

A pesar de ello, consideramos que esta metodología es una buena candidata sobre la que basarse para definir una metodología completa para el desarrollo de sistemas de información orientados a servicios, en la que se pueden representar de forma comprensible para las partes implicadas las prácticas actuales de negocio de la organización.

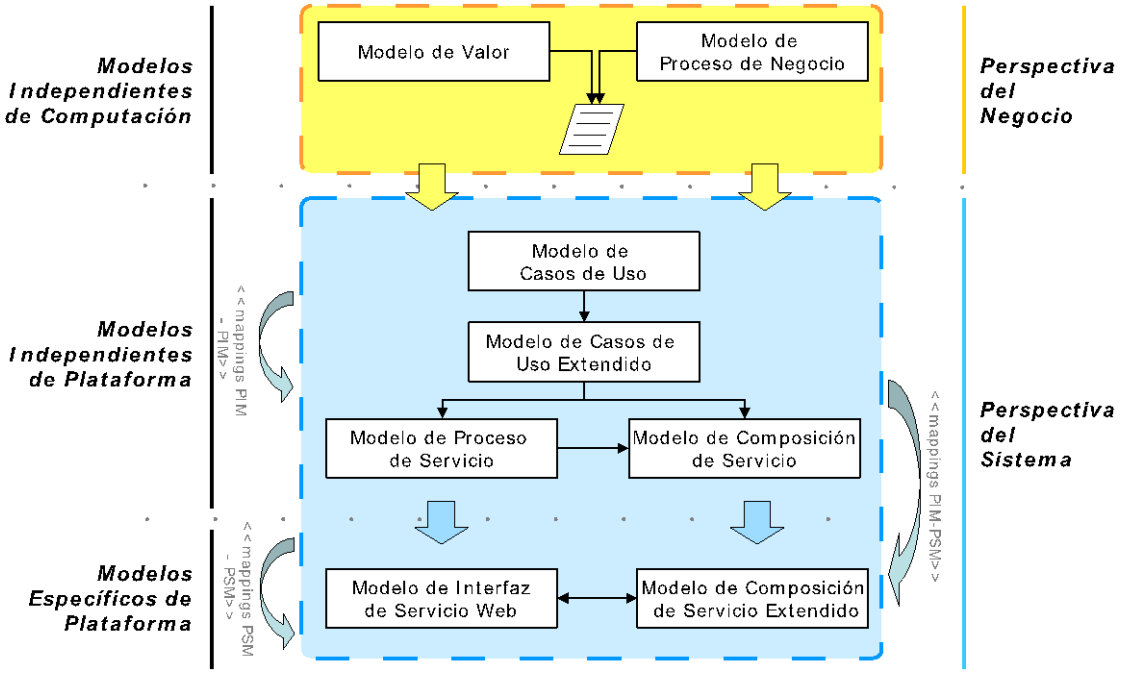

Figura 1. Esquema de los modelos utilizados en SOD-M

INTEGRACIÓN DE SOD-M CON MODELOS DE PRUEBAS

En la anterior sección revisamos algunas de las metodologías existentes e identificamos a SOD-M como una candidata viable de la que partir para definir una metodología SOA ligera completa. Sin embargo, pudimos comprobar una carencia grave en dicha metodología: no integraba procesos de prueba. En esta sección propondremos algunas mejoras sobre SOD-M para intentar cubrir dicha carencia y poder hacer pruebas sobre los sistemas de información.

TIPOS DE PRUEBAS A REALIZAR

A grandes rasgos, existen seis categorías de pruebas de sistemas de información [13]: pruebas de instalación, aceptación, sistema, función, integración y módulo. No consideraremos en este artículo las pruebas de instalación y módulo, ya que no requieren un tratamiento especial al trabajar con SOA.

Las pruebas de sistema cubrirían en el caso de SOA a la arquitectura completa, es decir, el ecosistema creado por todos los servicios expuestos y sus consumidores. Nos interesaría comprobar si nuestro sistema es incapaz en algún momento de cumplir ciertas propiedades deseables, como rendimiento (en transacciones/minuto), número máximo de accesos concurrentes, el volumen máximo de datos admitido, las restricciones de acceso impuestas, etc. Muchas veces, estas propiedades forman parte del contrato de un servicio (su *acuerdo de nivel de servicio* o Service Level Agreement).

Una prueba de función se referiría a un servicio de negocio, que normalmente reúne una serie de servicios web en uno solo de mayor nivel (utilizando, por ejemplo, el lenguaje *Web Service Business Process Execution Language* o WS-BPEL [14]). En este nivel nos interesa saber si se implementa la funcionalidad deseada y se cumplen el resto de restricciones (fijándonos en las de las pruebas de sistema).

Por último, una prueba de integración trata con un solo servicio web, que habrá sido implementado como un ensamblaje de varios módulos. Las comprobaciones a hacer son similares, salvo por el hecho de que en este caso trataremos normalmente con un programa normal escrito en algún lenguaje de programación de uso general, en vez de utilizar lenguajes específicos para componer servicios, como WS-BPEL.

EXTENSIONES PROPUESTAS

Las pruebas de aceptación pueden ayudarse de las relaciones de trazabilidad entre los modelos de los niveles CIM y PIM. Si nos aseguramos de que todas las actividades deseadas de cada proceso de negocio estén implementadas, y verificamos dichas relaciones hasta el nivel de servicio web, tendremos altas garantías de que se cumplen los requisitos acordados. No es necesario modificar el modelo, sino sólo las herramientas que lo manipulan.

Para las pruebas de sistema, proponemos anotar los modelos de procesos de servicio del nivel PIM con información acerca del nivel de servicio esperado (rendimiento, tiempo de respuesta, etc.) y las restricciones de seguridad. Dichas anotaciones se harían a nivel global y posteriormente se extenderían de forma local a cada una de las actividades, en función de la estructura de cada proceso. Dado que una actividad determinada puede formar parte de varios procesos de

negocio, se podrían agregar los requisitos individuales para obtener los requisitos específicos de cada servicio web que fuera a ser expuesto, consiguiendo información ya a nivel PSM. A partir de estos requisitos particulares se podrían generar casos de prueba para ver si en algunos momentos no se cumplen.

En la figura 2 en la página siguiente podemos ver un ejemplo de un modelo de proceso de servicio para atender a un pedido proveniente de un cliente. Se ha decorado el modelo mediante una serie de comentario, indicando para las actividades el número mínimo de transacciones por segundo que han de poder ser procesadas y el tiempo límite para cada una de ellas. Para las transiciones condicionales, se ha indicado una estimación de la probabilidad de que se tome cada rama. En función de la información especificada por el diseñador (distinguida con comentarios con un fondo coloreado), se podría derivar nueva información, como puede verse en los comentarios transparentes. Por ejemplo, si la probabilidad de aceptar el pedido es de 0,8, cada uno de los flujos concurrentes ha de poder procesar $0{,}8 \cdot 5 = 4$ transacciones por segundo.

Para realizar las pruebas funcionales y de integración podemos o bien generar una serie de casos de prueba con entradas y las salidas esperadas o hacer comprobaciones sobre el código final. En ambos casos, nos será de utilidad tener modelos para ver qué casos definir o qué comprobaciones hacer. Necesitamos saber, a nivel abstracto, qué ha de hacer cada proceso de servicio y cada servicio web expuesto. Proponemos anotar las composiciones en su conjunto de modelos de composición de servicio y las actividades marcadas como servicio web en los modelos de composición extendidos mediante condiciones lógicas [15] que relacionen las entradas, salidas y los cambios sobre la información almacenada. SOD-M en sí no dispone de modelos para representar la información almacenada en el sistema, pero sus autores ya lo han tenido en cuenta al integrarlo con la metodología para sistemas de información via Web MIDAS [5], definiendo un modelo conceptual de datos a nivel PIM.

En la figura 3 en la página siguiente puede verse un ejemplo de modelo de composición de servicio derivado del anterior modelo de proceso de negocio. Se han indicado con los estereotipos estándar <<precondition>> y <<postcondition>> de UML las condiciones a cumplir al inicio y final de la ejecución de la composición, y con <<localPrecondition>> y <<localPostcondition>> algunas de las condiciones sobre el servicio web “Hacer factura”. Especificamos que para dicho servicio web, inicialmente el pedido debe de hallarse abierto, aceptado, no vacío, y no debe tener ya una factura asociada. Tras su ejecución, debe haberse creado una factura nueva con los artículos, precios y valor total correctos, que se enviará a la acción “Realizar pago” para posterior procesamiento.

Una vez disponemos de especificaciones del comportamiento de cada proceso de servicio y servicio web, podemos aprovecharlas para integrar diversas técnicas de prueba. Por ejemplo, se podría derivar un grafo causa-efecto [13,16] del que extraer casos de prueba de forma semiautomática, o cualquier otro modelo dirigido a pruebas, como el de [17].

No basta con generar muchos casos de prueba, de todas formas: hay que ver si el conjunto tiene la suficiente calidad. Algunas de estas comprobaciones pueden hacerse sobre la especificación (es decir, nuestros modelos), y otras sobre los programas ya implementados [18]. Por ejemplo, [19] permite encontrar posibles fallos de programación que los casos de prueba no detectarían, y [20] identifica propiedades que pueden diferir de las esperadas de una composición. Otras posibilidades incluyen el cálculo del porcentaje de instrucciones ejercitados por varios casos de prueba, por ejemplo.

Posteriormente, tras ejecutar las pruebas sobre la versión final en código ejecutable de los servicios y composiciones y localizar fallos y posibles carencias en los casos de prueba, podríamos revisar nuestros modelos intermedios para transportar nuestros nuevos conocimientos y casos de prueba a otra plataforma, o volver a ejecutar las pruebas tras hacer algún cambio, para asegurarnos de que todo sigue funcionando.

CONCLUSIONES Y TRABAJO FUTURO

Las empresas de fabricación requieren, para mantenerse competitivas, integrar las últimas tecnologías en procesos de fabricación y revisar sus prácticas de negocio de acuerdo a la demanda del mercado. Para ello, la empresa pasa de ser un único ente centralizado y jerarquizado a ser un conjunto de elementos interrelacionados de forma dinámica, integrando a proveedores, clientes, diseñadores, subcontratistas y otros participantes en lo que conocemos como una empresa extendida. La importancia de este hecho puede verse en la aparición de modelos de empresa distribuida en el campo de la Ingeniería de Fabricación, como son las empresas holónicas, fractales y biónicas.

Creemos que una empresa extendida estructurada de tal forma debería tener un sistema de información que acompañara a esta visión, siguiendo una arquitectura orientada a servicios, en la que el sistema completo se estructura como un ecosistema de servicios, proveedores y consumidores, de tal forma que su libre reconfiguración permita la rápida revisión de las prácticas de negocio y la integración con sistemas externos. Sin embargo, no puede hacerse a la ligera: se trata de un esfuerzo a realizar por toda la organización, y un fallo en un servicio ampliamente reutilizado podría tener graves consecuencias. Por esta razón, se hace evidente la necesidad de seguir una metodología bien definida que facilite la comunicación y garantice el desarrollo de una arquitectura cuyos servicios cumplan los requisitos impuestos.

Comenzaremos por integrar las técnicas que puedan reportar los beneficios más inmediatos, como es el caso de las modificaciones a los modelos de proceso de servicio para integrar requisitos sobre el tiempo de respuesta, el rendimiento esperado, y las distribuciones de probabilidad de las ramas condicionales. Estas extensiones requerirán la extensión de los metamodelos especificados en la metodología SOD-M, y la implementación automática del mecanismo de inferencia en las herramientas de modelado escogidas.

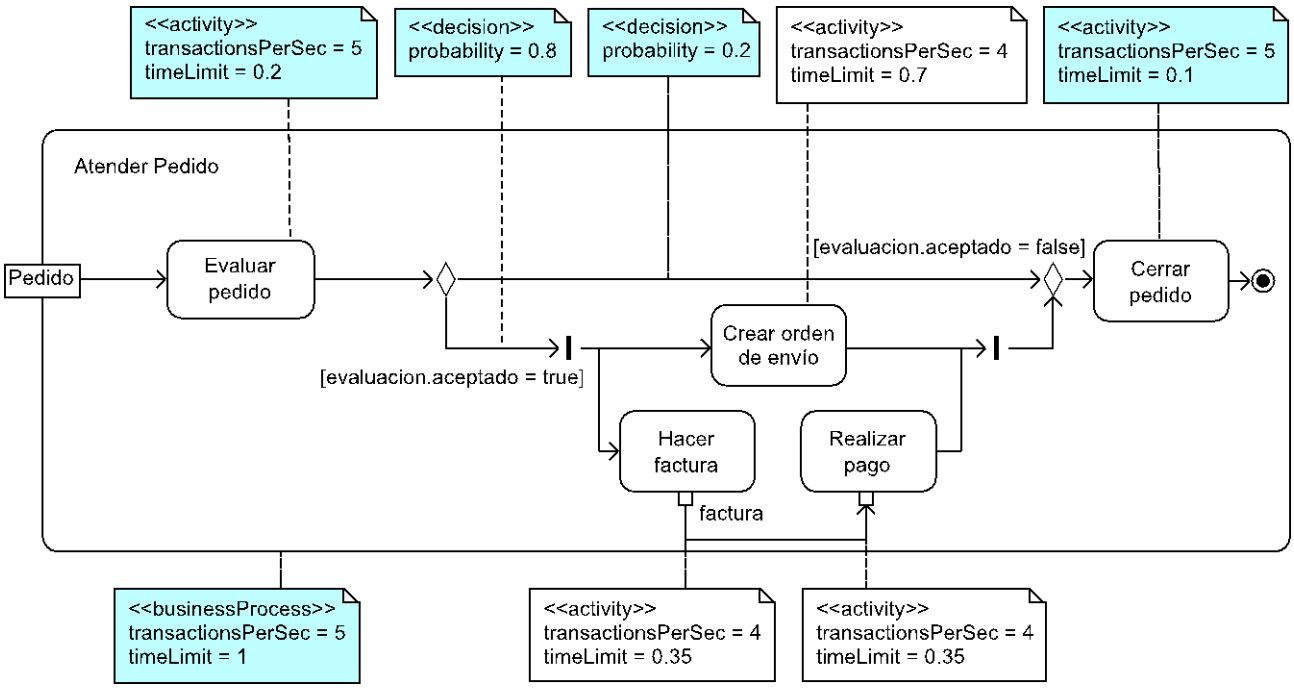

Figura 2. Modelo de proceso de servicio decorado con restricciones de nivel de servicio explícitas y derivadas

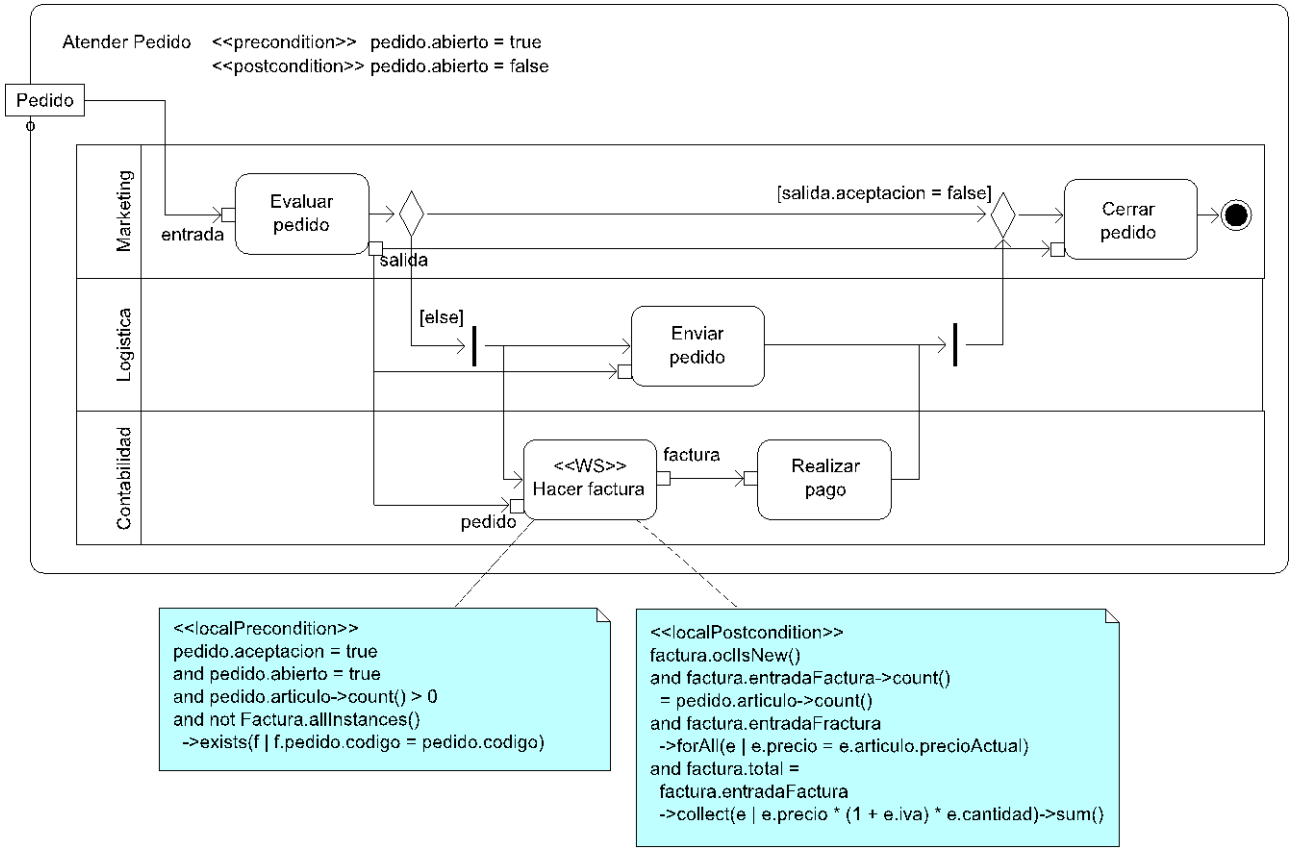

Figura 3. Ejemplo de modelo de composición de servicio decorado con restricciones OCL

Continuaremos seleccionando una notación ([15] es una primera posibilidad) para especificar el comportamiento de cada modelo de composición de servicio en base a las relaciones entre sus entradas y salidas y los cambios sobre los datos almacenados. Estudiaremos cómo utilizarla para generar casos de prueba y realizar comprobaciones sobre una implementación de dicha composición utilizando lenguajes específicos de dominio como WS-BPEL [14].

Finalmente, extenderemos este análisis para los servicios web individuales e integraremos técnicas para evaluar la calidad de los casos de prueba generados. Para ello, podemos aplicar combinar diversos criterios de cobertura de la estructura del programa con herramientas de análisis más avanzadas, como [20], que puedan tener en cuenta en mayor grado la funcionalidad del programa, más que únicamente su estructura.

REFERENCIAS

[1] M. Marcos, F. Aguayo, M. Sánchez Carrilero, L. Sevilla, y J.R. Lama, *Toward the Next Generation of Manufacturing Systems. Frabiho: a Synthesis Model for Distributed Manufacturing*, Proceedings of the First I*proms Virtual Conference, Elsevier (2005), págs. 35-40.

[2] A. Tharumarajah, A. Wells, y L. Nemes, *Comparison of emerging manufacturing concepts*, Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics, CA, USA (1998), págs. 325-331.

[3] Z. Stojanović, *A Method for Component-Based and Service-Oriented Software Systems Engineering*. Tesis doctoral, Delft University of Technology (2005).

[4] S.G. A. Arsanjani y A. Allam, *SOMA: a method for developing service-oriented solutions*, IBM Systems Journal 47 (2008), págs. 377-396.

[5] M.V. de Castro, *Aproximación MDA para el desarrollo orientado a servicios de sistemas de información web: del modelo de negocio al modelo de composición de servicios web*. Tesis doctoral, Universidad Rey Juan Carlos (2007).

[6] J. Browne, I. Hunt, y J. Zhang, *The Extended Enterprise (EE)*, Intelligent Systems for Manufacturing: Multi-Agent Systems and Virtual Organizations, L.M. Camarinha-Matos, H. Afsarmanes, y V. Merik, eds., Kluwer Academic Publishers, Londres (1998), págs. 3-30.

[7] T. Erl, *SOA: Principles of Service Design*, Prentice Hall, Indiana, EEUU (2008), ISBN 0-13-234482-3.

[8] Object Management Group, *MDA Guide version 1.0.1* (junio 2003), véase: http://www.omg.org/mda/.

[9] Object Management Group, *Business Process Modeling Notation 1.2* (enero 2009), véase: http://www.omg.org/spec/BPMN/1.2/.

[10] Object Management Group, *Unified Modeling Language 2.1.2* (noviembre 2007), véase: http://www.omg.org/technology/documents/modeling_spec_catalog.htm.

[11] J.M. Vara Mesa, E. Marcos, y M.V. de Castro, *Obteniendo modelos de sistemas información a partir de modelos de negocios de alto nivel: un enfoque dirigido por modelos*, Actas de las IV Jornadas Científico-Técnicas en Servicios Web y SOA, Sevilla, España (2008), págs. 15-28.

[12] J. Gordijn, *Value-based requirements engineering: Exploring Innovative e-Commerce Ideas*. Tesis doctoral, Vrije Universiteit (2002).

[13] G.J. Myers, *The Art of Software Testing*, John Wiley & Sons, segunda edición (2004), ISBN 0471469122.

[14] OASIS, *WS-BPEL 2.0 Standard* (abril 2007), véase: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html.

[15] Object Management Group, *Object Constraint Language Specification 2.0* (mayo 2006), véase: http://www.omg.org/technology/documents/formal/ocl.htm.

[16] A. Paradkar, M.A. Vouk, y K.C. Tai, *Specification-based testing using cause-effect graphs*, Annals of Software Engineering 4 (enero 1997), págs. 133-157.

[17] Y. Zheng, J. Zhou, y P. Krause, *An Automatic Test Case Generation Framework for Web Services*, Journal of Software 2 (septiembre 2007), págs. 64-77.

[18] H. Zhu, P. Hall, y J. May, *Software Unit Test Coverage and Adequacy*, ACM Computing Surveys 29 (diciembre 1997), págs. 366–427.

[19] J.J. Domínguez Jiménez, I. Medina Bulo, y A. Botaro Estero, *A framework for mutant genetic generation for WS-BPEL*, Actas de la 35th Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2008), Spindleruv Mlýn, República Checa (2008).

[20] M. Palomo-Duarte, A. García-Domínguez, y I. Medina-Bulo, *Improving Takuan to analyze a meta-search engine WS-BPEL composition*, Proceedings of the 4th IEEE International Symposium on Service-Oriented System Engineering, Jhongli, Taiwan (2008).