El patrón de diseño MVC integrado con los modelos de procesos

Lic. José Martinez Garro [2], Mg. Patricia Bazán[1]
[1] *LINTI (Laboratorio de Investigación en Nuevas Tecnologías Informáticas) - Facultad de Informática – UNLP* [2] *Facultad de Informática* – UNLP - Argentina

Resumen - **La gestión de los procesos de negocio o BPM (*Business Process Management*) es una metodología empresarial que permite representar las actividades de los procesos de negocios y las restricciones que se aplican sobre ellas. Automatizar esta tarea conlleva el beneficio de contar con un soporte que modela, simula y monitorea las actividades de dichos procesos de negocio.**
Actualmente, uno de los intereses fundamentales en esta metodología radica en el hecho de desarrollar aplicaciones reales orientadas a usuarios finales que se valgan del paradigma de procesos. En un esquema de integración con recursos existentes y en pos de agilizar los tiempos de desarrollo proponemos en el siguiente trabajo una metodología para la integración de procesos en una arquitectura MVC, donde los procesos se integran con las aplicaciones existentes cumpliendo el rol de modelo de negocio.

Temas clave: **Proceso, MVC, modelo de proceso, servicio, metodología, BPM.**

I. INTRODUCCIÓN

Un proceso de negocio es una actividad del mundo empresarial que consta de un conjunto de tareas lógicamente relacionadas, que cuando se realizan en la secuencia apropiada y siguiendo las reglas del negocio, producen una salida válida para el negocio. Por ejemplo, realizar una transacción bancaria [1]

Actualmente, los procesos de negocios se están tornando ineficientes y obsoletos básicamente por su incapacidad de adaptarse a los cambios. Por ende, la necesidad de modelizar y optimizarlos es cada vez más importante para las organizaciones [2]

El cambio de un proceso de negocios involucra examinar el mismo con el objetivo de reducir el número de actividades, eliminando las tareas de menor importancia y simplificando el proceso en general. Con el objetivo de poder introducir cambios, es necesario adoptar un enfoque que permita rediseñar los procesos de negocios. Típicamente, las áreas problemáticas de un cambio en los procesos, sólo pueden ser identificadas una vez que el mismo ha sido físicamente implementado. Por lo tanto, tener la capacidad de visualizar y evaluar un cambio en los procesos antes de su implementación puede tener un impacto positivo sobre la tasa de éxito de los futuros cambios. Una forma de lograr esto es utilizar modelado dinámico de procesos de negocios [2]

Se llama Business Process Management (BPM) a la metodología empresarial cuyo objetivo es mejorar la eficiencia a través de la gestión sistemática de los procesos de negocio que se deben modelar, automatizar, integrar, monitorear y optimizar de forma continua. [3]

BPMS (BPM Suite) es el conjunto de servicios y herramientas informáticas que facilitan la administración de procesos de negocio.

Existen distintos patrones de diseño que permiten configurar modelos de aplicaciones. En particular, han tenido bastante auge en la actualidad aquellos patrones que responden a metodologías orientadas a servicios. Estos patrones suelen implementar el patrón de diseño MVC (*Model View Controller*).

Este trabajo plantea una metodología de trabajo para integrar procesos de negocio con aplicaciones reales, de manera tal de lograr que los usuarios finales interactúen con los procesos sin restricciones de funcionalidad o amigabilidad. De esta manera, nos interesa en particular adaptar la metodología BPM a un esquema de integración MVC con orientación a servicios.

En la sección 2 enunciaremos las etapas de la metodología de desarrollo basada en el ciclo de vida de procesos propuesto en [1]. En la sección 3 presentamos cómo integrar el patrón de diseño MVC a los procesos ya modelados. Por último encontraremos las conclusiones y las propuestas para futuras líneas de investigación relacionadas con el presente tema.

II. DESARROLLO DE APLICACIONES CON ORIENTACIÓN A PROCESOS

Las organizaciones pueden encontrar en la orientación a procesos, y en particular en BPM un mecanismo para alinear cada una de sus actividades con los objetivos de negocio. Esto se ve de alguna manera asegurado por la

actividad interdisciplinaria planteada por BPM y por la toma de conocimiento acerca de cada una de las etapas y recursos involucrados en las distintas actividades organizacionales.

Se presenta entonces casi en forma obligada la necesidad de elaborar una metodología de trabajo que permita resolver los problemas encontrados en la orientación de procesos a través de varias etapas de refinamiento. De esta manera, para desarrollar una aplicación con orientación a procesos debemos considerar una serie de fases o etapas que deben llevarse a cabo:

- **Estrategia y organización:** es independiente de una instancia particular de proceso, porque se relaciona con la identificación de la estrategia de negocio, y sus objetivos asociados.
- **Estudio:** es la primera fase relevante para los procesos individuales, y para post-proyectos que implementan a los mismos. Aquí se definen los objetivos del proyecto, se establece el equipo para el mismo, y se reúne información para el ambiente.
- **Diseño:** se reúne la información obtenida y se la analiza, consolida y representa a través de los modelos de procesos. Dichos modelos serán una base de comunicación entre los distintos interesados.
- **Selección de la plataforma:** se utiliza la información sobre el ambiente organizacional y técnico para decidir la plataforma en la cual los procesos se representarán.
- **Implementación y test:** los modelos desarrollados anteriormente se vuelven aquí procesos ejecutables. Se considera el desarrollo de prototipos, y se procuran mejoras mediante interacciones con empleados con conocimiento del dominio. También es necesario considerar aspectos no funcionales, como optimizaciones de rendimiento **y** robustez. Sobre esta etapa consideraremos a posteriori la integración con el patrón MVC.
- **Despliegue o promulgación:** se despliega la implementación lograda en el ambiente de destino. Se deben tener en cuenta aspectos técnicos, para asegurar el correcto funcionamiento en producción.
- **Operación y control:** la aplicación ya está en ejecución en el ambiente, de manera tal que se hace necesario reunir información propia de la ejecución y utilizarla para realizar mejoras evolutivas.

Debemos tener en cuenta que el orden de estas fases puede variar, y se pueden dar dependencias entre cada una ellas. Además, la metodología tiene un enfoque iterativo e incremental, donde cada iteración produce resultados que retroalimentan a las siguientes [1] [4] [5] [6].

La estrategia que fija la organización para alcanzar los objetivos de negocio permite mejorar la competitividad a largo plazo de una manera sustentable, donde se tienda a minimizar los impactos en la infraestructura de la organización que puedan provocar los cambios en el mercado. Dicha estrategia es el hilo conductor en todo el ciclo de vida del proceso.

A. El patrón MVC aplicado procesos de negocio

En este punto contamos ya con el proceso completamente diseñado, habiendo de esta manera concluido con el análisis de los componentes que serán diseñados e implementados desde cero, además de los que serán reutilizados por ser preexistentes. También se ha podido delimitar con claridad los escenarios de uso del sistema y los actores que intervendrán en los mismos. De esta manera no es lo mismo desarrollar procesos que serán accedidos por usuarios finales, por ejemplo, a través de un sitio web, que procesos que serán propios de interacción con usuarios internos que podrían estar familiarizados con otros tipos de ambientes de ejecución.

El patrón MVC es un patrón de diseño de software en el cual toda la arquitectura está dividida en 3 capas. Típicamente estas capas son el Modelo, la Vista y el Controlador (figura 1).

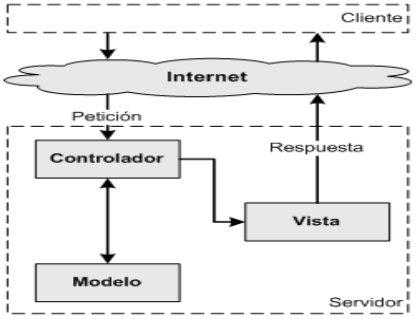

Fig. 1. Patrón MVC

El Modelo incorpora la capa del dominio y persistencia, es la encargada de guardar los datos en un medio persistente (ya sea una base de datos, un archivo de texto, XML, registro, etc.). En el Modelo es donde se hace el levantamiento de todos los objetos que el sistema debe de utilizar, siendo así el proveedor de recursos. En general en la actualidad la capa modelo no accede directamente a la base de datos, sino que interactúa con otra capa de acceso a datos (comúnmente denominada DAO, *Data Access Object*, lo cual permite acceder a la fuente de datos en forma transparente,

facilitando futuras actualizaciones o migraciones que pudieran existir).

La Vista se encarga de presentar la interfaz al usuario. En sistemas web, esto es típicamente HTML, aunque pueden existir otro tipo de vistas, dependiendo de la plataforma de desarrollo. En la vista sólo se deben de hacer operaciones simples, como bifurcaciones condicionales, ciclos, formateo, etc, delegando de esta manera el grueso de la funcionalidad en la capa modelo de la aplicación.

El Controlador es el que escucha los cambios en la vista y se los envía al modelo, el cual regresa los datos a la vista. Es un ciclo donde cada acción del usuario causa que se inicie una nueva interacción.

El patrón MVC permite modelar aplicaciones con una óptica de orientación a servicios, y por esto es aplicable a un gran espectro de casos. Existen en la industria numerosos desarrollos basados en esta óptica, y los mismos en general presentan buenos niveles de reutilización. Igualmente, a pesar de lo alentador en los pronósticos, no es trivial realizar un análisis de los posibles escenarios de ejecución del sistema, y de esta manera especificar a ciencia cierta si realmente el patrón se adapta a nuestra solución [5].

A la luz de lo expuesto, iniciamos un análisis detallado de cómo integrar el patrón MVC a un modelo de procesos existente. Comencemos por el componente más importante de la arquitectura: **el modelo**. Desde un comienzo hemos dejado en claro que nos interesa modelar el comportamiento de nuestra organización a través de procesos mediante BPM, puesto que esta metodología nos asegura que el proceso refleja el comportamiento integrado de las distintas áreas de la organización, y que además será capaz de absorber con velocidad los cambios impuestos por el mercado, alineado con los objetivos de negocio.

Así, desarrollaremos un proceso implementado en uno o varios lenguajes de tipo estándar (el formato de ejecución del proceso debería ser código BPEL, y los componentes podrían ser o bien *web services*, o bien componentes implementados en lenguajes ampliamente aceptados como Java, PHP o .NET).

Paralelamente a dicho desarrollo nos encontramos con la necesidad de modelar los datos. Estos se encargan de representar los flujos de información que se dan entre las actividades del proceso, como así también cualquier otra interacción del proceso con bases de datos externas a este. El conjunto de entradas y salidas del proceso pueden modelarse a través de un diccionario de datos con una notación estándar tipo XML, y luego, mediante una evolución de modelos, llegar a representar al mismo en una base de datos, por ejemplo del tipo relacional. De esta manera lograríamos representar los datos propios del proceso en un entorno persistente.

Vale recordar que tanto la base de datos representativa de las entradas y salidas, como cualquier otra requerida por el proceso serán accedidas a través de la capa DAO, que permitirá una abstracción del proceso con respecto al servidor de datos propiamente dicho.

Luego pasamos al componente de **tipo vista**, que se encargará de la representación visual de las interfaces necesarias para la interacción de los usuarios finales con el proceso. Será necesario considerar entonces aquellas entradas del diccionario que representan los flujos iniciales y finales del proceso, y plantearlas de un modo amigable al usuario. En general se puede utilizar un mecanismo similar a los EJB de Java, para poder tomar un archivo XML que representa datos propios del proceso, y generar a partir de este una vista HTML representativa del mismo. Para la generación de las vistas propiamente dichas se puede utilizar cualquier lenguaje que permita la generación de interfaces visuales (por ejemplo HTML, si se tratara de un entorno *web*) y que además permita la interacción con componentes de tipo servidor (tales son los casos de Java, .NET o PHP).

En esta instancia se considerarán criterios de presentación y amigabilidad ampliamente aceptados, como son, en el caso de las interfaces *web*, el uso de validaciones en *Javascript*, o la utilización de listas desplegables para los campos de dominio taxativo. Muchos BPMS generan automáticamente interfaces para la interacción con los procesos, pero en la mayoría de los mismos no se observan posibilidades de modificación y/o consideraciones de amigabilidad mínimas. De allí la necesidad de plantear el MVC como una alternativa válida para la interacción con los procesos.

Ahora bien, habiendo desarrollado el proceso de negocio que nos servirá como modelo, y las vistas que permitirán la interacción con los usuarios finales, necesitamos un componente que nos permita la interacción de ambos. Este rol es cubierto por el **Controlador**. El mismo deberá acceder al proceso, lo cual puede ser efectuado de dos maneras:

- una primera aproximación, la más simple de implementar, sería con el modo librería: el proceso es compilado y se lo utiliza como una librería por parte del controlador. Esto tiene como desventaja que cualquier cambio en el proceso no es trivial, ya que será necesario recompilarlo y actualizarlo en el controlador para que este funcione con normalidad.

- La otra alternativa, de mucho mayor calidad, será la de crear un *stub* (un objeto intermediario que representa al proceso dentro del controlador) que permite interactuar con el proceso, el cual ha sido previamente publicado como un *web service*. Esto aporta mayor reusabilidad, ya que el proceso puede ser accedido por cualquier aplicación que tenga acceso al *bus* de servicios, como así también mayor transparencia en la actualización, ya que el proceso puede ser modificado o incluso relocalizado sin tener esto impacto sobre las aplicaciones que lo invocan.

De esta manera, nuestra aplicación resultaría de los siguientes componentes ilustrados en la figura 3: un proceso implementado en código BPEL que se ejecuta sobre un servidor de procesos, y que cumple el rol de modelo, accediendo a los componentes de datos a través de una capa independiente denominada DAO. Por otro lado, una aplicación *web* que puede ejecutarse sobre un servidor genérico de aplicaciones y que presenta las interfaces necesarias para que nuestro proceso sea accedido por usuarios finales. Por último, un componente de tipo controlador implementado preferentemente en una tecnología compatible con las vistas, y que recibe los requerimientos generados a través de las interfaces de usuario, y los envía al proceso mediante un *stub* surgido de la publicación del proceso como un *web service*. Luego al recibir los resultados, el controlador los retorna al componente de tipo vista y este los despliega para su visualización. [6][7][8]

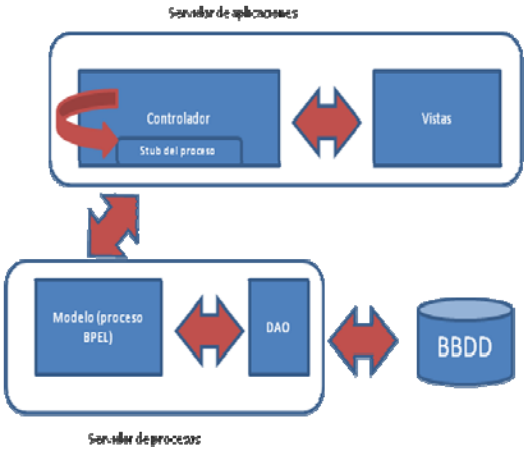

Figura 3. Gráfico de la arquitectura

El aspecto más importante de esta integración se da en el hecho de incluir los procesos propios de BPM a una arquitectura probada y con buenos resultados dentro de la industria, tal como lo es el patrón MVC. Uno de los riesgos que corren las organizaciones al adoptar una determinada metodología es adoptar el criterio del "todo o nada", aplicándolo a todo su espectro de problemas, o siguiendo en la inercia de la ausencia de actualización. Aquí estamos planteando un punto intermedio donde presentamos mecanismos innovadores propios de BPM, integrados con un patrón de diseño propio de aplicaciones tradicionales del esquema cliente servidor.

B. Conclusiones

BPM constituye en la actualidad la mejor aproximación para lograr alinear los procesos de negocio con los objetivos de la organización, y permitir así que los requerimientos del mercado se absorban con mayor velocidad por parte de los sistemas de información presentes en la organización. Esto se debe fundamentalmente a la acción interdisciplinaria que propone intrínsecamente la metodología, donde los procesos son diseñados, implementados y ejecutados por áreas distintas de la organización.

Uno de los mayores desafíos que encuentra BPM es la necesidad de nivelar la generación de nuevos componentes informáticos junto con la capacidad de reutilizar elementos existentes. Aquí, el tópico facilitador es SOA, ya que los servicios permitirán la reutilización de componentes existentes, como así también la portabilidad y escalabilidad de los nuevos desarrollos. La integración de componentes existentes en la organización es una de las capacidades fundamentales, ya que de otra manera las organizaciones se verían obligadas a desechar aplicaciones que les son funcionales y que han representado fuertes inversiones. Así, encontrando un equilibrio entre actualizaciones y nuevos desarrollos, con integración de recursos preexistentes se favorece a la absorción de BPM en organizaciones de nichos disímiles.

Durante el ciclo de vida de los procesos es necesario utilizar los BPMS para la gestión de *workflow*, aunque estos no siempre logran adaptarse a la diversidad de casos de usos posibles. Si necesitamos una interacción de usuarios finales con los procesos, entonces las interfaces autogeneradas por los mismos no son la mejor opción, ya que las mismas no cumplen con requisitos de amigabilidad y funcionalidad. Además, requerimos que las aplicaciones a desarrollar cumplan con normas de estandarización actuales y tengan la capacidad de reutilizar recursos preexistentes. Por esto planteamos una metodología de desarrollo que cubra las distintas fases en el ciclo de vida de un proceso, haciendo hincapié en la fase de desarrollo.

En esta última fase vemos la necesidad de crear aplicaciones que interactúen con los procesos, aunque no de manera caótica sino

ordenada. MVC es un patrón de diseño que conforma arquitecturas altamente probadas en la industria, y con muy buenos indicadores de adaptabilidad y escalabilidad. Para adaptar dicho patrón de diseño a nuestro entorno de procesos, hemos propuesto una aproximación que permite considerar a los procesos de negocio como el componente modelo que interactúa con los datos y que concentra el grueso de la funcionalidad. En si mismos, los procesos son el núcleo que nuestra arquitectura, y su funcionamiento es independiente de las aplicaciones que deseen accederlos en forma externa. Por otro lado tendremos los componentes de tipo vista independientes del modelo y que se encargan de la lógica de visualización. A estos componentes les pediremos que cumplan con aspectos funcionales y de amigabilidad que no poseen las interfaces autogeneradas por muchos de los BPMS actuales. En el medio, se encuentra el controlador que hace las veces de puente entre el modelo y las vistas, y cumple el rol de "broker" en una analogía con arquitecturas orientadas a servicios, donde el servicio principal en este caso está dado por el proceso de negocio.

En la actualidad se observa una tendencia dentro de las herramientas que permiten gestionar BPM a concentrar el desarrollo de los procesos en dos fases:

- la primera y fundamental en el diseño del proceso propiamente dicho, frecuentemente con la notación estándar BPMN, más algún otro elemento de diseño. Es el proceso quien se encargará de acceder a las bases de datos necesarias. Hasta aquí el componente modelo.
- La segunda radica en hacer un diseño meticuloso de las interfaces de usuario, las cuáles constituyen el componente de tipo vista. La tendencia es desarrollarlas para plataformas web y que las mismas cumplan con requisitos de validaciones y representación estándares en la actualidad.

Estas últimas dos tareas corren por cuenta del equipo de desarrollo.

Una vez que tenemos los procesos implementados y las interfaces que representarán los formularios para el ingreso de los insumos necesarios para los primeros, el controlador suele ser implementado en forma automática por la plataforma del BPMS. Esto es, el entorno genera automáticamente los mecanismos que asocian los procesos con sus interfaces para lograr la ejecución. Esto tiene la ventaja obvia del ahorro en tiempos de desarrollo y la disminución de errores. Por otro lado la desventaja que implica que el proceso no es completamente independiente de las vistas, ya que la unión entre ambos no es configurable al ser autogenerada.

Resta ver de esta manera la evolución de los BPMS actuales en la gestión de procesos integrados con una arquitectura MVC.

C. LÍNEAS DE INVESTIGACIÓN FUTURAS

Como posibles líneas futuras se plantean, en primer lugar la necesidad de indagar aún más en la generación del diccionario de datos del proceso, y la evolución del mismo en un modelo persistente. Esto debe apuntar a no abstraer en forma total la funcionalidad del proceso de los datos que le sirven de insumo al mismo. Esto apunta fundamentalmente a la capacidad de autogenerar la base de datos del proceso mediante el modelo de flujo de datos entre tareas.

En la misma línea se encuentran entonces la generación de interfaces de proceso relacionadas con las entradas y salidas generadas por este último, tratando que las mismas estén destinadas a usuarios finales bajo normas de amigabilidad y estandarización. Por otro lado, es necesario profundizar en el hecho que las aplicaciones que consumen el proceso como servicio generen información que permita luego efectuar actividades de minería tendientes a optimizar el funcionamiento del proceso, debido a que la ejecución del proceso excede el ambiente de propio del BPMS considerado. Esta última actividad tiende a aumentar las posibilidades de efectuar minería de procesos capaz de retroalimentar el desarrollo de procesos.

III. REFERENCIAS

[1] Mathias Weske – "BUSINESS PROCESS MANAGEMENT - Concepts, Languages,architectures". Springer-Verlag Berlin Heidelberg 2007.

[2] Judith Hurwitz, Robin Bloor, Carol Baroudi, Marcia Kaufman – "Service Oriented Architecture for dummies". Wiley Publishing Inc 2007.

[3] IBM Corporation – "An IBM Proof of Technology. Discovering the value of Websphere BPM for your organization". IBM Corporation 2008.

[4] Material didáctico del curso "SOA-BPM" organizado por SADIO. Noviembre 2008.

[5] Definición del patrón MVC http://java.sun.com/blueprints/patterns/MVC.html (al 27/07/2010)

[6] José Martinez Garro – "Análisis metodológico de la plataforma IBM Websphere BPM y sus equivalentes funcionales en herramientas de licenciamiento de código fuente abierto" – Tesina de Grado – Facultad de Informática UNLP 2010.

[7] Definiciones del patrón MVC http://web2development.blogspot.com/2007/05/patron-mvc.html (al 27/07/2010)

[8] Programación con MVC http://www.proactiva-calidad.com/java/patrones/mvc.html (al 27/07/2010)