Vol. VII, No. 3, septiembre - diciembre, 2016, pp. 54 - 63

Integración de motores de reglas utilizando la programación orientada a aspectos en el desarrollo de aplicaciones en Java

Juan Carlos García González
correo electrónico: jcgarcia@ceis.cujae.edu.cu
Universidad Tecnológica de La Habana José Antonio Echeverría, Cujae, La Habana, Cuba

Margarita André Ampuero
correo electrónico: mayi@ceis.cujae.edu.cu
Universidad Tecnológica de La Habana José Antonio Echeverría, Cujae, La Habana, Cuba

Artículo Original

Resumen
En varios enfoques de desarrollo de software la capa de lógica de negocio de las aplicaciones resulta ser compleja, debido a que debe implementar las reglas de negocio existentes en una organización. En algunos sistemas las reglas de negocio cambian con más frecuencia que otros elementos. El enfoque de reglas de negocio propone una vía de desarrollo de aplicaciones donde las reglas de negocio juegan un papel fundamental. Para implementar el enfoque de reglas se destacan dos vías fundamentales: el uso de motores de reglas y la programación orientada a aspectos. Este trabajo propone la integración de ambas variantes, buscando potenciar sus ventajas y minimizar las limitaciones. Con el propósito de validar la propuesta se definen seis medidas y se presenta un ejemplo basado en una tienda virtual. Se implementan las tres variantes en la solución del ejemplo, utilizando el motor de regla JBoss Rules, la herramienta AspectJ y la integración de JBoss Rules con AspectJ. Al ejecutar las variantes, utilizando las medidas propuestas, se demuestra que la integración resulta superior.

Palabras claves: enfoque basado en reglas de negocio, motores de reglas de negocio, programación orientada a aspectos, JBoss Rules, AspectJ

Recibido: 24 de junio del 2016 Aprobado: 22 de septiembre del 2016

INTRODUCCIÓN

En la actualidad ha aumentado la demanda de aplicaciones informáticas de corte empresarial. Las aplicaciones empresariales suelen estar compuestas por tres capas: presentación, lógica de negocio y persistencia o acceso a datos. La capa de lógica de negocio es encargada de implementar los principales procesos de negocio de las organizaciones. A menudo estas organizaciones están sujetas a numerosas reglas. Por lo tanto, las reglas de negocio juegan un papel protagónico como requisitos que debe cumplir la aplicación.

Una regla de negocio es una declaración que define u obliga un cierto aspecto del negocio. Con estas se pretende imponer la estructura del negocio o controlar su comportamiento [1]. Una segunda definición propuesta en [2] plantea que las reglas de negocio son definiciones explícitas que regulan cómo opera y se estructura un determinado negocio.

En varios enfoques de desarrollo de software las reglas del negocio identificadas se implementan dentro de la capa de lógica de negocio, en diferentes puntos del código fuente. El hecho que las reglas estén distribuidas por toda la capa de lógica de negocio implica dispersión del código que implementa las reglas. Esta dispersión dificulta el mantenimiento de las aplicaciones y afecta su adaptabilidad ante los cambios en los requisitos del negocio [3]. Para enfrentar esta situación se puede aplicar el enfoque de reglas de negocio, el cual propone una

serie de fases y pasos a seguir con el objetivo de obtener aplicaciones basadas en reglas en negocio [1] [2]. Estas aplicaciones separan las reglas de negocio lógicamente, y quizás físicamente, del núcleo funcional de lógica de negocio de la aplicación, con el propósito de reducir los tiempos de soporte y mantenimiento, así como obtener aplicaciones más adaptables a los frecuentes cambios en las reglas de negocio [4] [5].

El proceso de desarrollo de software propuesto por varios enfoques normalmente está compuesto por una serie, fases o etapas. En cada una de estas fases existe un grupo de pasos que, por lo general, resultan comunes entre los distintos enfoques existentes. En el enfoque de reglas de negocio se proponen seis fases: Definir el alcance del negocio, descubrimiento, análisis, diseño, implementación y prueba [1]. Para implementar el enfoque de reglas de negocio resulta factible utilizar tecnologías o herramientas que posibiliten una implementación basada en reglas de negocio. Actualmente, se destacan dos variantes de implementación, el uso de motores de reglas de negocio y el empleo de la programación orientada a aspectos [3] [6] [7].

La primera variante tecnológica propone utilizar un motor de reglas de negocio el cual se integra en la capa de lógica de negocio de las aplicaciones. Un motor de reglas es una aplicación que, a partir de una información inicial y un conjunto de reglas definidas en un repositorio, decide qué reglas deben aplicarse a una determinada colección de información, ejecuta las reglas requeridas, aplica y evalúa los cambios que esta ejecución tiene sobre la información suministrada [8] [9] [10] .

En diferentes trabajos se ha analizado la factibilidad de implementar el enfoque de reglas de negocio utilizando motores de reglas. En [9] [11] se presenta una solución utilizando un motor de reglas de negocio para implementar el enfoque de reglas de negocio en el desarrollo de aplicaciones sobre la plataforma empresarial de Java. En [10] se realiza un análisis de los resultados obtenidos al aplicar el enfoque de reglas de negocio en aplicaciones desarrolladas siguiendo el paradigma de arquitectura guiada por modelos. Por otro lado, en [12] se presentan los resultados obtenidos en la implementación de un sistema basado en reglas, para la detención de fallos en transformadores eléctricos. Sin embargo, en varios de estos trabajos se señala que al integrar un motor de reglas de negocio en una aplicación, aún persiste el inconveniente de que se genera un fenómeno de código disperso y entremezclado en varios puntos de la capa de lógica de negocio de la aplicación, ya que si bien las reglas de negocio se encuentran físicamente separadas del código de la funcionalidad principal, el código necesario para acoplar el motor de reglas y realizar la conexión de las mismas, se encuentra disperso dentro de la aplicación [9].

La otra variante tecnológica que se destaca en la implementación del enfoque de reglas de negocio consiste en utilizar el paradigma de la programación orientada a aspecto para extraer y encapsular las reglas de negocio dentro de aspectos y así, separarlas del código fuente que implementa la lógica de negocio en las aplicaciones. Un aspecto actúa como conector entre la lógica de negocio de la aplicación y la regla de negocio. La idea básica consiste en que dentro de cada aspecto se implementen una o varias reglas de negocio, buscando garantizar mantenibilidad y reusabilidad [12] [13]. Esta variante ha sido aplicada en diferentes trabajos [6] [10] [12] [14] [15] [16], los cuales resaltan la disminución del código entremezclado como el principal logro del empleo de AOP en la aplicación del enfoque de reglas de negocio. Sin embargo, los autores plantean que al desarrollar una aplicación utilizando AOP puede existir un aumento en la complejidad tecnológica ya que es preciso contar con el compilador del lenguaje (por ejemplo, Java) con el que se implementa el núcleo funcional de la aplicación y con el compilador del lenguaje AOP específico que permite implementar las reglas dentro de los aspectos (por ejemplo, AspectJ). Adicionalmente, en estos trabajos se señala como principal limitante que las reglas de negocio se implementan dentro de los aspectos en código fuente. Esto implica que los cambios en las reglas de negocio exigen la modificación del código fuente, lo que conlleva a realizar tareas como son: recompilar y desplegar la aplicación [16] [17].

Como se describe anteriormente, ambas variantes tecnológicas presentan fortalezas y limitaciones; sin embargo, en la bibliografía consultada no se han identificado propuestas donde se integren ambas variantes de implementación con vistas a potenciar sus fortalezas y mitigar sus limitaciones. Por otro lado el estudio realizado evidencia inexistencia de una propuesta formal de pasos dirigidos a la fase de implementación al aplicar el enfoque de reglas de negocio. También se ha identificado como una deficiencia la carencia de alguna propuesta de medidas que permitan evaluar las variantes existentes para implementar el enfoque de reglas de negocio. Es válido mencionar que este trabajo se enfoca en soluciones basadas en Java, primeramente por ser su carácter no propietario y también porque existe un grupo de herramientas y tecnologías maduras que permiten el desarrollo de aplicaciones empresariales y la implementación del enfoque de reglas de negocio basadas en este lenguaje [10].

A partir de lo expuesto anteriormente, en este trabajo se define como objetivo, integrar motores de reglas mediante la programación orientada a aspectos en la capa de lógica de negocio para el desarrollo de aplicaciones en Java, siguiendo el enfoque basado en reglas de negocio. Para cumplir este objetivo se proponen los pasos a seguir durante la fase de implementación del enfoque basado en reglas utilizando una solución que integra las dos variantes mencionadas. Además, se proponen medidas para evaluar el comportamiento de la implementación del enfoque de reglas utilizando las variantes existentes

y la propuesta de integración de ambas. Se compara el desempeño de las tres variantes evaluado las medidas propuestas mediante el desarrollo de un ejemplo.

MATERIALES Y MÉTODOS

A continuación se fundamenta la selección del motor de regla y la herramienta AOP basados en Java a utilizar en la implementación del enfoque basado en reglas. Dado el auge de las tecnologías de código abierto y el potencial del lenguaje Java para el desarrollo de aplicaciones empresariales, el estudio presentado en este trabajo se centra en motores y herramientas AOP desarrollados sobre el lenguaje Java.

Selección del motor de reglas a utilizar basado en Java

Un paso importante al implementar el enfoque de reglas de negocio empleando motores de reglas, es decidir qué motor utilizar, de modo que se facilite su integración en las aplicaciones desarrolladas.

Durante la última década se han desarrollado varios motores de reglas. Algunos son herramientas comerciales como: Blaze Advisor [9], LOG Rules [18], Oracle Rule Engine [19] y otras son herramientas libres y de código abierto como: Open Rules [20] y JBoss Rules [21].

A continuación se realiza un análisis comparativo de tres motores de regla de código abierto desarrollados sobre el estándar de Java JSR-94. Este estándar es una especificación que define lineamientos y proporciona una interfaz para programar aplicaciones común para la implementación y ejecución de motores de reglas en Java [4] [22]. Para realizar el análisis comparativo se definieron seis criterios. Para ello se tomaron como base diversos estudios donde se analizan diferentes implementaciones de motores de reglas de negocio [10] [23] [24] [25]. Los criterios empleados son:

1. Documentación disponible, elemento que resulta muy útil para asimilar la tecnología y poder explotarla al máximo.
2. Facilidad de integración con aplicaciones desarrolladas en Java. Este criterio está determinado por el hecho de si se sigue o no el estándar JSR-94.
3. Existencia de *plugins* que permitan la integración con el ambiente de desarrollo integrado (IDE, por sus siglas en inglés) Eclipse, hecho que facilita el desarrollo de aplicaciones en Java.
4. El rendimiento del motor. Los motores de reglas por sus características provocan un aumento en el consumo de recursos lo que puede afectar el rendimiento de las aplicaciones.
5. El mecanismo de inferencia que implementa. El mecanismo de inferencia está determinado por un algoritmo de emparejamiento que permite determinar las reglas que se han cumplido sobre la base de una serie de hechos.
6. Soporte de la herramienta, el cual está respaldado por la comunidad de desarrolladores de la herramienta. Para valorar este criterio se precisa la fecha en que se liberó la última versión estable de la herramienta.

En la tabla 1 se resume el análisis de tres motores de reglas, tomando en cuenta los criterios definidos anteriormente.

De los motores analizados se selecciona JBoss Rules o Drools como propuesta para implementar el enfoque de reglas de negocio en el desarrollo de aplicaciones de Java. Este motor de reglas de negocio posee un fuerte soporte por la comunidad de desarrolladores, es de código abierto, se integra fácilmente con aplicaciones en Java, posee una amplia documentación y un mecanismo de inferencia sólido, basado en una mejora realizada al algoritmo Rete (utilizado en varios mecanismos de inferencia de sistemas expertos). Incluso en el reporte de Forrester Research [25] destacan sus fortalezas al compararlo con potentes motores comerciales como ILOG Rules .

Selección de la herramienta de programación orientada a aspectos a utilizar basada en Java

Para el estudio y selección de la herramienta a emplear se utilizarán conceptos importantes del paradigma AOP como son: puntos de unión, puntos de corte, aviso, aspecto y el proceso de entretejido o recomposición de aspectos [26]. En primer lugar se tuvo en cuenta la forma en que se definen los aspectos, ya que esto facilita su declaración. En Java existen tres formas principales de declarar aspectos: por código, por anotaciones o mediante ficheros XML. Como segundo criterio se evaluó el tipo de mezcla o entretejido que implementa cada herramienta. En tercer lugar se analizó el entorno de ejecución de cada herramienta, prefiriendo utilizar herramientas que sean independientes de la plataforma que les da soporte. Por último, se analizó la cantidad de documentación disponible para cada herramienta ya que constituye un factor clave a la hora de asimilar y aplicar una tecnología. Entre las herramientas basadas en Java se destacan tres, las cuales son comparadas en la tabla 2.

Después del análisis comparativo se decide seleccionar AspectJ basándose en las potencialidades que tiene para el desarrollo sobre la plataforma Java. En la decisión se tomó en cuenta la posibilidad que ofrece para chequear de manera estática los aspectos declarados y la existencia de un amplio modelo de puntos de unión, lo que facilita la intercepción de los métodos de la lógica de negocio y por último, la existencia de una amplia documentación, hecho que favorece la curva de aprendizaje de la tecnología. AspectJ es una extensión del lenguaje Java que permite declarar los aspectos completamente a través de código y anotaciones, lo cual evita redundancia en la definición de tipos y reduce la ocurrencia de errores.

Tabla 1
Resumen del análisis de motores de reglas de negocio

Criterios	Hammurapi Rules	Open Rules	JBoss Rules o Drools
Documentación disponible	Pobre	Media	Amplia
Facilidad de integración con aplicaciones	Media	Media	Alta
Rendimiento	Alto	Alto	Alto
Mecanismo de inferencia	Hacia delante y detrás	No, es basado en tablas de decisión definidas en Excel	Hacia delante y detrás basado en una implementación mejorada del algoritmo Rete OO
Integración con el IDE Eclipse	No se conoce que disponga de *plugins* para integrarlo con Eclipse	Sí	Sí
Soporte, última versión estable de la herramienta	4.0 (noviembre 2008)	6.3.2 (diciembre 2014)	6.2 (febrero 2015)

Tabla 2.
Resumen del análisis de las principales herramientas AOP en Java

Criterio	AspectJ	JBoss AOP	Spring AOP
Definición de aspectos por:			
• Código	x		
• Anotaciones	x	x	
• XML		x	x
Mezcla de aspectos:			
• En tiempo de compilación	x	x	
• En tiempo de carga	x	x	
• En tiempo de ejecución	x	x	x
Documentación	Amplia	Media	Amplia
Entorno de ejecución	Programa plano en Java	Invocado y administrado por la plataforma JBoss	Invocado y administrado por la plataforma Spring

RESULTADOS

Como se mencionó anteriormente, en el enfoque de reglas de negocio, se proponen seis fases a transitar para desarrollar sistemas que basan su lógica de negocio en reglas. En el enfoque, las cuatro primeras fases están bien definidas y formalizadas. Ahora para la fase de implementación, el enfoque carece de una propuesta formal de pasos que permitan transitar por esta fase, fundamentalmente porque esta fase está vinculada a la selección tecnológica que se decida utilizar. A continuación se expone uno de los principales resultados de este trabajo que consiste en una propuesta de pasos a seguir en la fase de implementación de enfoque de reglas de negocio. Esta propuesta de nueve pasos está dirigida a la vía de solución que combina un motor de reglas de negocio y la programación orientada a aspectos.

1. Incluir en la aplicación un grupo de bibliotecas necesarias para integrar el motor de reglas de negocio. Algunos motores reglas de negocio permiten ser acoplados directamente al código fuente de la aplicación. Para esto se debe agregar a la aplicación un conjunto de bibliotecas que especifican las clases y métodos para trabajar con el motor de reglas seleccionado.

2. Crear una base de reglas. Esta base de reglas también se conoce como repositorio de reglas. En algunas herramientas o motores la base de reglas puede ser un fichero texto, en otros casos, las reglas se pueden almacenar en una base de datos y la herramienta se encarga de la gestión de las reglas.

3. Incluir en la base de reglas las clases o paquetes de clases que se van a utilizar para la definición de las reglas.

4. Definir las reglas en formato y lenguaje específico para el motor de reglas que se esté utilizando.

5. Validar la sintaxis de las reglas contenidas en el repositorio. En muchas herramientas este paso se realiza de forma automática. Para el caso de los motores es en el momento en que se cargan las reglas.

6. Identificar los puntos de la lógica que van a ser interceptados, por los aspectos que van a interactuar con el motor de reglas de negocio. Se debe tener identificado qué clases y métodos de la lógica de negocio van a ser interceptados y qué reglas deben ser invocadas en cada caso.

7. Definir puntos de corte y puntos de unión. El cómo se interceptan los métodos de la lógica de negocio está dado por el tipo de punto de corte a utilizar. Puede ser un punto de corte que intercepta antes de entrar a la ejecución del método (punto de corte conocido como *before)*. Puede ser un punto de corte de tipo *after*, que se ejecuta la intercepción al concluir el método.

8. Definir qué información se pasará a las reglas. Al utilizar AOP cuando un aspecto intercepta un método este captura información que está en el contexto del método en cuestión. Dicha información es pasada al aspecto, dentro del cual puede ser modificada como resultado de la ejecución de alguna regla de negocio y al finalizar la ejecución del aspecto, la información debe regresar al contexto del método con los cambios realizados.

9. Implementar dentro del aspecto la interacción con el motor de reglas. Esto implica incorporar las líneas de código necesarias para interactuar con el motor de reglas.

Los cinco primeros pasos son propios de la variante que solo utiliza el motor de reglas de negocio. Los pasos del seis al nueve son similares a la propuesta donde solo se aplica AOP como variante de solución. La diferencia está dada por la función de los aspectos. En el paso nueve para la variante que solo usa AOP, los aspectos son los responsables de implementar las reglas de negocio. En esta nueva propuesta de integración, los aspectos se encargan de implementar la comunicación entre el motor de reglas y el núcleo de lógica de negocio. Es válido mencionar que para obtener esta propuesta integradora de nueve pasos, fue necesario previamente definir las propuestas de pasos para la fase de implementación, tanto para la vía que solo utiliza el motor de reglas, así como para la vía que solo utiliza programación orientada a aspectos.

La figura 1 ayuda a comprender cómo sería la interacción entre el núcleo principal de lógica de negocio de la aplicación y el motor de reglas mediante los aspectos. Los aspectos mediante los puntos de corte definidos, interceptan los métodos de la lógica de negocio y capturan la información de la llamada a los métodos. Esto se representa por las flechas etiquetadas con 1. Dentro de los aspectos está la interacción con el motor de reglas. La información que es capturada por los aspectos se pasa al motor, esto se representa por las flechas etiquetadas con 2. Sobre la base de la información recibida, el motor evalúa las reglas y ejecuta las que se cumplen. La información puede ser modificada por la ejecución de las reglas. Seguido a esto, la información regresa a los aspectos, lo cual se representa con las flechas marcadas con 3. Finalmente, la información es devuelta al contexto de la lógica de negocio y la aplicación continúa su flujo de ejecución, esto es representado mediante las flechas identificadas con 4.

A partir de la revisión bibliográfica realizada, se pudo observar que solo en algunos trabajos que implementan el enfoque de reglas de negocio se utilizan medidas para valorar las ventajas que este ofrece. Estas medidas van dirigidas fundamentalmente al análisis de la cantidad de líneas de código y la mantenibilidad de cada solución. Por otro lado, la norma ISO 9126 Parte 3 propone medidas que permiten evaluar la calidad interna de un software. Para evaluar la característica de calidad eficiencia se definen varias medidas, las cuales son utilizadas para predecir la eficiencia del comportamiento de un software. Asociado a la eficiencia se define la medida comportamiento del tiempo. Esta medida define atributos que permiten predecir el comportamiento del tiempo de rendimiento de una aplicación. Uno de los atributos es el tiempo de respuesta, que se define como el tiempo estimado para completar una tarea específica.

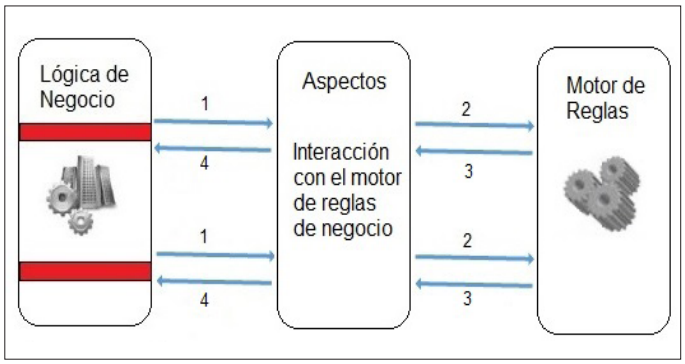

Fig. 1. Interacción entre la lógica de negocio y el motor de reglas utilizando aspectos.

Otras medidas propuestas por la ISO 9126 permiten evaluar la característica asociada a la facilidad de mantenimiento o mantenibilidad de un software. Uno de los atributos de esta medida es la capacidad de cambios que puede presentar un software. Esta medida está destinada a estimar el esfuerzo necesario cuando se trata de implementar una modificación al software, o sea, el esfuerzo ante un cambio. Basado en lo anterior se propone tener en cuenta las siguientes medidas para evaluar la solución propuesta y compararlas con las dos variantes tecnológicas identificadas en la literatura. No se utilizan estrictamente las medidas propuestas por la ISO pues estas van dirigidas a productos de software en fase final de desarrollo o en explotación.

- Rendimiento de la aplicación: El rendimiento de la aplicación indica su comportamiento respecto el tiempo de ejecución. Esta medida será expresada en segundos.

Para evaluar la facilidad de mantenimiento se propone utilizar dos medidas:

- Tiempo para cambiar una regla. Los valores de esta medida se propone que sean evaluados en segundos.

- Complejidad del cambio. Para evaluar esta medida se propone contar la cantidad de operaciones necesarias para modificar una regla de negocio.

Adicionalmente, se recomienda utilizar como medida:

- Cantidad de líneas de código. Esta medida es el resultado de contar las líneas de código presentes en cada clase necesaria para implementar las reglas de negocio.

Por otro lado, cuando se analizaron los motores de reglas como vía para implementar el enfoque de reglas, se identificó como deficiencia la generación de código de dispersión y código entremezclado en la capa de lógica de negocio. En la literatura revisada no se identificaron propuestas para evaluar estos fenómenos. Ante esta situación se proponen dos medidas para analizar cómo se comportan estos fenómenos:

- Dispersión del código. Para esta medida se evalúa la cantidad de veces que es necesario repetir en la lógica de negocio las líneas de código necesarias para validar y ejecutar reglas de negocio.

- Código entremezclado. Para esta medida se propone contar la cantidad de líneas de código necesarias para validar y ejecutar reglas de negocio que están mezcladas con la lógica de negocio de la aplicación.

DISCUSIÓN

Para evaluar la propuesta de integración de motores de reglas mediante programación orientada a aspectos en el desarrollo de aplicaciones en Java siguiendo el enfoque de reglas de negocio, se elabora un ejemplo basado en una tienda virtual. En la tienda existen dos procesos de negocio fundamentales, la compra y la devolución de productos. Para estos procesos la tienda tiene definidas seis reglas de negocio que se muestran en la tabla 3.

Para dar solución al caso de estudio se modelaron e implementaron las siguientes clases en Java (ver figura 2). Se debe destacar en este modelo las clases *OrdendeCompra* y *OrdendeDevolución* en las cuales están presenten los métodos *CalcularDescuento* y *CalcularImpuestoPorDevolucion* donde es necesario validar y ejecutar reglas de negocio.

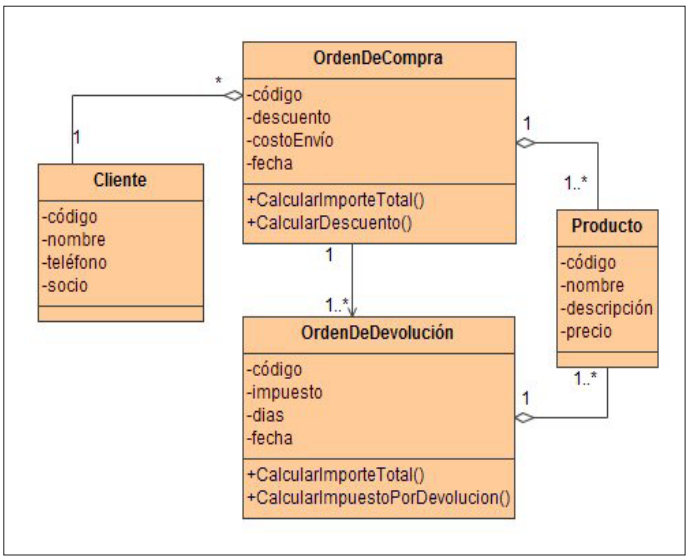

Fig. 2. Modelo de clases del caso de estudio

Como parte de la propuesta de solución un paso importante es implementar las reglas de negocio que son validadas y ejecutadas dentro del motor de reglas. Para este caso se trata del motor *JBoss Rules*. A continuación se muestra un fragmento del fichero *online-shop.drl* donde está definida la base de reglas que utiliza el motor. Como se observa en la figura 3, las reglas de negocio están implementadas en una sintaxis que es interpretada por el motor de reglas. Primeramente, se define el nombre de la regla usando la palabra reservada *rule*. A continuación aparece la palabra *when* que se utiliza para definir las condiciones de las reglas y por último, la sentencia del *then*, que define la acción que ejecuta la regla.

El otro paso importante dentro de la propuesta de solución es la interacción con el motor de reglas. Esta interacción consiste en cuatro pasos, primero se invoca al motor de reglas, luego se crea la sesión de trabajo, se insertan hechos a la sesión activa y se mandan a ejecutar las reglas. Todo esto se corresponde con cuatro líneas de código dentro del método *ShopPointCut*, el cual se muestra en la figura 4.

Tabla 3
Reglas de negocio definidas para el ejemplo basado en la tiendaa virtual

Código de la regla	Descripción de la regla de negocio en lenguaje natural
RN1	Si es un cliente asociado a la tienda entonces el costo de envío es gratis
RN2	Si no es un cliente asociado a la tienda, el costo de envío es el 2 % del importe total de la compra
RN3	Si el importe total de la compra está entre $500 y $1 000 se debe aplicar un 5 % de descuento
RN4	Si el importe total de la compra es superior a $1 000 se debe aplicar un 10 % de descuento
RN5	Si la devolución tiene lugar en los primeros siete días posteriores a la compra, se le reintegra al cliente el 90 % del importe total del precio original de los productos a devolver
RN6	Si la devolución tiene lugar entre 8 y 30 días posteriores a la compra, se le reintegra al cliente el 75 % del importe total del precio original de los productos a devolver

```
rule "RN1 Envío Gratis"
  when
   $orden : Orden
   OrdenDeCompra ($cliente: Cliente)
   Cliente (socio == true) from $cliente
  then
   $orden.costoEnvío = 0.0 ;
end

rule "RN2 Calcular Costo Envío"
  when
   $orden : Orden
   OrdenDeCompra ($cliente: Cliente)
   Cliente (socio == false) from $cliente
  then
   $orden.costoEnvío =
        orden.CalcularImporteTotal() * 0.02;
end
```

Fig. 3. Fichero *online-shop.drl* con la definición de las reglas de negocio.

Como se observa en la tabla anterior, al implementar la primera y la tercera variante las medidas son similares, tomando en consideración el rendimiento y la capacidad de mantenimiento. En ambas variantes se logran mantener las fortalezas al utilizar motores de reglas, asociada a la capacidad de mantenimiento. Ello implica que utilizando la solución propuesta se disminuye el tiempo y la cantidad de operaciones necesarias para cambiar una regla, respecto a utilizar la variante con AOP. Para ejecutar un cambio con la primera y la tercera variante solo se debe modificar el fichero de las reglas. Este es un fichero de texto y solo es necesario abrirlo, localizar y modificar la regla, y salvar el cambio. La próxima vez que el motor vuelva a cargar el fichero ya la regla está actualizada. En cambio, al utilizar la segunda variante se requiere abrir el código fuente con la herramienta Eclipse, localizar el aspecto que contiene la regla que se desea cambiar, modificar la regla, salvar el cambio, recompilar y volver a ejecutar la aplicación.

```
public class DiscoutOrderAspect {

    @PointCut ("excution (* model.OrdenDeCompra.CalcularDescuento(..) ) && args (orden)")
    public void ShopPointCut(OrdenDeCompra orden){
    }

    @Before("ShopPointCut(orden)")
    public void ShopPointCut (final JoinPoint thisJoinPoint, OrdenDeCompra orden){
        WorkingMemory workingMemory = ManageRules.getRuleBase().newStatefulSession();
        workingMemory.insert (orden);
        workingMemory.fireAllRules();
        workingMemory.dispose();
    }
}
```

Fig. 4. Código fuente de un aspecto encargado de interactuar con el motor de reglas.

Tabla 4
Resumen del desempeño de las variantes para implementar el enfoque basado en reglas utilizando las medidas propuestas

Medidas	Característica / Atributo	Variantes de solución		
		JBoss Rules	AspectJ	JBoss Rules + AspectJ
Tiempo de respuesta (s)	Rendimiento	2	1	2
Número de líneas de código fuente para implementar reglas de negocio	Tamaño	10	32	22
Número de Operaciones necesarias para cambiar una regla de negocio	Capacidad de mantenimiento	3	6	3
Tiempo necesario para cambiar una regla de negocio (s)	Capacidad de mantenimiento	<60	Entre 120 y 180	<60
Cantidad de veces que es necesario repetir en la lógica de negocio las líneas de código necesarias para validar y ejecutar reglas de negocio	Dispersión del código	2	0	0
Cantidad de líneas de código necesarias para validar y ejecutar reglas de negocio que están mezcladas con la lógica de negocio de la aplicación	Código entremezclado	8	0	0

Respecto a potenciar las ventajas que ofrece utilizar la variante AOP, al integrarla con el motor se logra eliminar el fenómeno de código disperso y entremezclado. O sea, tanto al aplicar la segunda como la tercera variante desaparece de la lógica de negocio el código responsable de validar y ejecutar la reglas de negocio, ya que en la segunda variante, el código se implementa en el cuerpo de los aspectos y en la tercera, en el cuerpo del aspecto se invoca al motor de reglas, quien se encarga de estas tareas. Por lo tanto, ambas variantes superan a la primera.

Respecto al tamaño, o sea, la cantidad de líneas de código para implementar reglas de negocio, se observa cómo la variante que requiere menos líneas de código es la que utiliza únicamente el motor de reglas y la solución que requiere más líneas, resulta ser la que utiliza solo AspectJ. Esto se debe a la cantidad de líneas de código que se necesitan para implementar cada aspecto. Respecto a la tercera variante, solución propuesta, el número de líneas de código es superior a la primera variante. Esto pudiera considerarse como una desventaja, pero no es así, ya que la tercera variante al utilizar aspectos para integrar el motor a la aplicación, reduce los efectos de dispersión de código y código entremezclado que se produce cuando hay que incorporar el motor de reglas directo al núcleo de lógica de negocio de la aplicación.

Por lo explicado anteriormente, se puede concluir que la solución propuesta resulta la variante de mejor desempeño. Sin embargo, un aspecto a atender es el rendimiento, ya que el tiempo de ejecución utilizando la variante AOP fue ligeramente inferior que utilizando las variantes que aplican motores de reglas.

CONCLUSIONES

Al culminar el trabajo se arriban a las siguientes conclusiones:

- Para desarrollar aplicaciones empresariales utilizando el enfoque de reglas de negocio se necesitan herramientas o tecnologías que deben ser definidas en la fase de diseño y después juegan un papel importante de la fase de implementación.
- JBoss Rules se destaca como un potente motor de reglas de negocios para implementar el enfoque de estas en aplicaciones desarrolladas en Java, cumple con la especiación JSR94 lo que facilita su integración en aplicaciones y además posee una amplia documentación y cuenta con un buen soporte por la comunidad de desarrolladores.
- AspectJ se destaca entre las herramienta AOP para el entorno Java, dado que cuenta con un amplio modelo de puntos de unión, lo que facilita la intercepción de los métodos de la lógica de negocio, así como a la existencia de una amplia documentación.
- Se definieron los nueve pasos a seguir para implementar el enfoque de reglas de negocio integrando el motor de reglas de negocio JBoss Rules mediante el empleo de la herramienta de programación orientada a aspectos AspectJ para desarrollar una aplicación en Java.
- Se definen seis medidas para evaluar el desempeño de las variantes que implementan el enfoque de reglas de negocio: una para valorar el rendimiento, otra para el tamaño, dos para la capacidad de mantenimiento de las reglas, y dos para evaluar el código disperso y entremezclado, respectivamente.
- Se comprobó que la solución propuesta constituye la mejor variante de las evaluadas para implementar el enfoque de reglas de negocio ya que combina las potencialidades de las variantes que integra, aprovecha la capacidad de mantenimiento que ofrece el empleo de los motores de reglas y la fortaleza de AOP para eliminar el código disperso y entremezclado.

Para trabajos futuros se propone: incorporar nuevas medidas que ayuden a mejorar la evaluación del desempeño de las variantes tecnológicas, implementar el enfoque de reglas utilizando otras herramientas AOP y otros motores de reglas, así como diseñar un caso de estudio más complejo que incluya más reglas de negocio para mejorar la comparación de las variantes de implementación del enfoque basado en reglas.

REFERENCIAS

1. **VON HALLE, B.** *Business Rules Applied-Building Better Systems Using the Business Rules Approach. USA: John Wiley & Sons, Inc*, 2002.
2. **ROSS, R. G.** *Business Rule Concepts-Getting to the Point of Knowledge. USA: Business Rule Solutions Inc*, 2005.
3. **MESERVY, T. O.; ZHANG, Chen.** *et. al*."The Business Rules Approach and Its Effect on Software Testing". *IEEE COMPUTER SOCIETY.* 2012, núm.4, vol. 29, pp. 60-66.
4. **PACHECO, Y.; ESTÉVEZ, S.; MARTÍNEZ, M. E.** "Integración de un sistema de gestión de reglas de negocio al flujo de trabajo control de historias clínicas para trasplante renal". *Revista Cubana de Informática Médica.* 2015, núm. 1, vol. 7, pp. 105-112.
5. **CHAWARE, S.; KARANDIKAR, A,** "Development of Loan Module using Business Rules Engine". *COMPUSOFT, An International Journal of Advanced Computer Technology.* 2014, núm. 7, vol. 3, pp. 1007-1011.
6. **CASAS, S.; MARCOS, C.** *et al.* "Estrategias para la Integración y Conexión de Reglas de Negocio con Aspectos". XII Workshop de Investigadores en Ciencias de la Computación, pp. 337-341, 2010.
7. **GANG, Z.; TAO, G.; JIEJUN, L.** "Implementation of business rules approach in transformer fault detection system". *7th International Conference on Computer Science & Education (ICCSE), IEEE,* vol. 12, pp. 135-138, 2012.
8. **CHAWARE, S.; KARANDIKAR, A.** "A Review of Methods for Developing Business Rules Engine" *International*

Journal of Advanced Research in Computer and Communication Engineering. 2014, núm. 3, vol. 3, pp. 5878-5880.

9. **GARCÍA, J. C.; ANDRÉ, M.** "Implementación del enfoque de reglas de negocio utilizando motores de reglas en el desarrollo de aplicaciones Java." *Ciencias de la Información.* 2015, vol. 46, núm. 1, pp. 41-46.
10. **ALONSO, J. L.** "Aplicación del Enfoque de Reglas de Negocio sobre el paradigma MDA". Tesis de Maestría en Informática Aplicada, Departamento de Ingeniería de Software, Universidad Tecnológica de La Habana José Antonio Echeverría, Cujae, Cuba, 2011.
11. **GARCÍA, J. C.** "Utilización de Motores de Reglas de Negocio en Aplicaciones J2EE". Tesis de Grado, Departamento de Ingeniería de Software, Universidad Tecnológica de La Habana José Antonio Echeverría, Cujae, Cuba, 2009.
12. **KELLENS, A.; DE SCHUTTER, K.** *et al*. "Experiences in modularizing business rules into aspects". *IEEE International Conference on Software Maintenance, ICSM. IEEE,* vol. 8, pp. 448-451, 2008.
13. **MUKUNDAN, A.; CHANG, L.** "An Engine-Independent Framework for Business Rules Development". *15th IEEE International Enterprise Distributed Object Computing Conference, IEEE,* vol. 11, pp. 75-84, 2011.
14. **CIBRÁN, M. A.; D'HONDT, M.** "Composable and Reusable Business Rules Using AspectJ" in *Proceedings of the Workshop on Software Engineering Properties of Languages for Aspect Technologies (SPLAT) at the International Conference on Aspect-Oriented Software Development*. Boston, USA, 2003.
15. **SOUMEYA, D.; DJAMEL, M.** "Quantitative and qualitative evaluation of AspectJ, JBoss AOP and CaesarJ, using Gang-of-Four design patterns". *International Journal of Software Engineering and Its Applications.* 2013, vol. 7, núm. 6, pp. 157-174.
16. **BOGGIANO, M. B.; CASTRO, A.** "SLD149 Validación de la funcionalidad del traductor LPT-SQL para la base de datos de trasplante renal". IX Congreso Internacional Informática en Salud, La Habana, Cuba, 2013.
17. **WU, B.; GUO, Z.** "Research and Application of Rule Engine in Scheduled Outage Management" in *Proceedings of 2nd International Conference on Computer and Information Application (ICCIA 2012)*, pp. 446-449, 2012.
18. IBM. *Business Rules Management System (BRMS)*. 2015. Available: http://www-01.ibm.com/software/websphere/products/business-rule-management/.
19. **SILVERIA, L.** "Diseño e implementación de un motor de reglas dinámicas usando especificaciones GeneXus". Tesis de Maestría, Facultad de Ingeniería, Universidad de la República, Uruguay, 2010.
20. OPENRULES. *Sitio Oficial de Open Rules*. 2012. Available: http://openrules.com/.
21. DROOLS. *Reference Manual Drools 6.1*. Available: http://docs.jboss.org/drools/release/6.1.0.Final/drools-docs/html_single/index.html. 2013.
22. **DI, L.; TAO, G.; JIANG-PING, X.** "Rule Engine based on improvement rete algorithm" *International Conference on Apperceiving Computing and Intelligence Analysis (ICACIA),* vol. 10, pp. 346-349, 2010.
23. **MARTÍNEZ, J.** "Introduciendo Semántica en un Proceso de Desarrollo de Software a través de Reglas de Negocio". Tesis Doctoral, Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid, España, 2010.
24. **RIVILLAS, O.** "Estado del arte de los motores de reglas de negocio BRM". Tesis de Grado, Facultad de Ingeniería, Universidad de San Buenaventura, Colombia, 2012.
25. **RYMER, J.; GUALTIERI, M.** Market Overview: Business Rules Platforms 2011. Forrester Research Report.
26. **HERNÁNDEZ, L. R.** "Un modelo para la implementación de la seguridad de una aplicación Web con el uso de la Programación Orientada a Aspectos". Tesis de Maestría en Informática Aplicada, Departamento de Ingeniería de Software, Universidad Tecnológica de La Habana José Antonio Echeverría, Cujae, Cuba, 2012.

AUTORES

Juan Carlos García González
Ingeniero Informático, Facultad de Ingeniería Informática, Universidad Tecnológica de La Habana José Antonio Echeverría, Cujae, Cuba

Margarita André Ampuero
Ingeniera Informática, Doctora en Ciencias Técnicas, Profesora Titular, Facultad de Ingeniería Informática, Universidad Tecnológica de La Habana José Antonio Echeverría, Cujae, La Habana, Cuba

Rule Engines Integration Using Aspect Oriented Programming in Java Applications Development

Abstract
In several approaches of software development, the business logic layer of the application tends to be complex, because it must implement the existing business rules in an organization. In some systems business rules change more frequently than other elements. The business rules approach proposes a way of application development where business rules play a key role. To implement the business rules approach two fundamental ways stand: rules engines and Aspect Oriented Programming. This paper proposes the integration of these two variants of solution in order to maximize the benefits of both and minimize limitations. In order to validate the proposal, six measures are defined. The measures are evaluated using the rule engine JBoss Rules, the AspectJ tool and JBoss Rules integration with AspectJ. Finally, through an example it is found that the integration of the two approaches is superior.

Key words: business rules approach, business rules engine, Aspect Oriented Programming, JBoss Rules, AspectJ