Recomendaciones para la incorporación de reglas de negocio y construcción de un Editor de Reglas en lenguaje natural

Informe de Proyecto de Grado 2012

Instituto de Computación Facultad de Ingeniería Universidad de la República

Estudiantes: Stephanie Catarino – Mauricio Hernández
Tutor: Guzmán Llambías
Cliente: Mariana Grunfeld - BPS

Contenido

Resumen

En las organizaciones existen especificaciones llamadas reglas de negocio. Las reglas de negocio pueden ser vistas como condiciones en donde se define o se restringe algún aspecto del mismo. Estas reglas suelen cambiar con bastante frecuencia, lo que genera la necesidad por parte de las organizaciones de responder de la forma más rápida y eficiente posible para lograr una rápida adaptación a los cambios.

El Banco de Previsión Social (BPS) se encarga cubrir las contingencias sociales de la comunidad en su conjunto y de la recaudación de recursos. BPS posee una realidad muy variada y un negocio complejo, el cual resulta particularmente dinámico al verse afectado por cambios en normas, decretos y reglamentaciones. Por estas razones las reglas de negocio varían de forma frecuente y existe interés a nivel organizaciones de comenzar gestionar las mismas.

Uno de los objetivos de este proyecto es arrojar recomendaciones de cómo administrar las reglas de negocio a nivel organizacional y de cómo se debe administrar la regla en sí, todo esto buscando responder a los cambios de las mismas de la mejor forma posible.

Para esto se realizaron investigaciones acerca de gestión de las reglas de negocio en una organización, donde se documentó debidamente lo estudiado y donde además se realizó una propuesta propia de cómo debe llevarse a cabo.

Hoy en día existen diferentes tipos de software para la administración de reglas de negocio, esto implica definirlas, mantenerlas, lograr que los cambios en estas impacten inmediatamente en la organización, entre otras cosas.

Partiendo de estándares para la expresión de reglas en lenguaje natural, de forma que estas no resulten ambiguas, de software existente para manipulación de reglas de negocio, así como de otras herramientas, este proyecto de grado tiene a la postre como objetivo mejorar las herramientas para la definición de las reglas de negocio de una manera más amigable para quién las define, que comúnmente suele ser un analista de negocio. Esto implicó desarrollar una herramienta donde se permite expresar las reglas en lenguaje natural y de forma asistida. El lenguaje natural presenta ambigüedades que pueden resultar un problema a la hora de escribir una regla, para eso se ha estudiado un estándar con pautas para la expresión de las mismas sin que contengan múltiples sentidos, dotando a nuestra herramienta de las bondades de este estándar.

El escenario de aplicación de este proyecto es el "Cierre de obras" que lleva a cabo el área de Atyr dentro del Instituto de Previsión Social (BPS), las cuales fueron editadas con el Editor de reglas desarrollado, permitiendo un rediseño del mismo basado en reglas de negocio y el BRMS JBoss Drools.

1. Introducción

En las distintas organizaciones existen especificaciones llamadas reglas de negocio. Las reglas de negocio pueden ser vistas como condiciones en donde se define o se restringe algún aspecto del mismo. Estas reglas suelen cambiar con bastante frecuencia, lo que genera la necesidad por parte de las organizaciones de responder de la forma más rápida y eficiente posible para lograr una rápida adaptación a los cambios.

1.1 Motivación.

En cualquier organización, es importante que los Analistas de Negocio puedan definir y mantener las reglas necesarias, apropiadas y ajustadas para responder correctamente a las realidades circunstanciales de su negocio. Poder evitar tener que rehacer el trabajo de forma indefinida y concentrar las 'claves' del negocio en una única fuente son atributos valiosos a nivel organizacional. Las reglas de negocio tienden a cambiar con mucha frecuencia lo que conlleva a que se presenten distintos tipos de dificultades que nos hacen plantear la necesidad de definir ciertos criterios, entre las que se destacan facilitar la creación, implementación y mantenimiento de reglas permitiendo su integración con el resto de los Sistemas de Información.

De manera natural aparece como la principal motivación de nuestro proyecto de grado investigar los métodos, procesos, técnicas y tecnologías, necesarios para acercar a los Analistas de Negocio al proceso de creación y mantenimiento de las reglas que representan el comportamiento del negocio.

El Banco de Previsión Social (BPS) en los últimos tiempos ha visto incrementado la necesidad de gestionar las reglas de negocio dentro de la institución, debido a la gran casuística y dinámica con la que cambian las mismas. Hoy en día para BPS este es un terreno desconocido con lo se nos presenta como motivación realizar las investigaciones pertinentes para poder brindar asesoramiento al organismo.

Otra motivación que surgió a raíz de un planteo genérico del cliente fue el desarrollo de un artefacto de software genérico que nos brinde la posibilidad de instanciar el mismo para facilitar así la incorporación de las reglas de negocio en los distintos aplicativos.

1.2 Resultados esperados.

Al comienzo del proyecto fueron definidos a grandes rasgos tres resultados esperados.

En primer lugar se planteó la necesidad de realizar un relevamiento de las reglas de negocio del área Construcción de Fiscalización de Atyr así como una clasificación de las mismas en diferentes tipos.

En segundo lugar aparece la necesidad de poder contar con recomendaciones genéricas que ayuden la incorporación de reglas de negocio dentro de distintos tipos de casos que se puedan presentar. Esto incluye como se definen, como se identifican, como se clasifican y como se incorporan las reglas. Existe un aplicativo en el área de Construcción de Fiscalización en donde

las reglas de negocio cumplen un rol muy importante y donde hoy en día se realiza un manejo muy rígido de las mismas. Se pretende entonces también poder tener recomendaciones para poder incorporar reglas de negocio a este aplicativo.

Por último, un resultado esperado importante al que se pretendió llegar, fue crear un artefacto de software genérico que facilite la edición de las reglas de negocio por parte de los usuarios funcionales, evitando la asistencia del equipo técnico especializado.

1.3 Evolución del proyecto.

A medida que se fue avanzando en la investigación de todo lo referente a las reglas de negocio en las organizaciones, se fue informando al cliente de lo aprendido mediante la entrega de diferentes documentos y reuniones.

De esta forma, con algunos meses de investigación donde se ganó terreno en el aprendizaje de la temática y donde también se pudo realizar un relevamiento del área de negocio y de las potenciales reglas, pudimos ver lo planteado inicialmente con un mayor conocimiento y plantear de forma más concreta entre todos los actores involucrados (cliente, estudiantes y tutor) cuáles serían los entregables esperados como resultado del proyecto.

Se acordó entonces como entregable del proyecto un conjunto de recomendaciones y definiciones que ayuden a la incorporación de las reglas de negocio a la organización, donde podemos dividir esto en los siguientes entregables:

1. Documento donde se presente la investigación realizada acerca de la regla de negocio en sí y donde se incluya una definición propia de parte de los estudiantes.
2. Documento presentando lo investigado acerca de clasificación de reglas de negocio y al igual que en el entregable anterior, una clasificación propia.
3. Documento que contiene recomendaciones de cómo llevar a cabo el ciclo de la regla de negocio en sí, tomando a la regla de negocio como algo atómico.
4. Documento que también es una recomendación de cómo llevar a cabo el ciclo de vida de la regla de negocio, pero esta vez visto desde el punto de vista organizacional.
5. Editor de reglas de negocio. Permite editar reglas de negocio en lenguaje natural, realizar testeo de las mismas e integrarlas con plataforma tecnológica de BPS.

Los entregables 1 y 2 son referentes a definiciones y los 3 y 4 son de recomendaciones.

El entregable 5 es resultado también de la investigación realizada. Se visualizó la necesidad de crear un entorno para la expresión de reglas que sea amigable para el Analista de negocio, dado que los editores que se investigaron eran potentes para la definición de reglas pero demasiado técnicos para un Analista y poco amigables, es decir, requerían de un especialista técnico que acompañe el proceso de creación.

Estas razones llevaron a ponernos como objetivo la creación de un entorno donde se puedan crear, mantener, testear entre otras cosas las reglas de negocio de una manera intuitiva y utilizando términos familiares para los usuarios. A su vez se pretendió que este editor de reglas se pueda acoplar con otras herramientas para la gestión de las reglas, o bien que posea determinadas funcionalidades que permitan completar el ciclo de gestión de las reglas de negocio.

1.4 Resultados alcanzados.

Una vez finalizado el proyecto se realizaron los siguientes entregables pactados con el cliente:

- Definición de reglas
- Clasificación de reglas
- Ciclo de vida de la regla
- Proceso de incorporación de reglas negocio dentro de una organización.
- Editor de reglas de negocio.

El Editor de Reglas está desarrollado utilizando los conocimientos adquiridos en la etapa de investigación, donde se conjugan y se acoplan los conceptos definidos para la definición de las reglas de negocio, su ciclo de vida y las tecnologías existentes investigadas.

El hecho de que no se haya encontrado en el mercado un software open source con este propósito y que además, suele ser una problemática frecuente el no poder independizar la edición de las reglas de un equipo técnico, hizo más valioso para nosotros la construcción de esta herramienta.

Finalmente podemos afirmar que se alcanzaron los objetivos trazados al inicio del proyecto y el alcance definido ha sido cubierto.

2. Marco conceptual

En esta sección se abarca la investigación teórica realizada acerca de cada uno de los conceptos que fueron los pilares de este proyecto de grado. Estos son la regla de negocio y las distintas formas para expresarla; el ciclo de vida de la misma visto desde distintos puntos de vista; y por último se estudió cuáles eran las formas de clasificarlas.

En esta búsqueda nos encontramos con diferentes definiciones las cuales las iremos exponiendo y finalmente en la sección 4 daremos una definición expresada por nosotros de los conceptos principales, resultado de la investigación previa realizada.

La definiciones encontradas sobre estos conceptos giran siempre alrededor de algunas personas que han realizado investigaciones y han estandarizado todo lo referido a la temática de reglas, es así que antes de continuar queremos dejar una reseña de alguno de ellos.

Ronald G. Ross: Autoproclamado y finalmente reconocido mundialmente como el padre de las reglas de negocio, es la persona que más ha publicado y que más conferencias brinda en lo referido a esta temática. Es miembro fundador del “Business Rules Group” por los años 80’. Este grupo se centró en cómo las reglas de negocio podían implementarse directamente sobre las tecnologías de la información.

Terry Moriarty: Formado en diversas ramas referentes a carreras en Sistemas de Información, ha desarrollado una metodología que integra el análisis de reglas con la gestión del negocio para poder así abordar las principales preocupaciones a nivel empresarial.

Barbara von Halle: Ha escritos libros de referencia en el área del manejos de las reglas de negocio y ha trabajado en consultorías desde los años 80’.

Tony Morgan: Autor de importantes títulos de referencia, entre otros Business Rules and Information Systems.

En la subsección 2.1 estaremos presentando la investigación realizada referente a la definición de Regla de Negocio, hablaremos posteriormente de Ciclo de Vida de las reglas, tanto a nivel de una organización como de la regla de negocio en sí. Más adelante se presentará lo investigado acerca de la Clasificación de las reglas. Luego, expondremos uno de los conceptos que tiene más relevancia en el trabajo realizado en este proyecto, el Estándar para la expresión de reglas de negocios en lenguaje natural. Por último estaremos hablando de algunas herramientas que tienen soporte a las reglas de negocio.

2.1 Regla de Negocio.

En esta subsección avanzaremos sobre el tema de las reglas de negocio, qué son y qué no son, y además como se identifican.

2.1.1 *¿Qué es una regla de negocio?*

El concepto entorno al cual gira este proyecto de grado es la Regla de Negocio. Esta podría ser considerada como la unidad atómica de nuestro trabajo. Es por eso que antes de interiorizarnos en otros conceptos más abstractos, creímos conveniente buscar una definición para la misma. Exponemos entonces algunas definiciones de regla de negocio realizadas por algunas organizaciones y expertos que han marcado el camino a lo largo de los años.

"*Una regla de negocio es una restricción sobre el negocio*". [1]

"*Es una declaración explícita que estipula una condición que debe existir en un ambiente del negocio cuya información fue extraída de ese ambiente y debe ser consistente con las políticas del negocio.*", Daniel Appleton, citado en el libro de Moriarty. [1]

"*Son sentencias en lenguaje natural que describen requerimientos de datos para los usuarios del negocio*" - Barbara von Halle y Alice Sandifer, citadas en el libro de Moriarty. [1]

"*Una regla de negocio puede ser considerada como un requerimiento del usuario que no es expresada en un procedimiento ni en una forma técnica. Una regla de negocio representa una declaración sobre el comportamiento del negocio.*" [2]

"*Una regla de negocio es una declaración que define o restringe un aspecto del negocio. Su intención es afirmar la estructura del negocio o controlar e influir sobre el comportamiento del mismo.*" [3]

"*Una regla de negocio es una directiva que tiene por objetivo influir o guiar el comportamiento del negocio. Tales directivas existen en apoyo de la política empresarial, que se ha formulado en respuesta a los riesgos, amenazas y oportunidades.*", Business Rules Group, 2000.[4]

"*Una pieza atómica de lógica de negocio reutilizable, que se especifica de forma declarativa*" [5]

"*Una Regla de Negocio es una frase compacta sobre algún aspecto del negocio que puede expresarse en términos directamente relacionados con el negocio utilizando un lenguaje simple y no ambiguo accesible para todas las partes interesadas: desde el propietario del negocio hasta el arquitecto de software pasando por el analista de negocio.*"[6]

De las definiciones expuestas anteriormente podemos afirmar que las reglas de negocio son restricciones, condiciones, requerimiento de datos, políticas de negocio, prácticas de negocio, son declaraciones que definen o restringen, son requerimientos del usuario, afirman, controlan e influencian. La mayoría de las definiciones de expertos desde los 90' mencionan a las reglas con tres categorías básicas: términos, hechos y reglas; ya que el conocimiento siempre viene en un término (definición), en un hecho (tipo) o una regla y un término es la palabra o frase que tiene un significado específico para el negocio. Y un hecho afirma una asociación entre dos o más términos, es decir, se expresa una relación entre las condiciones.

En general existe una idea de lo que es una "Regla de Negocio", literalmente, es aquello que usamos para operar un negocio. Son vistas como guías que determinan cómo se lleva el día a día de las operaciones. Sin reglas se estaría en una situación en la que cada decisión se resuelve en el momento, eligiendo alternativas caso a caso. Este modo de operar suele ser muy lento, costoso y puede llegar a generar resultados inconsistentes.

El término de “Regla de Negocio” varía dependiendo del ámbito en el que se está utilizando, por ejemplo, para un especialista en procesos de negocio puede significar un conjunto de requerimientos asociados a proceso. Para un especialista en Base de Datos, puede ser un requerimiento específico que se satisface mediante la restricción de algún campo en una tabla y para un Analista de Negocio puede ser entendido como la “manera de hacer las cosas”. Por lo que en la subsección 2.1.2 veremos lo que no es una regla de negocio.

2.1.2 *¿Todas son reglas de negocio? No.*

Las Reglas de Negocio no son software, estas suelen ser implementadas en el software, pero eso es solo una opción de muchas para implementarlas. Se debe entender que como su nombre lo indica, son parte del negocio y no de alguna plataforma particular de hardware o software.

No todas las reglas son Reglas de Negocio, por ejemplo:

“Un login debe ser cancelado si el usuario ingresa el password incorrecto más de 5 veces.”

Los Ingenieros en Sistemas ven como una pérdida de tiempo y antinatural separar las complejidades del negocio de las del sistema de software. Dicen, que alimenta la idea errónea de que hay una solución de negocio independiente de la solución del sistema. Esto no es correcto, veremos más abajo que sí hay diferencia entre las reglas de negocio y las reglas de un sistema de software.

Ronald Ross presenta tres condiciones que se debe cumplir para que una regla sea realmente una Regla de Negocio [7]:

1) La regla debe representar una acción concreta, al leerla se debe entender exactamente qué se debe o no hacer. Por ejemplo “la seguridad es lo primero”, este ejemplo es no representa una acción, no se sabe qué se debe o no hacer. Sin embargo en este ejemplo “en una construcción se debe usar casco protector”, se puede interpretar que se debe hacer.
2) La regla debe ser sobre el negocio, no se trata de un sistema de datos ni registros que dan soporte al negocio, ni tampoco es una plataforma utilizada para implementar dicho sistema.
3) La regla debe ser expresada en el lenguaje del negocio, no en el lenguaje de una base de datos o de un sistema de información.

En el ejemplo del login, la regla cumple las condiciones 1 y 3 pero no la 2, no es una regla sobre el negocio.

La declaración: "Sin tarjeta de crédito, no se aceptan órdenes" en sí es una regla de negocio. Si la expresamos de la siguiente manera: "Si [condición], entonces [la acción]" también es una regla, pero en un formato diferente, es posible que una persona del negocio ya ni la reconozca. Para ser una Regla de Negocio, una persona del negocio debe ser capaz de reconocerla como tal, ahora en este nuevo formato es una regla de negocio interpretada para su implementación, para alguna plataforma o sistema. Las reglas de sistemas o plataformas no siempre surgen de las reglas de negocio, sino que también pueden surgir de forma independiente en el contexto específico de un determinado diseño o de la plataforma.

Ahora consideremos el ejemplo: “Una persona que ha iniciado sesión en el sistema como un administrativo puede solicitar datos de los clientes de la base de datos de clientes sólo desde el servidor local.” Esta regla cumple con el criterio 1, representa una acción, pero no cumple con el 2 y el 3, no es una regla de negocio pero no significa que no sea importante para el negocio.

Un buen ejercicio a practicar para saber si una regla es de negocio o no, es pensar en que sucedería si tiramos el sistema y comenzamos a gestionarnos aunque sea en papel y lápiz, ¿la regla sigue siendo importante en el funcionamiento de la empresa? Es evidente que la regla anterior no pasa la prueba. Sin embargo la regla: “Un administrador puede ser informado sólo sobre clientes de su barrio local” es necesaria independientemente de los sistemas.

Muchas veces surge la duda de si los requerimientos son reglas de negocio, podemos decir que algunas veces sí y otras no. Un requerimiento de software es algo que se quiere o necesita, debe ser interpretado (en algún diseño) antes de que el desarrollador de software pueda hacer algo con él, el requerimiento de software no siempre cumple con la condición de reglas de negocio de que pueda ser interpretada como una acción.

Los profesionales en base de datos tienden a igualar lo que son “Restricciones de Integridad” con Reglas de Negocio. Generalmente es aceptado que cualquier restricción de integridad puede ser violada, por eso se definen, para prevenir que pase eso. Consideremos los siguientes ejemplos:

1) *Un conductor que ha sido detenido no debe tener en su posesión un coche de alquiler*.

   Se trata de una regla operativa (conducta). Se puede violar en la vida real por una variedad de razones, por ejemplo un robo. Esto muestra una regla de negocio real, es relevante para el negocio sin ninguna mención de un sistema.

2) *Un coche de alquiler siempre tiene un cliente*.

   Esta es una regla de negocio estructural (definición). El sentido de esta regla es que algo simplemente no es un coche de alquiler sin un cliente que lo alquile. Realmente no se puede violar una regla que establece si algo es o no es por definición. Esto muestra también una regla de negocio real, que es de nuevo relevante para el negocio sin ninguna mención de un sistema.

3) *Un registro del cliente puede incluir como máximo un número de teléfono*.

   Este caso es diferente. Generalmente no hay tal regla en la vida real o de negocios, alguien ha transformado una decisión de diseño de la base de datos en forma de regla. Un sistema puede evitar violaciones de los datos, pero no en la vida real. Esta es una regla de sistema, no una regla de negocio.

De los ejemplos anteriores, el tercero es una restricción de integridad, y con algo de trabajo se pueden transformar también las dos primeras. Pero para eso habría que transformarlos a algo más allegado a los datos, y ahí está la diferencia, las Reglas de Negocio tienen que ver con la conducta y decisiones de la gente, mientras que las restricciones de integridad tienen que ver con los datos.

2.1.3 *¿Cómo identificar Reglas de Negocio?*

Existen distintas metodologías para detectar las reglas de negocio en una organización a partir de distintas fuentes de información, dependiendo de cuál sea el dominio de aplicación del sistema. Estas fuentes podrían ser desde documentos, regulaciones, estructura organizativa hasta cualquier recurso que contenga información sobre los conceptos principales del negocio. Con este fin existen metodologías como PROTEUS [15] donde se detalla un conjunto de pasos a seguir para facilitar la detección de reglas de negocio y lenguajes de alto nivel de abstracción, RuleSpeak [14] o el estándar SBVR [16] donde se establecen pautas para expresar las reglas lo más concisas posibles tratando de eliminar la ambigüedad.

Cuando se van a identificar las reglas de negocio de una organización debemos dirigirnos a todo el abanico de reglas existente y no sólo a aquellas que puedan ser soportadas por los sistemas de información. Por ejemplo la regla “está prohibido fumar en todo el edificio” es una regla de negocio que lógicamente no estará implementada en ningún sistema de información de la empresa.

Para proceder a la recopilación de las Reglas de negocios un buen punto de partida son los principios denominados “The business rule approach” escritos por Ronald Ross, estos son:

1) Deben ser explícitas y escritas.
2) Expresadas en términos sencillos.
3) Existen independientemente de los procedimientos y workflows (ej: modelos).
4) Se construyen a partir de hechos, éstos se definen a partir de conceptos, los que a su vez se representan por medio de términos (ej: glosarios).
5) Guían o influencian el comportamiento conforme a una forma pre-establecida.
6) Son motivadas por factores de negocios identificables e importantes.
7) Son accesibles a las partes autorizadas (ej: tienen dueños).
8) Están en una fuente única (ej: repositorio de reglas).
9) Son especificadas por las personas que tienen directa relación con ellas y que poseen el conocimiento relevante (ej: los usuarios claves).
10) Son gestionadas, administradas (ej.: son parte de la Gobernabilidad de los Procesos de Negocios).

2.2 Ciclo de vida de la regla de negocio a nivel Organizacional.

El ciclo de vida de las reglas de negocio de una organización habla de la planificación y la estrategia que se debe llevar a cabo a la hora de introducir la gestión de las mismas. Lo que vamos a presentar a continuación está basado en un paper llamado “A Lifecycle Approach towards Business Rules Management” [8] que está enfocado en esta temática.

Los autores realizaron este estudio mediante entrevistas y recolección de información en donde se pudo revelar que el manejo del ciclo de vida de las reglas de negocio en una organización tiene los siguientes estados dentro de un ciclo: planificación, captura, organización, autoría, distribución, testeo, aplicación, mantenimiento.

La tendencia a la automatización de las reglas de negocio en las organizaciones va en aumento según plantea este estudio, así como los beneficios que esto brinda. Es así que la necesidad de manejar la gestión de las reglas de negocio con una perspectiva más amplia también va en aumento.

Es importante hacer notar que esto no es un proceso de desarrollo, ya que pone énfasis en el proceso de cómo incorporar reglas de negocio en una organización.

2.2.1 Diagrama del ciclo

La metodología planteada ha sido llamada Business Rule Management Life Cycle (BRMLC). Ésta consta de 8 etapas que pasaremos a describir más adelante, las mismas pueden ser clasificadas en tres grupos.

1) Planificación de una estrategia para incorporar reglas de negocio en la organización.
2) Relevarlas a partir de varios escenarios.
3) Organizarlas para poder empezar con su creación e implementación.

La figura 1 ilustra una síntesis realizada a partir de la metodología de investigación utilizada, lo que sería el BRMLC (Ciclo de vida del manejo de reglas de negocio). Se muestran las ocho etapas y a que grupo pertenece cada una.

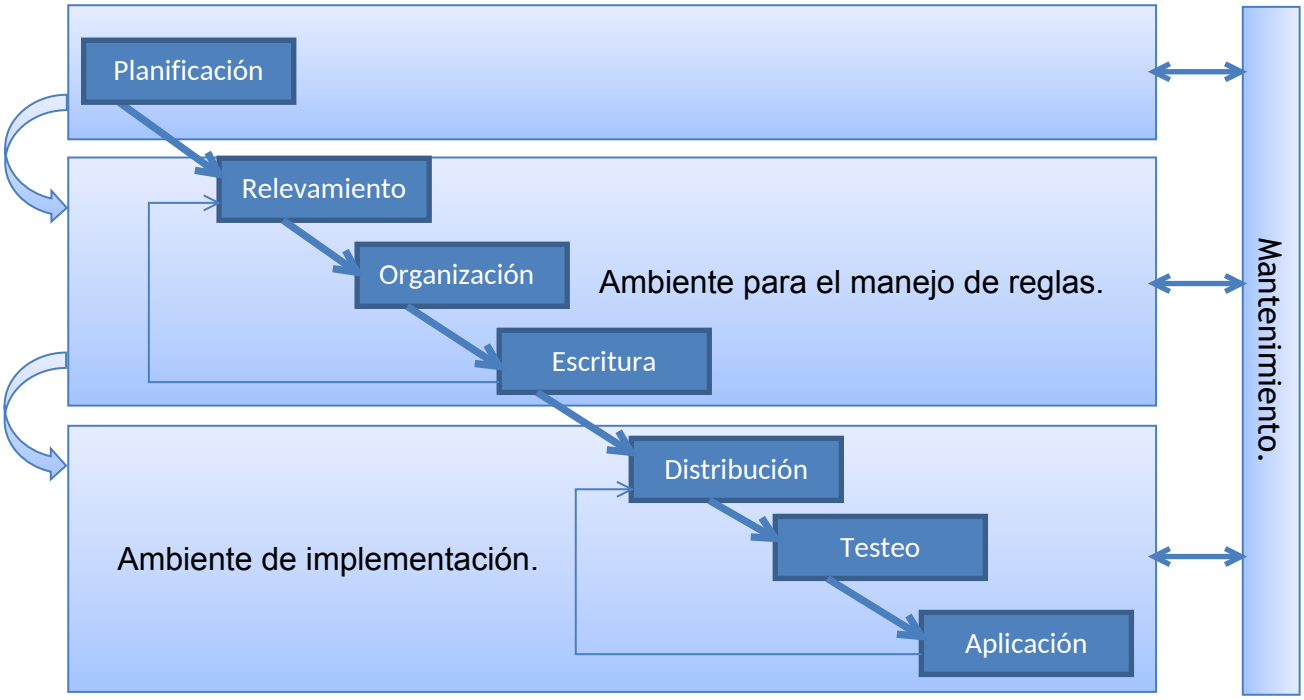

Figura 1: Etapas del proceso.

En la tabla 1 “Tabla de las etapas” haremos una breve descripción de cada una de las etapas y mencionaremos cuales son los actores que participan en cada una de ellas.

Etapa	Descripción	Actores
Planificación.	Planificación y organización macro acerca de cómo se llevará a cabo la puesta en marcha del manejo de las reglas de negocio en la organización, desde el modelo de datos hasta las tecnologías a utilizar.	Analistas de negocio y Especialistas técnicos.
Relevamiento.	Relevamiento de reglas en base a diferentes fuentes, estas pueden ser documentación de la organización e información del personal para identificar las posibles reglas de negocio.	Especialistas técnicos, Analista de negocio, Expertos en el dominio, Empleados con mucha antigüedad.
Organización.	Después de identificar las posibles reglas de negocio se verifica que éstas realmente sean reglas de negocio. Se organiza dónde y cómo van a ser implementadas (sistemas, procesos, motores de reglas). El output de esta etapa debería ser la regla formalmente creada.	Analistas el negocio, Especialistas técnicos, Expertos en el dominio.
Escritura.	Las reglas creadas en la etapa anterior son transferidas al repositorio de reglas de negocio. El repositorio es diseñado y manipulado por los analistas del negocio de la organización y se usa vocabulario del negocio.	Personal del negocio, Personal de la gerencia que autorice decisiones, Expertos en el dominio. Analista de negocio y especialistas técnicos.
Distribución.	Se distribuyen las reglas del repositorio a los sistemas donde serán implementadas. Se obtiene una solución automatizada para pasar del repositorio a las implementaciones.	Proveedores de BRMS, Especialistas técnicos de la organización, Personal técnico de la gerencia.
Testeo	Se asegura la interoperabilidad entre el repositorio y el entorno de ejecución	Expertos en el dominio, Analistas del negocio, Especialistas técnicos, Proveedores de BRMS.
Aplicación	Queda operativo el sistema con las reglas de negocio.	Personal de la gerencia, Analistas de negocio, Especialistas técnicos.

Mantenimiento	Este estado es horizontal a todos los sistemas anteriores. Se debe realizar el mantenimiento del sistema, verificar los cambios en las reglas, controlar que todo se mantenga consistente en el repositorio y en los diferentes ambientes.	Expertos en el dominio, Analistas de negocio, Especialistas técnicos.

Tabla 1: Tabla de las etapas del proceso.

Los hallazgos sugieren que el ciclo de vida que ha surgido de la gestión de las reglas es distinto a los ciclos de vida de gestión de conocimiento tradicionales. Los efectos primarios del enfoque del ciclo de vida presentado incluyen:

- Separación del manejo de reglas del entorno de desarrollo.
- Uso independiente de un repositorio de reglas.
- Desplazamiento del control de la edición de las reglas de negocio desde los técnicos informáticos hacia los Analistas de negocio.
- Alineamiento de las puestas en producción de las reglas de negocio con iniciativas similares en toda la empresa.

En el Anexo I (Estados de ciclo de vida a nivel organizacional) se podrá ver una descripción más completa del significado de cada uno de los estados.

2.3 Ciclo de vida de la regla de negocio.

En la sección 2.2 presentábamos el ciclo de vida a nivel del proceso de una organización para incorporar reglas de negocio, ahora expondremos todo lo referente al ciclo de vida de la regla de negocio en sí.

El enfoque que veremos a continuación es elaborado por Primatek [9]. El mismo no se liga con ningún sistema por lo que puede tomarse como algo genérico y por esta razón hemos decidido presentarlo.

Primatek plantea que las reglas de negocio tienen vida propia, desde que nace hasta que deja de existir se debe saber cómo ha sido esa vida.

A grandes rasgos se pueden destacar cuatro grandes fases, ellas son:

- Nacimiento y desarrollo: Una regla es creada y definida.
- Validación: La regla debe ser testeada y validada.
- Vida útil: La regla se ejecutará mientras esta tenga sentido.
- Retirada: Cuando la regla deja de tener sentido es quitada de ejecución.

Bajando un poco más el nivel podemos ver en la figura 2 el ciclo de vida definido por Primatek expresado mediante una máquina de estados. También se puede ver cuáles son los roles que llevan a cabo la transición entre un estado y otro.

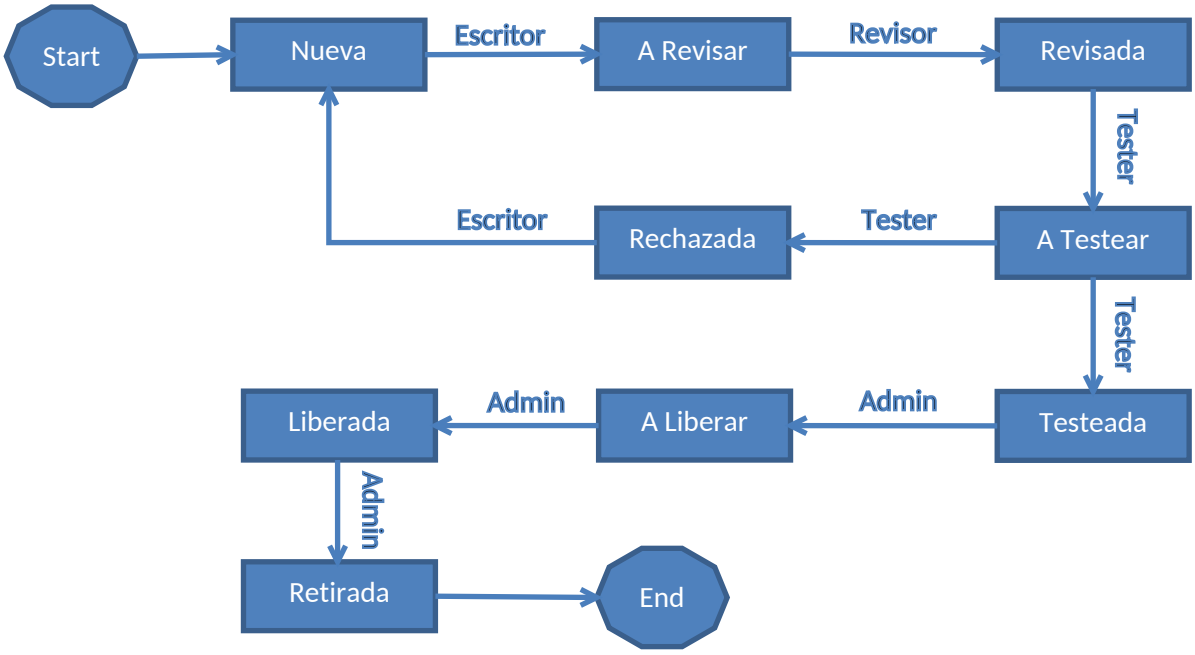

Figura 2: Máquina de estados ciclo de vida de Primatek.

Estos estados representan lo siguiente:

- Nueva: La regla es creada y puede ser modificada.
- A Revisar: El escritor de la regla completa su trabajo inicial y un revisor puede verificar que la regla representa fielmente el negocio. El revisor puede ser un Analista de negocio al igual que el que la escribió inicialmente.
- Revisada: Ha sido revisada y está pronta para pasar al siguiente paso.
- A Testear: Está lista para ser testeada y permanecerá en este estado hasta que el tester acepte o rechace la regla. Este testeo es a nivel integración dentro del sistema, no a nivel lógico.
- Rechazada: El resultado del testeo ha sido insatisfactorio y la regla debe ser reescrita.
- Testeada: El resultado del testeo ha sido satisfactorio.
- A Liberar: Estado anterior a que la regla pase a producción.
- Liberada: Pasa a ser parte de un conjunto de reglas en un ambiente de producción.
- Retirada: La regla fue desplegada en un ambiente de producción pero por alguna razón del negocio esta pasa a ser obsoleta.

Los roles que se presentan en este ciclo de vida son los siguientes:

- Escritor: Encargado de crear y modificar la regla en caso de que sea necesario.
- Revisor: Encargado de haya sido escrita correctamente y que esta representa fielmente lo que se quiso expresar a nivel de negocio. Trabajará junto al escritor hasta que se logre este cometido.
- Tester: Encargado de testear la regla hasta que esta quede integrada dentro del sistema de forma correcta.
- Admin: Encargado de poner la regla en el ambiente de producción una vez que esta ha sido marcada para liberar. Encargado de quitarla del ambiente de producción cuando sea pertinente.

En el Anexo C (Ciclo de vida de las reglas de negocio) se puede encontrar más información acerca del planteo anterior, además se puede encontrar una implementación de este ciclo utilizando la herramienta de IBM llamada iLog y otras definiciones de ciclos de vida de reglas.

2.4 Clasificación de las reglas de negocio.

La utilidad de un sistema para la clasificación de reglas de negocio depende de quienes van a ser los que las utilicen. Un esquema de clasificación es útil para el descubrimiento e identificación de las reglas, análisis de las mismas, e incluso para diseñar reglas basadas en la clasificación.

No hay una clasificación universal de reglas, hay muchas clasificaciones, algunas realizadas por organizaciones y otras por individuos particulares. Incluso se llegan a realizar para cierto negocio determinado. Particularmente para el área de seguridad social no se encontró ninguna clasificación.

Algunos sistemas de clasificación de reglas están destinados a públicos como desarrolladores de software. Éstos pueden contener clasificaciones que diferencian a las reglas de validación de reglas de integridad, ya que normalmente hay una diferencia entre estos dos tipos desde una perspectiva de programación. Un programador puede escribir las reglas de validación, pero las reglas de integridad son ejecutadas en el DBMS. Un ejemplo de clasificación para desarrolladores de software es la clasificación de Versata [10]. Las categorías de este tipo de clasificación dependen del software al cual están dirigidas, comúnmente incluye las categorías:

- Restricción: establecen que es lo que no está permitido. Generalmente son valores mínimos aceptados, valores inválidos o valores que deberían ser obligatorios.
- Derivación: definen o calculan como se obtiene información nueva a partir de información existente.
- Validación: reglas que mantienen consistente los datos en el sistema. Por ejemplo atributos obligatorios, tipos válidos, mínimos, máximos, etc.
- Integridad: Son reglas que representan y controlan las relaciones entre entidades. Son asociaciones de base de datos, multiplicidad, restricciones, triggers, etc.
- Acción: estas reglas expresan como se debe comportar el sistema en determinadas situaciones, especifican que debe hacer el sistema automáticamente.
- Presentación: reglas que permitan que contenido dinámico pueda ser presentado a los usuarios. Definen que cosas deben aparecer y en qué orden.

Otros sistemas de clasificación de reglas están destinados a públicos como los que manejan las bases de datos. Pueden contener clasificaciones distintas para reglas según si es para una tabla o columnas, ya que por ejemplo pueden tener diferentes implicaciones de rendimiento. En el año 2000, Chris Date presento una clasificación para este propósito. Además también se pueden clasificar a las reglas dividiendo las reglas según entidades, reglas de atributos, y reglas de las relaciones. Dicho sistema de clasificación es más intuitivo para los analistas de datos y modeladores de datos, Chris Date presenta una clasificación de este estilo también [11].
Un enfoque más orientado al negocio en sí mismo no debería distinguir entre quien releva las reglas, o si las reglas son simples o complejas. Esto nos lleva a otra clasificación de reglas destinada a los analistas del negocio (no profesionales técnicos) y específicamente para su uso

durante el descubrimiento de reglas y su validación. Entre los ejemplos de esas clasificaciones se encuentran entre varios a Ronald Ross (1997) y al BRG (Business Rules Group) (2000) [12][13]. En general este tipo de clasificaciones incluye categorías como:

- Cálculo: es una afirmación que proporciona un algoritmo para llegar a un valor. Por ejemplo: El total de una orden se calcula como la suma del costo de cada producto de la orden más impuestos.
- Inferencia: es una sentencia que pone a prueba las condiciones de la regla y al encontrarlas verdaderas establece la verdad de un hecho nuevo. Por ejemplo: Si un cliente no tiene facturas pendientes de pago, el cliente es un cliente preferencial.

- Guías: es una pauta que no fuerza una circunstancia, simplemente advierte algo y el ser humano toma la decisión. Por ejemplo: Un cliente no debería tener más de 10 órdenes abiertas al mismo tiempo.

- Acción: es una afirmación completa que corrobora las condiciones de la regla y al encontrar que son verdaderas inicia un nuevo evento. Por ejemplo: Si el pedido de un cliente es válido, a continuación iniciar el proceso de pedido.
- Restricciones: son condiciones que restringen comportamientos o especifican autorizaciones. Por ejemplo: El monto total en dólares del pedido del cliente no debe ser mayor al límite de crédito de dicho cliente.

Para ver los distintos tipos de clasificaciones diríjase al anexo Clasificación de Reglas de Negocio.

2.5 Estándar para la expresión de reglas de negocios en lenguaje natural.

En esta sección presentaremos en la sección 2.5.1 los problemas que surgen al expresar las reglas de negocio y en la sección 2.5.2 hablaremos sobre RuleSpeak que brinda una solución a los problemas planteados.

2.5.1 El problema de expresar las reglas de negocio

El Business Rules Group (BRG) es una organización que se encarga de definir los principios que deben seguir las reglas de negocio y establecer estándares para expresarlas y definirlas, el estándar definido está alineado y aceptado por el Object Management Group (OMG).

El BRG establece que la declaración de las reglas de negocio debe ser expresada en lenguaje natural, consideradas como requerimientos, siendo estas una expresión de cómo se opera en el negocio. Define a la regla como una directiva destinada a influenciar o guiar el comportamiento del negocio, que apoya una política que ha sido formulada en respuesta a una oportunidad, amenaza, fortaleza o debilidad. Para preservar el punto de vista desde la perspectiva del negocio las reglas deben ser entendidas por la gente del negocio (además de los programadores). Es conveniente que las reglas se expresen en lenguaje natural, no solo para que las comprendan todos los involucrados sino también para independizarlas de la implementación o del lenguaje de programación utilizado.

Hoy en día la mayoría de los informáticos prefieren expresar las reglas de negocio como sentencias IF-THEN-ELSE, debido a que es más cercano al lenguaje de programación o al requerimiento del motor de reglas que utilizan. Sin embargo esto los aleja de la perspectiva del área de negocio que manejan.

Una sentencia IF-THEN en un lenguaje de programación no es una regla de negocio, es una sentencia condicional diseñada para tener control sobre las ramificaciones del programa. Esa sentencia se transforma en regla de negocio si está definida en una programación orientada a las mismas. Las reglas de negocio están diseñadas para resolver problemas complejos, requieren conocimiento sobre el negocio para poder tomar decisiones, las reglas pueden justificar sus conclusiones y decisiones ya que razonan sobre hechos o premisas usando deducciones o inducciones del negocio.

Hay una diferencia entre el lenguaje del negocio y del lenguaje informático tanto para expresarlas como para gestionar las reglas de negocio. Expresar las reglas usando if-then genera tres grandes separaciones respecto a la expresión de negocio. Por ejemplo, si tenemos la regla:

"La cantidad reclamada por un vendaje debe ser menor o igual al límite especial por un accidente si el solicitante está fuera del estado y el monto de la demanda es menor o igual a $500."

La gente de negocio se focaliza en que no se viole(n) la(s) condición(es), sin embargo el estilo con if-then se focaliza en la condición que causa la violación. Hoy en día la mayoría de los lenguajes de programación y motores de reglas requieren hacer ese cambio de mirar la lógica inversa.

En la regla del ejemplo:

Perspectiva del negocio La condición que no debe ser violada	La cantidad reclamada debe ser menor o igual al límite especial
Perspectiva de programación La condición que causa la violación	IF la cantidad reclamada es mayor o igual que el limite especial THEN

Para los agentes de negocio el foco está en la restricción a ser aplicada, eso es lo primero. Sin embargo, cuando se expresa con if-then se comienza con if y todas las condiciones. Esto se debe a que los motores de reglas y los lenguajes de programación miran primero si las condiciones se satisfacen o no para tomar alguna acción, haciendo que la regla no tenga significado real hasta llegar a la acción (si es que se llega).

Perspectiva del negocio La restricción es lo primero	La cantidad reclamada **debe ser** menor o igual al límite especial
Perspectiva de programación La condición es lo primero	**IF** el solicitante esta fuera del estado y la cantidad reclamada es menor o igual a $500…

La gente de negocio generalmente expresa las reglas asumiendo la respuesta por default de que si no se cumple se rechaza. En el estilo if-then no hay respuesta por default, cada acción o

respuesta debe ser especificada. Esto genera que los programadores critiquen a las reglas de negocio por no estar bien detalladas por la falta de respuesta a ciertas condiciones.

Perspectiva del negocio Se asume el rechazo por violación	-
Perspectiva de programación Se toma una acción si la regla es violada	IF … THEN acción

Resumiendo la regla quedaría:

Perspectiva del negocio	*La cantidad reclamada por un vendaje debe ser menor o igual al límite especial por un accidente si el solicitante está fuera del estado y el monto de la demanda es menor o igual a $500.*
Perspectiva de programación	*IF el solicitante está fuera del estado y la cantidad reclamada por un vendaje es mayor o igual al límite especial para un accidente y el monto de la demanda es menor o igual a $500 THEN establecer exoneración del pago.*

En la regla original la restricción era clara, en la regla transformada la restricción pasó a ser una condición más, perdiendo su importancia. En una regla con muchas condiciones (que es la mayoría), el verdadero sentido de la regla se pierde fácilmente.

La clave está en qué hacer si la regla es violada, la respuesta generalmente va a ser que depende de las circunstancias, sobre todo si se quiere considerar que las reglas sean re usables. En el negocio que una regla sea re usable significa que una misma regla puede ser usada en diferentes puntos del proceso de negocio de una empresa. En la regla del ejemplo se podría tener dos escenarios, la consulta de si corresponde pagar y la ejecución del reclamo.

Escenario	***Respuesta a la regla***
Consulta	Devuelve un mensaje al usuario.
Reclamo de pago	Se marca exoneración del pago.

La regla de negocio queda exactamente igual que antes, sin cambios, la regla permanece constante aunque la respuesta a la violación de la misma cambie según la circunstancia. Una regla puede tener múltiples respuestas.

Eso no ocurre cuando usamos el estilo if-then, ya que para cada respuesta selectiva se debe tener una regla y se deben agregar condiciones, en el ejemplo de la regla quedaría:

Escenario	***Regla IF-THEN***
Consulta	IF el solicitante está fuera del estado y la cantidad reclamada por un vendaje es mayor o igual al límite especial para un accidente y el monto de la demanda es menor o igual a $500 y el reclamo se presenta únicamente

	para consultas THEN mostrar "Lo sentimos, usted no estaría cubierto"
Reclamo de pago	IF el solicitante está fuera del estado y la cantidad reclamada por un vendaje es mayor o igual al límite especial para un accidente y el monto de la demanda es menor o igual a $500 y el reclamo se presenta por el pago THEN establecer exoneración del pago.

Cada condición del IF se trata por igual sin saber si alguna tiene especial importancia, no se distingue en cuál es la restricción principal y tampoco las diferentes circunstancias para la misma regla. Esto no solo oculta la verdadera intención de la regla sino que también aumenta la cantidad de las mismas, en este caso una regla de negocio ya se transformó en dos, generando problemas a gran escala.

El Business Rules Group definió un estándar para modelar la semántica del negocio llamada Semantics of Business Vocabulary and Business Rules (SBVR). Es un medio para describir la estructura del significado de las reglas en lenguaje natural definidas por los analistas de negocio. Con el SBVR se realiza una "formulación semántica", que son estructuras para brindar significado a las reglas. Es para esa formulación que el SBVR en vez de utilizar un lenguaje formal que traduzca el significado, define un vocabulario y un esquema XML para comunicar la semántica. SBVR no apunta a resolver las necesidades de los motores de reglas y BRMS, se enfoca en las reglas independientemente de las posibilidades de automatizarlas en algún sistema.

2.5.2 RuleSpeak: Una guía para definir las reglas

Dados los problemas mencionados en la sección 2.5.1, Ronald G. Ross miembro del BRG creó una guía, llamada RuleSpeak [14], la cual es referenciada en el estándar SBVR, para expresar las reglas del negocio de forma concisa y amigable para el personal de negocio, estableciendo pautas a seguir para una expresión concisa tratando de eliminar la ambigüedad. No es un lenguaje o sintaxis en sí mismo, sino más bien un conjunto de mejores prácticas para aportar claridad y consistencia en la escritura de reglas del negocio, tanto para comunicación entre el mismo personal de negocio como en la comunicación desde negocio con el personal de tecnología. El principal objetivo es expresar y retener de forma efectiva los criterios de decisión y el saber hacer del negocio, evitando en la definición de la regla ambigüedades que se presentan en el lenguaje natural.

A la hora de expresar las reglas en lenguaje natural se debe evitar la ambigüedad, las plantillas van a acotar la obligación y la prohibición, ya que pueden expresarse con todos sus grados y matices en lenguaje natural de forma afirmativa o negativa.

Se emplea a menudo el "**debe**" en las formas más elevadas de guía de negocio (es decir, el lenguaje de las leyes, regulaciones, contratos, políticas y demás). En RuleSpeak, se prefiere el "tiene que" debido a que el "debe" está contaminado por "debe de". El "debe de + infinitivo" indica probabilidad no obligación como indica "debe + infinitivo". Esta ambigüedad es inoportuna cuando se expresan reglas.
Cualquier regla del negocio puede declararse usando "**debería**" y "**no debería**" en vez de "tiene que" y acompañando a "está/tiene prohibido" ("debería estar/tener prohibido"). Esta

elección refleja una asunción o decisión sobre el nivel de cumplimiento/refuerzo de la regla. En RuleSpeak, se prefiere documentar cómo estrictamente va a ser implementada la regla cuidando la expresión misma. Por lo tanto siempre se prefiere la palabra clave "tiene que" o "está/tiene prohibido" en vez de "debería" o "debería estar/tener prohibido".

También es posible usar de muchas maneras la palabra "**puede**", la ambigüedad resultante es particularmente inoportuna cuando se expresan reglas. Por ejemplo "*Las mujeres pueden ingresar en el Ejército*" tiene hasta 6 interpretaciones distintas:

- Lo tienen permitido/están autorizadas.
- Son capaces.
- Están obligadas.
- Se las anima a ello.
- En el futuro, es posible que lo hagan.
- Advertimos de su posible ingreso.

Es debido a esto que en RuleSpeak se desaconseja el uso de "puede"/"no puede" en todas las reglas del negocio. Por tanto siempre usaremos la palabra clave de sentencia de permiso "**está/tiene autorizado**".

La plantilla de RuleSpeak presenta palabras clave de una regla, estas palabras deben usarse a la hora de expresar las reglas de negocio. Hay dos tipos de reglas, las que restringen algo eliminando algún grado de libertad, y las denominadas *sentencias de permiso*, que no eliminan ningún grado de libertad simplemente clarifican que algo está permitido o no es obligatorio.

Las palabras clave para eliminar algún grado de libertad son:

- "Tiene que"
- "Tiene prohibido" o "está prohibido"
- "Ha de ser"

Las palabras clave de permiso son:

- "Tiene autorizado" o "está autorizado"
- "No tiene por qué"

Cada sentencia guía que exprese una regla debe seguir las reglas gramaticales del lenguaje, las oraciones deben incluir un sujeto aunque en ocasiones puede ser un sujeto implícito. Lo más común es que el sujeto aparezca al principio de la oración, y eso es lo que recomienda RuleSpeak.

Otra cosa que se recomienda es referenciar a una instancia, una instancia es algo único en contra posición a una clase de cosas. Por ejemplo: El indicativo "inflamable" tiene que ser mostrado en cualquier tanque que contenga combustible. En el ejemplo "inflamable" es una instancia de una clase de cosas, en este caso indicativos, sin las comillas podría indicarse que el indicativo es el inflamable. Para diferenciar una instancia de una clase se utilizan comillas simples en el término.

La mayoría de sentencias incluyen alguna condición indicando las circunstancias bajo las cuales hay que hacer cumplir la regla. A menudo esta condición se expresa siempre mediante un

condicional. Debería empezar únicamente con “si”, evitando en lo posible otras opciones (“cuando”, “mientras”, “donde”…) que pueden crear.

Ejemplo: Todo envío tiene que ser asegurado si el valor del envió es mayor que $500.

Algunas cualificaciones se pueden expresar sin usar “si”.

Ejemplo: Cualquier empleado retirado tiene prohibido estar asignado a un proyecto.

Aquí la cualificación “retirado” se usa para modificar directamente a “empleado”. Esta forma natural de cualificación es perfectamente aceptable.

Ejemplo: Cualquier pedido ordenado por un cliente nacional tiene autorizado el no incluir los gastos de envío en el precio total.

Aquí la cualificación “ordenado por un cliente nacional” modifica de forma natural “pedido”, siendo también aceptable.

La palabra “si” indica que la cualificación es continúa en el tiempo. La palabra “cuando” debería ser usada si la sentencia guía se aplica sólo en cierto momento temporal.

Ejemplo: El total a pagar correspondiente a la matrícula de 1 estudiante para 1 semestre tiene que ser $5000 cuando el estudiante se registre para ese semestre.

Como hemos comentado, la regla del negocio se aplica solo en el momento en el tiempo en que un estudiante se matricula para algún semestre. Asumimos que en cualquier otro momento temporal, la regla no tiene por qué aplicar.

Ejemplo: Todo producto está autorizado a ser devuelto si se proporciona el ticket de compra.

Esta sentencia expresa un consejo, literalmente no dice nada acerca de si un producto está autorizado a ser devuelto si *no* se proporciona el ticket. Probablemente, la intención de negocio es no permitir devoluciones sin una prueba de la compra. Asumiendo que este sea el caso, la palabra clave de regla “solo” tiene que usarse tal y como se indica en el siguiente ejemplo, transformando el consejo en una regla del negocio.

Ejemplo: Todo producto está autorizado a ser devuelto <u>solo</u> si se proporciona el ticket de compra.

Las cualificaciones en sentencias de permiso basadas en la plantilla básica “…tiene/está autorizado…” expresadas sin él “si” no son tan fáciles de malinterpretar. Por lo que son aceptables si se usan cuidadosamente.

Ejemplo: Cualquier empleado retirado <u>está autorizado</u> a trabajar a tiempo parcial en Montevideo.

Esta sentencia tiene que interpretarse como un consejo. No dice nada sobre empleados no retirados, o sobre empleados que no estén trabajando a tiempo parcial, o empleados que no estén trabajando en Montevideo. Si hay alguna otra regla de negocio que necesita cubrir estas

otras circunstancias tiene que ser expresada de forma separada, tal y como ilustra la siguiente sentencia de regla.

Ejemplo: Cualquier empleado retirado tiene prohibido trabajar a tiempo parcial fuera de Montevideo.

Si algo no es una regla del negocio bajo ciertas circunstancias, la mejor plantilla básica para expresar consejo es normalmente "… *no tiene por qué* …".
Ejemplo: La comprobación de crédito de cualquier pedido no tiene por qué ser solicitada si el importe del pedido es menor de $1000.

Observemos que la sentencia de permiso deja abierta la cuestión de si la comprobación de crédito es obligatoria para pedidos de $1000 o más. Si se requiere esta comprobación de crédito, se tiene que establecer una regla del negocio por separado. Lo que debemos recordar sobre expresar guías es que nada es requerido (todo está permitido) a menos que una regla del negocio elimine de forma explícita algún grado de libertad.

Se puede estudiar la guía completa de Rulespeak en el anexo correspondiente, para el marco del proyecto de grado se han elegido estas plantillas para expresar las reglas en lenguaje natural.

2.6 Herramientas para soporte a las reglas de negocio.

Un BRMS (Business Rules Management System) que en español significa sistema de gestión de reglas de negocio, es justamente un sistema que permite definir, mantener, reutilizar, depurar y evaluar reglas de negocio.

Permiten el manejo de reglas de la forma 'IF-THEN' para ser ejecutadas por un motor de reglas. Por ejemplo, los triggers o las restricciones de integridad de una base de datos también pueden ser vistas como reglas de negocio, pero sin embargo, este tipo de reglas no son controladas por un BMRS, por lo menos en la actualidad.

Los principales componentes y funcionalidades así como la interacción entre los mismos que comparten todos los BRMS se ilustran en la figura 3.

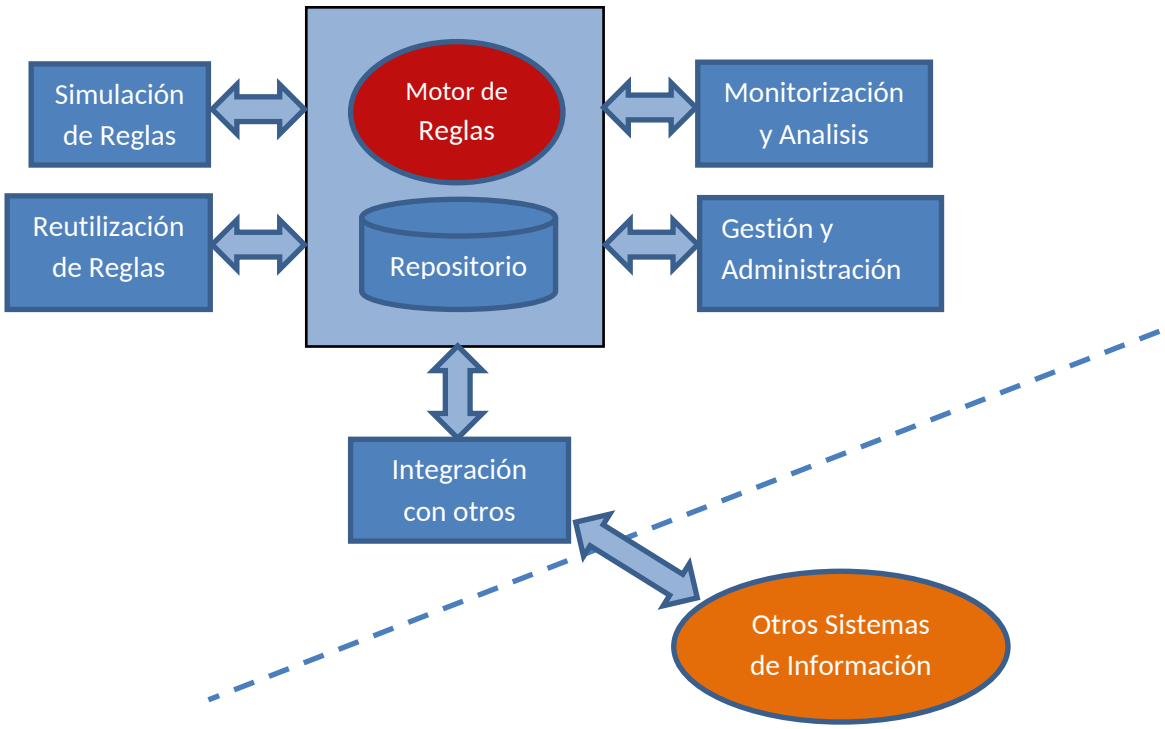

Figura 3: Componentes de los BRMS

A continuación conoceremos la descripción de cada uno de estos componentes:

Repositorio para la gestión de la configuración: Sirve para almacenar las distintas versiones de las reglas de negocio, se lleva un control de los cambios que se han producido en las reglas así como el registro de que usuario fue el que efectuó dichos cambios.

Motor de reglas: Ejecuta y gestiona las reglas de negocio. Los motores además de sustentar el uso de reglas, manejan prioridades de ejecución, exclusión mutua, precondiciones, entre otras funcionalidades. Los motores más utilizados son aquellos con reglas de inferencia que manejan el comportamiento con sentencias IF-THEN, éstas son reglas invocadas por los usuarios. Los otros motores más utilizados son aquellos que trabajan con tipos de reglas enfocadas en eventos, en las que el motor reacciona a eventos o procesos definidos, esta reacción es automática.

Integración con las aplicaciones de desarrollo de la organización: Es necesario integrar las reglas de negocio desarrolladas con el software desarrollado en la organización. Para realizar esta integración los BRMS proporcionan entornos de desarrollo basados en modelos los cuales reducen la cantidad de código generado por los desarrolladores a la hora de introducir las reglas en los sistemas de la organización.

Simulación de las reglas: La verificación y validación de una regla de negocio juega un rol tan importante como lo juega cualquier otra pieza que conforma un sistema de software, por lo que se brinda soporte para que esto pueda ser llevado a cabo.

Monitorización y análisis: Se cuenta con herramientas de monitorización y análisis que proporcionan estadísticas e informes de la utilización de las reglas de negocio. Se pueden obtener datos como los siguientes:

- Reglas ejecutadas por transacción.
- Momento en que fueron ejecutadas las reglas.
- Quién ejecutó qué reglas.
- Resultado de ejecución.
- Información de interacción con otros sistemas.

Gestión y administración: El BRMS cuenta con componentes que permiten realizar el despliegue de las reglas de negocio, gestionar la seguridad en el acceso así como el versionado y efectuar un seguimiento del rendimiento del estado del sistema.

Reutilización de reglas de negocio: Algunos BRMS ofrecen la facilidad de poder reutilizar las reglas de negocio, esto puede llevarse a cabo mediante plantillas de reglas que permiten definir estructuras parametrizables que reducen el tiempo de implementación de decisiones comunes en determinados dominios o entornos.

Existen diferentes productos que implementan un BRMS, algunos de código abierto y otros pagos. Entre los de código abierto se encuentra JBoss Rules.

2.6.1 Características de los BRMS

Implementación de un algoritmo de ejecución de reglas eficiente: Los motores de reglas incluyen una implementación de un algoritmo de reconocimiento de patrones, por ejemplo un algoritmo comprobaría cada regla con los hechos residentes en una base de conocimientos activando la regla que corresponda.

Incorporación de gestión de la configuración: Permite manejar el versionado de las reglas de negocio, así como mantener información de los usuarios que realizan las modificaciones sobre las mismas y en que instante se realizaron los impactos.

Distinción de roles de usuarios: Permite asignar distintos tipos de roles a los usuarios del BRMS, el caso típico es el rol de desarrollador, que pueden manipular las reglas de negocio a nivel más técnico y por otro lado, el rol de analista de negocio que puede definir las reglas a alto nivel mediante una interfaz más amigable dispuesta a tales efectos.

Herramientas de despliegue: Es posible que se cuente con herramientas que faciliten el despliegue de la aplicación, se encarga de colocar los artefactos necesarios en el entorno de ejecución que corresponda.

Herramientas para validación: La mayoría de los BRMS disponen de la posibilidad de depurar y validar la ejecución de las reglas de negocio.

Interfaz de usuario en lenguaje natural: Es raro que los BRMS brinden una interfaz de usuario que permite expresar las reglas en lenguaje natural, solamente Haley Expert Rules (recientemente comprado por Oracle) utiliza este enfoque. El resto maneja un lenguaje semi

técnico que se aproxima al lenguaje natural, todos toman al inglés como lenguaje para la expresión de las reglas.

Integración con el proceso de desarrollo: La mayoría de los BRMS poseen una interfaz en un navegador para que los analistas de negocio y otra interfaz para el uso de los desarrolladores que puede estar incorporada en el propio entorno de desarrollo. Por otro lado, las reglas se definen en función de un modelo de negocio propio de la organización, por lo cual debe ser posible importar ese modelo desde el ambiente donde se pretende definir las reglas de negocio.

Mostraremos por otra parte, un conjunto de desventajas que presentan los BRMS:

Deficiencias en estandarización: La interacción con un BRMS consta de dos elementos principales, una es la interfaz de programación que define las operaciones posibles a realizar sobre el sistema y la otra es el lenguaje de representación de las reglas que es capaz de interpretar. En la interfaz se han realizado esfuerzos para conseguir la estandarización donde se ha presentado avances pero también se pueden encontrar algunas carencias. Para el caso de java se ha realizado un JSR (Java Specification Request) que respetan todos los BRMS, pero sin embargo este estándar tiene la carencia de que no permite conformar reglas formadas por la combinación de otras reglas. Con respecto al lenguaje de representación de las reglas cada BRMS implementa una forma distinta de hacerlo. Más allá de que existen los primeros esfuerzos por intentar estandarizar los puntos anteriores, la imposibilidad de poder representar las reglas de una forma estándar hace que exista una fuerte dependencia con el BRMS que se esté usando.

Costo de integración en el proceso de desarrollo: A la hora de identificar que procesos y objetos de negocio intervienen para adaptarlos al BRMS con el que se está trabajando es que se produce una dependencia con el mismo, causando dificultades en caso de querer migrar de BRMS.

Difícil de reutilización de reglas: Las reglas creadas para determinado proyecto no es fácil que puedan ser reutilizadas en otro, esto debido a que los cambios en los modelos de los proyectos suelen distintos.

Escasas facilidades de integración de expertos del negocio: Una de las cosas que más se destacan a nivel comercial de un BMRS, es la posibilidad que estos ofrecen para que los analistas de negocio que tiene escasos conocimientos técnicos puedan definir las reglas. Para esto los BRMS poseen lenguajes específicos que permiten generar expresiones de las reglas de negocio próximas al lenguaje natural. Estos lenguajes no están estandarizados y deben ser construidos para cada modelo con el que se desee trabajar además que la construcción de estos lenguajes suele ser costosa y esto no es publicitado por las empresas encargadas de la comercialización de los BRMS.

2.6.2 Herramientas de software para la gestión de reglas

Existen hoy en día una gran variedad de herramientas de software para la gestión de reglas de negocio, a continuación se presenta una lista de algunas de ellas:

 Jess	Jess es un motor de reglas y un entorno de desarrollo escrito en Java por Ernest Friedman-Hill en el laboratorio de Sandia National en Livermore, CA. Con Jess se puede construir un software en Java con la capacidad de razonar usando el conocimiento proporcionado con las reglas declaradas. Jess es uno de los motores más livianos y rápidos, su lenguaje permite acceder a todas las APIs de Java.
 Drools	Drools es un sistema de gestión de reglas de negocio basado en el algoritmo de Charles Forgy para la JVM. Lo más importante es que Drools ofrece programación declarativa y es lo suficientemente flexible como para reconocer la semántica de su dominio con lenguajes específicos de dominio, también ofrece herramientas para la edición gráfica, herramientas basadas en web y herramientas de productividad para los desarrolladores.
OBSJ	OPSJ es una herramienta para agregar reglas a los programas de Java. El lenguaje de la regla OPSJ es totalmente compatible con el estándar de los programas de Java. Las reglas OPSJ pueden operar de manera transparente en cualquier clase de Java, pueden llamar a cualquier método, y los métodos de Java pueden invocar a las reglas de OPSJ.
 Jamocha	Jamocha es un motor de reglas open source. El objetivo de Jamocha es proporcionar un motor de reglas de alta calidad y un ambiente de sistema experto. Además, proporciona las herramientas de desarrollo necesarias y una metodología para el desarrollo de aplicaciones basadas en reglas.
 JRules	ILOG JRules es un sistema completo para el manejo de reglas de negocio (BRMS). Es la herramienta de IBM.
PegaRULES	Pegasystems' tiene el enfoque único para la gestión de reglas que se basa en la creencia de que, al igual que los datos de negocio, las reglas de negocio debe ser manejadas como un activo corporativo separado. El motor de reglas PegaRULES separa la lógica de negocio de sus aplicaciones críticas y permite a la empresa administrar y ejecutar sus políticas y prácticas empresariales.
Mandarax	El objetivo de mandarax es integrar el razonamiento a Java. El Mandarax es una API para manejar reglas derivadas y un motor de inferencias para ejecutar las reglas.

OpenRules	OpenRules es un BRMS completo. Brinda componentes open source para el desarrollo basado en reglas. Usa herramientas conocidas como MS Excel, Google Spreadsheets, y Eclipse IDE, para apoyar el sistema de decisiones complejas del Business Rules Repository.
Zilonis	El objetivo de este proyecto es proveer un motor de reglas multithreaded y un ambiente de desarrollo para aplicaciones Java.
SweetRules	SweetRules es un conjunto integrado de herramientas para las reglas de la web semántica y ontologías, que gira en torno a RuleML (Rule Markup/Modeling Language) estándar emergente para las reglas de la web semántica, y apoya a SWRL (Semantic Web Rule Language), junto con el estándar de ontologías OWL de la web semántica, que a su vez utiliza XML y, opcionalmente RDF.
Prova	Prova (Prolog+Java) es un sistema basado en reglas para Java y es un agente para desarrollar, integrar y extender el motor Mandarax con una sintaxis apropiada. Combina la sintaxis natural de Java con las reglas de Prolog. Las llamadas Java pueden incluir tanto a constructor como a métodos, además de acceder a variables públicas de las clases.
Hammurapi Rules	Hammurapi Rules es un motor de reglas para desarrolladores Java. Las reglas se desarrollan en Java y compatible con el razonamiento hacia adelante y hacia atrás.
Common Knowledge	Common Knowledge de Object Connections combina un motor de reglas de negocio con un ambiente visual para representar la lógica de negocio en múltiples formatos intuitivos para reflejar cómo se comporta la organización. Maneja formatos como arboles de decisión, tablas de decisión, reglas, workflows, graficas, mapas y scripts.
Haley Business Rules Engine	El motor de reglas Haley Business (conocido antes como HaleyRules, fue adquirido por Oracle) es uno de los motores más rápidos, simplifica el mantenimiento y reduce el ciclo necesario para hacer cambios en las reglas de negocio. Permite definir las reglas en lenguaje natural.
JADEX	Jadex Rules es un motor de reglas, pequeño y liviano, que tiene gran eficiencia para identificar reglas definidas. Es muy parecido a otros motores como JESS y Drools.

2.6.2.1 JBOSS Rules

Para el marco del proyecto se trabajó con el BRMS de Jboss dado que es lo que usa el cliente y es una herramienta open source. En el anexo de tecnologías se detalla un poco más las características de otros BRMS.

Drools (o JBoss Rules) es el componente de JBoss que permite gestionar, almacenar, clasificar, modificar y desplegar “en caliente” reglas de negocio, que podrán ser invocadas vía servicios web desde aplicaciones Java.

Esta herramienta apunta a soportar otros tipos de lógica, como workflows y procesamiento de eventos, en vez de "orientado a las reglas". Drools está "orientado al conocimiento" e introduce el concepto de Plataforma de Integración de Lógica de Negocios (BLiP - Business Logic integration Platform). El futuro de Drools es crear una solución integrada para Reglas, Procesos y Procesamiento de Eventos Complejos (CEP).

El motor de reglas está basado en inferencia, usando una implementación avanzada del algoritmo Rete. El algoritmo Rete es un algoritmo de reconocimiento de patrones eficiente para implementar un sistema que intenta proveer alguna forma de inteligencia, utilizando reglas para establecer el comportamiento. Una implementación simple de un sistema experto basado en reglas comprobaría cada regla con los hechos de la base de conocimiento activando la regla si corresponde, y pasando a evaluar la siguiente.

Proporciona un interfaz Web, que facilita la definición de reglas a personas que no trabajan habitualmente con entornos de desarrollo, principalmente Analistas de Negocio y Expertos en Reglas. Utiliza JCR (JackRabbit) para la gestión del repositorio de reglas y el estándar JAAS para la autorización y autenticación.

Las componentes de Jboss Rules son:

- **Drools Expert** es el motor de reglas tradicional. Es un ambiente de desarrollo basado en reglas. Se enfoca en lo que se quiere hacer y no en cómo se hace.

- **Drools Guvnor** es un repositorio centralizado para las bases de conocimiento de Drools. Es una herramienta web para gestionar las reglas de negocio. Es en Guvnor donde se escribe y edita las reglas. Es el repositorio donde se guardarán las distintas versiones de las reglas, los modelos, funciones y procesos de las bases de conocimiento. El acceso es controlado y es posible restringir funcionalidades según el conocimiento del usuario, lo que permite que usuarios no desarrolladores vean y editen las reglas sin ser expuestos a la totalidad de funcionalidades.

- **Drools Fusion** es el instrumento que maneja CEP (Complex Event Processing) en el motor de reglas. Un evento es una entidad especial que registra un cambio en el estado del dominio de la aplicación, tienen características especiales como restricciones temporales. Drools Fusion entiende lo que es un evento y permite modelar las reglas y procesos dependiendo de la ocurrencia o ausencia de ellos.

- **jBPM** (process/workflow) es un Business Process Management (BPM), es el puente entre los analistas de negocio y los desarrolladores, a diferencia de otros BPM que en general están orientados a gente del negocio, este BPM tiene características para

todos. Puede ser ejecutado en cualquier ambiente Java, embebido en la aplicación o como un servicio, tiene un editor para Eclipse y un editor web para dar soporte visual al proceso de negocio y se pueden hacer conexiones JPA/JTA. Modela procesos complejos y dinámicos, y se pueden combinar con las reglas de negocio y CEP.

Drools trabaja con un lenguaje de reglas "nativo" llamado DRL (Drools Rule Language). Este formato es muy ligero en términos de puntuación, y DSL (Lenguajes Específicos de Dominio) a través de extensiones que permiten que el lenguaje se transforme al dominio del problema. En general la estructura de una regla tiene la sintaxis presentada en la figura 4.

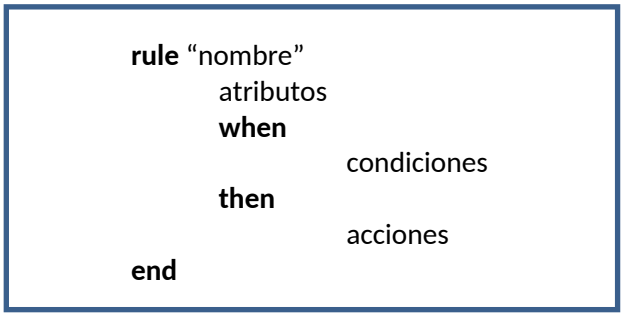

Figura 4: Sintaxis de la regla de negocio.

3 Análisis del negocio de Fiscalización e identificación de reglas

El proyecto tiene como cliente al BPS (Banco de Previsión Social), dentro del cual se encuentra Atyr (Asesoría Tributaria y Recaudación) y particularmente la parte de Construcción de Obras dentro del área Fiscalización. Iremos adentrándonos en la realidad explicando de qué se trata este instituto.

BPS tiene como objetivo principal según se puede extraer de su sitio institucional:

"Brindar servicios para asegurar la cobertura de las contingencias sociales a la comunidad en su conjunto y la recaudación de los recursos, en forma eficaz, eficiente y equitativa, promoviendo políticas e iniciativas en materia de seguridad social, aplicando los principios rectores de la misma en el marco de las competencias que le asigna la Constitución y las Leyes."

Atyr está orientado a la "*recaudación de los recursos*", y su cometido es:

"Asegurar la recaudación de las obligaciones procurando el máximo cumplimiento voluntario y una rápida detección y cobro a los contribuyentes evasores y morosos, mediante una eficaz aplicación de las normas legales y reglamentarias efectuando la registración de la historia laboral y la distribución de los aportes personales de los trabajadores."

Dentro de este marco, Atyr tiene la responsabilidad del registro, la gestión y el cierre de las obras.

En este proyecto vamos a estar enfocados en lo que refiere al cierre de obras.

Hace un tiempo atrás para poder cerrar una obra era necesario someter a la misma para sea inspeccionada, en caso de que la obra estuviera en condiciones de ser cerrada se hacía efectivo el cierre. Con la inspección se entiende al control por parte del BPS de una serie de puntos que debe cumplir la obra en cuanto a su regularidad, un ejemplo sería constatar que la cantidad de jornales declarados por la empresa constructora sea coherente con la obra en cuestión.

Surgió entonces en BPS la necesidad de automatizar este proceso y para eso creó un aplicativo llamado "Constructor" que se encarga de realizar dicha tarea y del cual estaremos hablando a continuación.

Este aplicativo que mencionábamos, hace un tiempo atrás tuvo un acercamiento hacía una solución que contemplara la gestión de las reglas de negocio pero que luego de desestimó. De todas formas BPS aún quiere modificar el diseño de la aplicación para incorporar dicha gestión.

3.1 Estudio del área de negocio.

Para realizar un relevamiento de las reglas, primero tuvimos realizar un estudio del área de negocio del cual presentaremos un resumen, en el Anexo J, Estudio del área de negocio, se podrá profundizar más en este estudio.

Las obras en fiscalización presentan determinadas características, a la hora de realizar un cierre dichas características son tenidas en cuenta ya que dada una determinada combinación de las mismas se puede afirmar que una obra está en condiciones para ser cerrada. Existen tres formas de realizar un cierre de una obra desde el aplicativo Constructor, estos son:

- Cierre de obras automático masivo.
- Cierre de obras automático sin deuda.
- Cierre de obras automático con deuda.

3.1.1 Cierre de obras automático masivo.

El Constructor para realizar este procesamiento masivo tiene un proceso que está dividido en dos grandes partes, el llamado "*Carga del universo*" y el "*Cierre de Obras*" propiamente dicho.

La carga del universo es un proceso previo al cierre de las obras. La ejecución recaba datos de diferentes sistemas a fin de obtener información que posibilitará evaluar individualmente cada obra para saber si esta puede ser cerrada.

Los servicios externos a Fiscalización que se debieron consultar para reunir la información de las obras fueron los siguientes:

- Registro de Obras
- Registro de Empresas
- Recaudación Nominada
- Fiscalización
- Cuenta Empresa

La carga recibe determinados parámetros, a saber:

- Fecha desde
- Fecha hasta
- Número de agencia
- Tipo de obra (pública o privada)
- Con F9/Sin F9

Se cargarán en este caso las obras que cumplan las siguientes condiciones:

- Las empresas correspondientes a las obras que no hayan presentado el formulario F9.
- La fecha de finalización estimada presentada en el formulario F1 se encuentra entre las fechas "fecha desde" y "fecha hasta" ingresadas como parámetro en la carga.
- La fecha de finalización estimada presentada en el formulario F1 más 6 meses, es posterior al día de la ejecución del cierre automático. Esta cantidad de meses puede variar entre una ejecución y otra, o sea que puede llegar a redefinirse.

Como se mencionaba anteriormente, se obtiene información de diferentes negocios. La tabla 2 “Tabla de datos” muestra cuáles son los datos que se extraen de cada uno de ellos a la hora de realizar la carga del universo.

Registro de obras	Recaudación nominada	Registro de empresas	Cuenta empresa
Empresa titular	Contribuyente Titular	Número interno de contribuyente	Saldo
Obra	Empresa Titular		
Padrón	Aportación Titular		
Unidad	Mes de cargo desde		
Fecha desde	Mes de cargo hasta		
Fecha hasta	Cantidad de nóminas presentadas		
Forma de realización	Cantidad de días trabajados		
Fecha de inicio	Mes de presentación de la última nómina		
Fecha de fin estimada			
Fecha de presentación del F9			
Departamento			
Características			

Tabla 2: Tabla de datos

Hasta aquí se presentó el proceso de carga del universo, pasamos a describir el “*Cierre de Obras*” propiamente dicho.

Una vez realizada la carga, se procede a realizar un procesamiento a cada una de las obras que fueron cargadas para determinar si corresponde o no ser cerrarse.

Este procesamiento implica evaluar determinadas condiciones pre-definidas por el usuario, estas suelen variar a corto plazo, y es necesario que el usuario del sistema pueda definirlas cuando lo crea conveniente sin la necesidad de modificar el sistema y sin la necesidad de la intervención del equipo técnico. Hoy en día las condiciones son semi-variables, a esto nos referimos con que pueden deducirse de los datos de una tabla almacenada en base, entonces pese a no tener la flexibilidad que se requiere se pueden variar los datos de la tabla y con eso se logra cambiar en cierta forma la condición. Algo no menor es que para modificar la tabla es necesaria la intervención de técnicos y que éste método no hace las condiciones lo suficientemente flexibles.

Las condiciones que se deben evaluar para determinar si una obra puede ser cerrada son las siguientes:

- La fecha desde registrada de la obra debe ser mayor o igual setiembre que 2006.
- La cantidad de días trabajados de la obra debe ser menor o igual a 150.

- Si se está evaluando una obra de una empresa privada entonces esta tiene que tener presentado ante BPS más del 70% de sus nóminas. En caso de que la obra sea pública no se evalúa esta condición.
- El saldo en cuenta de empresa debe ser menor o igual a 3 unidades reajustables, tomando el valor de la unidad reajustable del último mes de la ejecución del cierre.
- Si se está evaluando una obra de una empresa privada entonces la duración de la obra debe ser menor o igual 12 meses. En caso de que la obra sea pública no se evalúa esta condición.
- La agencia de la obra debe ser igual a 1, que es el correspondiente a Montevideo (en el cierre de este proyecto, se está extendiendo para más agencias).
- Si existe una actuación asociada a la empresa correspondiente a la obra, esta no es cerrada.
- Si la obra tiene asociado algún trámite en las oficinas 226, 229 y 2004 la obra no puede ser cerrada.
- El código del departamento debe ser igual a 1, que es el código correspondiente a Montevideo (en el cierre de este proyecto, se está extendiendo para más departamentos).

Todas las obras que pasen los filtros enunciados anteriormente serán cerradas.

3.1.2 Cierre de obras automático sin deuda.

Luego de tener disponible un cierre masivo de obras, se comenzó a trabajar en una funcionalidad que permitiera a los usuarios realizar el cierre de obras automático pero esta vez a demanda y no de forma masiva, esto implica que el usuario debe indicar al aplicativo cuál es la obra a ser cerrada con su correspondiente empresa.

Para evaluar si la obra puede ser cerrada o no, se buscan los mismos datos de los mismos negocios que se enumeraron para el cierre masivo en la carga del universo.

Estos son los parámetros de entrada que se deben recibir para realizar un cierre a demanda sin deuda:

- Oficina
- N° SESP
- Empresa
- Obra

Se deben realizar las siguientes validaciones:

- Formato del número de SESP (número de trámite con el que queda registrado el cierre).
- Existencia del número de SESP.
- Disponibilidad de número de SESP.
- Empresa válida.
- Obra válida.
- Relación entre la empresa y la obra válida.

Con estos datos se sugiere una fecha desde y una fecha hasta que se utilizará como rango en el cual se va a efectuar el cierre.

Para que la obra pueda ser cerrada debe pasar por una serie de filtros, estos son:

- La fecha desde debe ser mayor a setiembre de 2006.
- Se verifica que no existan actuaciones en el sistema *Guia-Gdt*.
- Se verifica que existan movimientos en *Cuenta Empresa* para el último mes correspondiente a la fecha hasta.
- Se debe verificar que el saldo en *Cuenta Empresa* sea menor a 3 unidades reajustables.

En caso de que pase por cada uno de estos filtros la obra será cerrada.

3.1.3 Cierre de obras automático con deuda.

Esta funcionalidad fue desarrollada una vez puesto en producción el cierre automático a demanda sin deuda.

A grandes rasgos este tipo de cierre es similar al anterior en el sentido que permite realizar cierres a demanda. Una diferencia con el *sin deuda* es que es posible que la obra tenga un saldo en *Cuenta Empresa* mayor a 3 unidades reajustables.

Además, se puede realizar el cierre para una empresa contratista en particular. Las obras siempre tienen una empresa asociada cumpliendo el rol de *titular*, pero además la obra puede tener asociadas empresas contratistas(o contratadas).

Los siguientes datos son los que se deben ingresar para realizar el cierre a demanda con deuda:

- Oficina
- Número de Sesp
- Empresa Titular
- Número de Obra
- Artículo 7° (Se puede definir con o sin artículo 7°)
- Empresa Contratista. (Solo se ingresa este datos cuando se optó por artículo 7°)
- Tipo de cierre (total o parcial).
- Recargos Condicionados. Se debe definir si incluye recargos condicionados o no, una vez obtenida la deuda a pagar la misma puede estar condicionada a determina forma de pago con el fin de obtener o no beneficios en los recargos. En caso de elegir con recargos condicionados se debe ingresar la fecha de F9.
- Fecha F9, fecha de presentación del formulario F9, es un campo calculado.
- Fecha de recargo personal, representa la fecha a la que se van a calcular las deudas correspondientes.

Los filtros que deben pasar las obras para que pueda ser cerrada son los mismos que para el cierre a demanda con deuda excepto el que hace referencia al saldo en la cuenta, donde ahora se debe verificar que este sea mayor a 3 unidades reajustables.

Hasta aquí hemos presentado un breve estudio del área de negocio. Ahora identificaremos en base a este estudio, cuales son las reglas de negocio.

3.2 Relevamiento de reglas de negocio (basado en las plantillas RuleSpeak).

En esta sección se expondrán las reglas relevadas del estudio del área de negocio. La expresión de las mismas será utilizando la plantilla de RuleSpeak, de la cual se hablara en la sección 2.5. Dividiremos en cuatro partes la agrupación de las mismas, estas son: carga del universo, cierre masivo, cierre demanda sin deuda, cierre demanda con deuda.

3.2.1 Carga del universo

Comenzamos a enumerar las reglas relevadas para el cierre masivo, enumerando las reglas de la carga del universo en primer lugar:

- El usuario tiene que definir una fecha desde para el cierre masivo.
- El usuario tiene que definir una fecha hasta para el cierre masivo.
- El usuario tiene que definir una agencia para el cierre masivo.
- El usuario tiene que definir un tipo de obra (pública o privada) para el cierre masivo.
- Las obras candidatas a ser cerradas tienen que tener la fecha de presentación de F9 comprendida entre la fecha desde y la fecha hasta definidas por el usuario si éste ha optado por realizar cierres "ConF9"[1].
- Las obras candidatas a ser cerradas tienen que tener la fecha de fin estimado del formulario F1 comprendida entre la fecha desde y la fecha hasta definidas por el usuario si éste ha optado por realizar cierres "Sin F9"[2].
- La diferencia entre la fecha de fin estimado del formulario F1 y la fecha de ejecución del cierre tiene que ser menor a seis meses si el usuario ha optado por realizar cierres "Sin F9".
- Las obras candidatas a ser cerradas tienen que ser de la agencia definida previamente por el usuario.
- El tipo de obra de las obras candidatas a cerrar tienen que ser del tipo de obra definido previamente por el usuario.
- Las obras candidatas a ser cerradas tienen que presentar movimientos en *Recaudación* para el periodo definido por el usuario.
- Las obras candidatas a ser cerradas tienen prohibido tener actuaciones
- integrales no anuladas en el sistema *Guia-Gdt*.
- El saldo en la cuenta para cada obra candidata a ser cerrada ha de ser calculado como la suma de todos los meses de cargo de la empresa titular, con su respectiva empresa contratista y número de obra, para todos los códigos y subcódigos[3].

Continuamos con el cierre masivo, ahora con la parte referente al cierre, estas son las reglas identificadas:

- La fecha desde registrada de la obra tiene que ser mayor o igual que setiembre de 2006.
- La cantidad de días trabajados de la obra tiene que ser menor o igual a 150.

[1] Las obras candidatas a ser cerradas son aquellas que son cargadas por el sistema para que luego se le puedan aplicar los diferentes filtros correspondientes a las reglas identificadas para el cierre masivo.

[2] Las obras candidatas a ser cerradas son aquellas que son cargadas por el sistema para que luego se le puedan aplicar los diferentes filtros correspondientes a las reglas identificadas para el cierre masivo.

[3] Los códigos y subcódigos se utilizan para identificar dentro de un plan de cuentas los distintos rubros que lo componen.

- La obra tiene que tener presentado ante BPS más del 70% de sus nóminas si es privada.
- Tener presentado ante BPS más del 70% de las nóminas no tiene por qué cumplirse si la obra es pública.
- El saldo en cuenta de la empresa tiene que ser menor o igual a 3 unidades reajustables.
- El valor de la unidad reajustable a considerar tiene que corresponder al del último mes correspondiente a la ejecución del cierre.
- La duración de la obra tiene que ser menor o igual 12 meses si la obra es privada.
- La validación de la duración menor o igual a 12 meses no tiene por qué realizarse si la obra pública.
- La agencia de la obra tiene que ser igual a 1[4].
- La empresa correspondiente a la obra tiene prohibido tener una actuación asociada.
- La obra tiene prohibido tener algún trámite en las oficinas 226, 229 y 2004.
- El código del departamento de la obra tiene que ser igual a 1.

Hemos presentado las reglas de negocio del cierre masivo, ahora es el turno de los cierres a demanda sin deuda, estas fueron las reglas identificadas:

- La fecha desde sugerida ha de ser calculada como la menor fecha desde de todos los periodos activos de la obra.
- La fecha hasta sugerida ha de ser calculada como la mayor fecha hasta de todos los periodos activos de la obra si el último periodo activo está cerrado.
- La fecha hasta sugerida tiene que ser definida por el usuario si el último periodo activo está abierto.
- El periodo ingresado por el usuario tiene que estar comprendido en el periodo sugerido.
- Una vez realizado el cierre este tiene que indicar si el mismo es definitivo o parcial.
- La fecha desde registrada de la obra tiene que ser mayor o igual setiembre de 2006.
- El saldo en la cuenta de la empresa tiene que ser menor o igual a 3 unidades reajustables.
- El valor de la unidad reajustable a considerar tiene que ser del último mes correspondiente a la ejecución del cierre.
- La empresa correspondiente a la obra tiene prohibido tener una actuación asociada.
- La empresa correspondiente a la obra tiene que tener movimientos en *Cuenta Empresa* en el último mes de cargo correspondiente a la fecha hasta definida por el usuario.
- La agencia de la obra tiene que ser igual a 1.
- El código del departamento de la obra tiene que ser igual a 1.

Estas fueron las reglas relevadas del cierre a demanda sin deuda, ahora y por -último presentamos las reglas correspondientes a los cierres a demanda con deuda:

- La fecha desde sugerida ha de ser calculada como la menor fecha desde de todos los periodos activos de la obra si no se ha optado por la opción del artículo 7.

[4] La agencia 1 es la agencia correspondiente a Montevideo.

- La fecha hasta sugerida ha de ser calculada como la mayor fecha hasta de todos los periodos activos de la obra si el último periodo activo está cerrado y no se ha optado por la opción del artículo 7.
- La fecha hasta sugerida tiene que ser nula si el último periodo activo está abierto y no se ha optado por la opción del artículo 7.
- La fecha desde sugerida ha de ser calculada como la máxima entre la menor de las fechas de los periodos activos de la obra y la fecha que está registrada en la relación entre la empresa titular, la empresa contratista y la obra si se ha optado por la opción del artículo 7.
- La fecha hasta sugerida ha de ser calculada como la fecha mínima entre la mayor de las fechas de los periodos activos de la obra y la fecha que está registrada en la relación empresa titular, empresa contratista y la obra si se ha optado por la opción del artículo 7.
- La fecha hasta sugerida tiene que ser nula si alguno de los valores al que se le va a calcular el mínimo es un valor nulo y se ha optado por la opción del artículo 7.
- El periodo de fechas ingresado por el usuario tiene que estar comprendido en el periodo sugerido.
- La fecha desde registrada de la obra tiene que ser mayor o igual setiembre de 2006.
- El saldo en la cuenta de la empresa tiene que ser mayor o igual a 3 unidades reajustables.
- El valor de la unidad reajustable a considerar tiene que ser del último mes correspondiente a la ejecución del cierre.
- La empresa correspondiente a la obra tiene prohibido tener una actuación asociada.
- La empresa correspondiente a la obra tiene que tener movimientos en *Cuenta Empresa* en el último mes de cargo correspondiente a la fecha hasta definida por el usuario.
- Una vez realizado el cierre este tiene que indicar acerca de si el mismo es definitivo o parcial.
- Una vez realizado el cierre este tiene que indicar si el mismo contiene recargos condicionados o no.
- Una obra tiene prohibido no tener presentado el formulario F9 si se le quiere realizar un cierre con recargos condicionados.
- Una vez realizado el cierre este tiene que contener información acerca de la fecha de presentación del formulario F9.
- La fecha de presentación de F9 registrada en el cierre tiene que ser la fecha de presentación de F9 correspondiente a la obra si se ha optado por la opción de incluir recargos condicionados.
- La fecha de presentación de F9 registrada en el cierre tiene que ser nula si no se ha optado por la opción de incluir recargos condicionados.
- Una vez realizado el cierre este tiene que contener información acerca de la fecha de recargo.
- La fecha de recargo registrada en el cierre ha de ser calculada como la fecha correspondiente al último día del mes anterior al día de la ejecución si se ha optado por realizar un cierre no condicionado.
- La fecha de recargo registrada en el cierre ha de ser calculada como la menor fecha entre la correspondiente al último día del mes anterior al día de la ejecución y la correspondiente al último día del tercer mes posterior a la fecha de presentación del F9 si se ha optado por realizar un cierre no condicionado.
- La agencia de la obra tiene que ser igual a 1.
- El código del departamento de la obra tiene que ser igual a 1.

4 Definiciones para este proyecto

Con lo investigado y exhibido en las secciones anteriores y en los anexos vemos que hay muchas definiciones y puntos de vista para cada concepto relacionado a las reglas, las clasificaciones y el ciclo de vida. Por lo que en esta sección definiremos qué vamos a considerar como definición para cada uno, siendo la organización de la información y su estructuración uno de los aportes principales de este proyecto.

4.1 Definición de regla de negocio.

Para el contexto de nuestro proyecto una regla de negocio es una combinación de todo lo expuesto en la subsección 2.1.1. "Una regla de negocio será considerada como una definición o restricción que afirma, controla e influencia aspectos del negocio. Debe expresarse en términos directamente relacionados con el negocio utilizando un lenguaje simple y no ambiguo accesible para todas las partes interesadas."
"Ciertas reglas de negocio serán reflejadas en el sistema de software pero también habrá otras que no, las reglas de negocio son del negocio", como lo indica la palabra por lo cual estarán presentes en cualquier sistema independientemente de su implementación, habrá reglas definidas según el software que no son inherentes al negocio.

4.2 Definición de una clasificación de reglas.

Como se puede ver en el anexo de Clasificación de reglas la mayoría de las clasificaciones se basan en la del Business Rule Group. Consideramos que es una clasificación bastante completa por lo que la hemos elegido como base para definir una clasificación de reglas propia, adicionando la categoría de reglas de presentación que aparece en varios de los enfoques orientados al desarrollo. A continuación se puede observar en la figura 5 el esquema de clasificación del BRG.

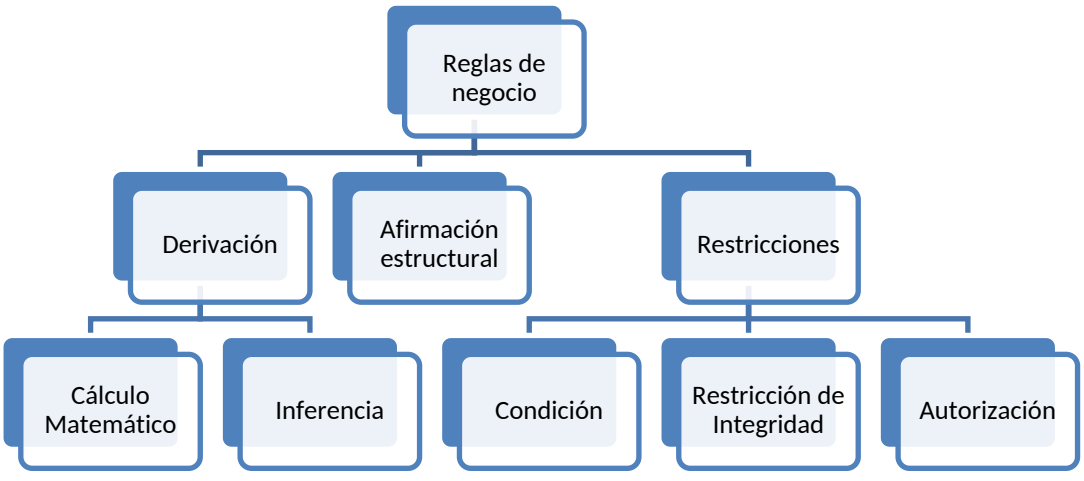

Figura 5: Clasificación del BRG.

A la clasificación del BRG le agregamos las reglas de presentación ya que pueden ser útiles para definir aspectos visuales del sistema y permiten que contenido dinámico pueda ser presentado a los usuarios.

Se pudo ver que la clasificación de Oracle (ver el anexo de Clasificación de Reglas) se ajusta a la realidad del día a día y además está orientada al desarrollo por lo que se decidió agregarla para complementar la clasificación orientada a los analistas de negocio. Como se puede ver en el esquema que corresponde a la clasificación de Oracle en la figura 6, es una clasificación que profundiza más cada categoría.

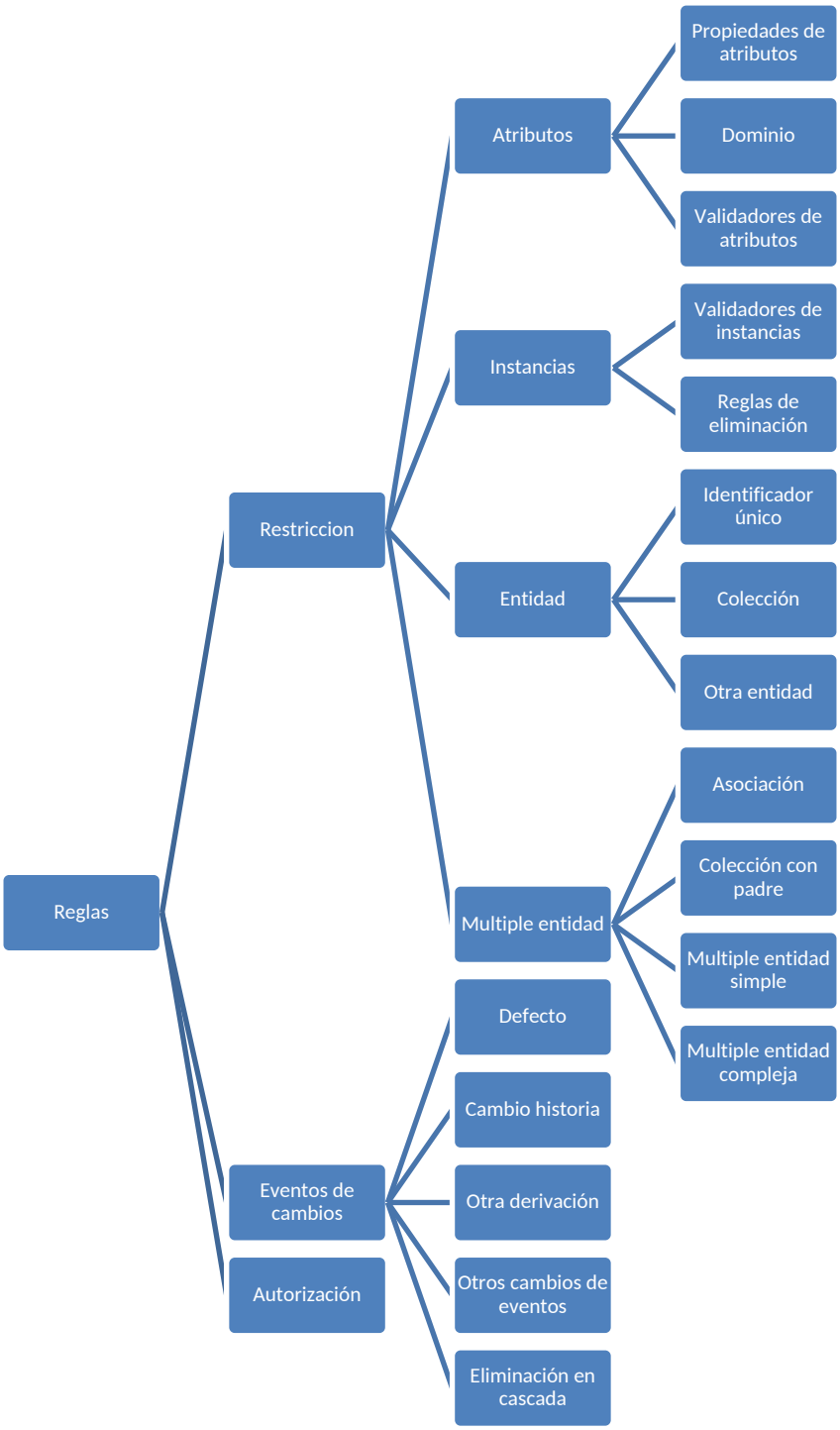

Figura 6: Clasificación de Oracle

Mostraremos una tabla donde se muestra como se incorporaría las categorías de Oracle en nuestra clasificación.

Categoría Business Rule Group	**Categorías Oracle**
Afirmación estructural	Reglas de Atributos
Restricción de integridad	Reglas de Instancias
	Reglas de Entidad
	Reglas de Múltiple Entidad
Autorización	Reglas de Autorización
Afirmación de acción	Reglas de Cambios de Eventos

Tabla 3: Comparación entre la clasificación del BRG y Oracle

Por lo tanto, nuestra clasificación sería:

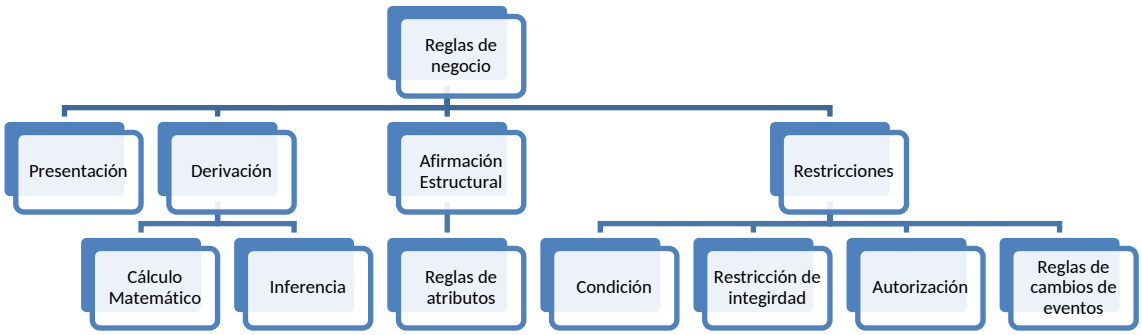

Figura 7: Esquema de clasificación propio

Cálculo: Son sentencias que refieren a un cálculo matemático.
Ejemplo: El total es el costo de mano de obra multiplicado por las horas de trabajo.

Inferencia: Es una derivación que produce un hecho usando lógica inductiva o deducciones.
Ejemplo: Si el cliente tiene una mala calificación crediticia, el método de facturación se establece como prepago.

Estructural: Definición a nivel de estructura de algo importante para el negocio, puede ser un concepto o una relación con otro interés para el mismo.
Ejemplos:

- Devolución.
- Color es un atributo de un auto.
- Los autos son alquilados por clientes.

Condicional: Reglas que indican que si algo es verdadero se aplica una regla de negocio.
Ejemplo: Si el conductor no presenta la licencia de conducir vigente entonces le será aplicada una multa.

Restricción de integridad: Indican que una declaración siempre debe ser verdadera. A diferencia de las reglas de condición no disparan ninguna acción.
Ejemplo: Los autos deben estar empadronados.

Autorización: Define permisos sobre cierta regla o cierta acción.
Ejemplos: Sólo el manager de la empresa puede autorizar una devolución del producto.

Eventos: Definen una acción automática provocada por un cambio en el estado del sistema.
Ejemplos: Al eliminarse una factura se deben eliminar automáticamente las líneas de productos asociadas a la misma.

Guía: Una guía es una declaración completa que expresa un aviso sobre una circunstancia que debe ser verdadera o no.
Ejemplo: Un cliente no debería tener más de 10 órdenes abiertas al mismo tiempo.

Instrucción: Son reglas que instruyen a los usuarios sobre qué hacer en el sistema en ciertas situaciones, no están implementadas en el sistema, sino en manuales. También son usadas para guiar a los usuarios en las partes más complejas del sistema.
Ejemplo: use EUA en vez de Estados Unidos de América ya que esa abreviación es la reconocida en el sistema.

Presentación: Son reglas que definen aspectos de las interfaces de usuario. Se definen aspectos de color como el color de fondo, el color de letra, el tipo de letra, la fuente, que cosas deben decir los labels, la alineación de los objetos en la aplicación, etc.
Ejemplo: La información sobre el producto y el precio del mismo deben ser mostrados juntos en la misma pantalla.

En la tabla 4 se muestra cada categoría de regla y las distintas clasificaciones presentadas en el documento, se marca aquellas categorías que la clasificación incluye.

	BRG	R. Ross	Von Halle	Usoft	Versata	IBM	Oracle	Primatek	Nuestra
Cálculo	X	X	X		X	X	X	X	X
Inferencia	X	X	X	X	X	X	X	X	X
Estructural	X		X				X	X	X
Condicional	X	X	X		X	X	X	X	X
Restricción integridad	X	X	X	X	X	X	X	X	X
Autorización	X			X			X	X	X
Eventos	X	X	X	X	X	X	X	X	X
Presentación		X		X	X			X	X
Guía			X						
Instrucción				X					

Tabla 4: Asociación de las clasificaciones estudiadas con la definida en este proyecto.

4.3 Definición del ciclo de vida de las reglas.

En esta subsección se definirá un ciclo de vida de las reglas de negocio basado en los ciclos de vida que hemos investigado y que se pueden estudiar en los anexos. También definiremos los parámetros para la gestión de las reglas de tal manera que conserven la calidad y tener la trazabilidad de las mismas.

4.3.1 Definición del ciclo de vida

A continuación presentamos en la figura 8, un diagrama de lo que nosotros vamos a considerar como ciclo de vida de las reglas.

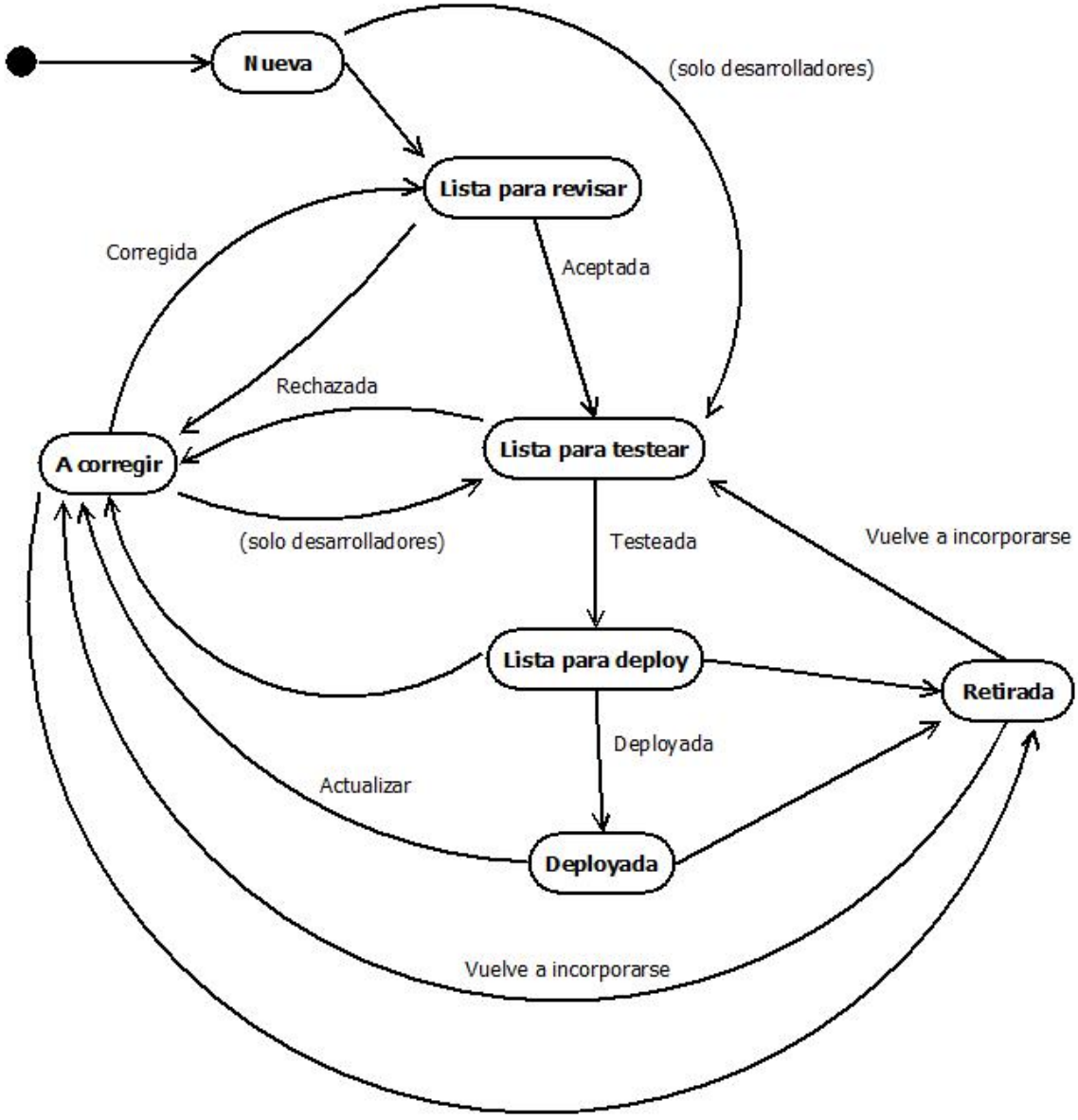

Figura 8: Ciclo de vida de la regla definido en el marco de este proyecto

Se detalla a continuación los actores para cada estado, la descripción de las tareas a realizar, el soporte que idealmente debería brindar un BRMS, y el soporte que brinda Drools (el BRMS de Jboss).

Estado: Nueva	
Actores	Analista de negocio, Desarrollador, Experto de negocio.
Descripción	Estado para la definición de reglas, las formas de expresarlas pueden ser: - Lenguaje natural siguiendo las plantillas de RuleSpeak, estas reglas pueden ser expresadas tantos por analistas como por expertos de negocio. - Expresadas utilizando un lenguaje formal donde es posible expresar en forma gráfica reglas de la forma condición-acción, donde condición puede ser una conjunción o disyunción de operandos, y son estos a su vez distintos tipos de condiciones. Las acciones pueden ser desde un seteo en un atributo de una entidad, hasta la invocación de un procedimiento o el despliegue de algún tipo de mensaje. Al igual que en el caso anterior, estas reglas pueden ser expresadas tantos por analistas como por expertos de negocio. - Expresadas en un lenguaje formal técnico específico para desarrolladores, donde se podrán definir reglas con un nivel de complejidad que imposibilitaría la expresión de la misma en alguna de las formas definidas anteriormente. Este tipo regla sería expresada por un desarrollador.
Soporte sistema	El sistema debe brindar soporte para expresar reglas de negocio en cualquiera de las tres formas puntualizadas anteriormente. Si la regla es válida el sistema deberá realizar la traducción del lenguaje del analista de negocio (lenguaje natural con RuleSpeak o lenguaje formal) al lenguaje del motor de reglas. A su vez la herramienta debe permitir para cada una de estas formas, expresar reglas de todas las categorías de la clasificación de reglas. Se ofrece la posibilidad de definir un flujo de reglas de negocio que serán ejecutadas en cualquier orden o bien de acuerdo a una prioridad previamente definida por el usuario. El sistema tiene que proveer un mecanismo de soporte para la gestión de la configuración. Los actores Analista de Negocio y Experto de Negocio tienen la posibilidad de marcar la regla como “Lista para revisar” mientras que el actor Desarrollador puede marcarla como “Lista para testear”.
Soporte de Drools para este estado	Lenguaje natural: Tiene la forma de escritura DSL (Domain Specific Languaje) que permite la expresión en lenguaje natural de sentencias con condiciones y acciones. Pero se deben mapear estas construcciones sintácticas que formarán el DSL con las instrucciones en lenguaje DRL (Drools Rule Language) para que lo pueda interpretar Drools. Esa asociación la deben realizar los especialistas técnicos. Por otro lado se podrían escribir reglas que vayan con lo que dice la plantilla de RuleSpeak en el DSL, pero no se pueden

	realizar controles para verificar que el formato de plantilla este correcto. También permite expresarlas utilizando un lenguaje formal y en un lenguaje formal técnico específico para desarrolladores.

Estado: Lista para revisión	
Actores	Experto de negocio, Analista de negocio.
Descripción	Este estado indica que la regla ha sido creada completamente y está lista para que un revisor (Experto de Negocio o un Analista de Negocio) realice la validación de la regla que corresponda, previamente se debió marcar la regla como para lista para revisión.
Soporte sistema	El sistema deberá proveer una forma de poder visualizar la regla de una manera amigable a la hora de realizar la validación correspondiente. Sería bueno contar con la posibilidad de agregar comentarios y anotaciones a la misma. El Experto de Negocio debe tener la forma de marcar esta regla para que pase al estado “Lista para testear” en caso de que haya pasado las validaciones correspondientes o de marcarla para que pase al estado “A Corregir” en caso contrario.
Soporte de Drools para este estado	No provee una forma de visualizar la regla orientada a un revisor. Si es posible visualizarla de la misma manera que la ve el creador de la misma. También es posible ver la regla en un documento que Guvnor crea automáticamente como documentación de la misma donde en este lugar no la puede editar. Lo que más se aproxima a la posibilidad de agregar comentarios y anotaciones es que Guvnor provee una forma de agregar comentarios genéricos de la regla. Sucesivos comentarios van quedando disponibles en una especie de log. Guvnor brinda la posibilidad de realizar gestión de estados en su módulo de administración, esto significa que es posible crear, modificar y eliminar estados. Estos luego podrán ser usados por los usuarios que crean o editan las reglas para asignárselos a la misma. De esta forma es posible realizar las transiciones que corresponda. Algo que no es posible es definir precedencia entre los estados, lo que implicaría que el usuario lleve el control del flujo entre los mismos, ya que es posible llegar a cualquier estado partiendo de cualquier otro. Se permite visualizar conjuntos de reglas de acuerdo al estado en el que se encuentran.

Estado: Lista para testear	
Actores	Experto de negocio, Analista de negocio, Desarrollador.
Descripción	En este estado la regla ha sido revisada y validada por un experto del negocio, esta etapa indica que la regla esta pronta para ser testeada. Este estado se alcanza si la revisión fue aprobada por un revisor, pero también si la regla fue creada por un desarrollador o si se realizaron correcciones y el desarrollador la marcó para ser testeada nuevamente.
Soporte sistema	El desarrollador genera casos de testeo que son validados por el experto y analistas de negocio. La herramienta debe proveer un ambiente adecuado para ejecutar los casos de prueba, una vez que el desarrollador da por aprobado el testeo éste puede marcar la regla como "Lista para Deploy", en caso contrario marca esta regla como rechazada para que pase al estado "A Corregir" del ciclo de vida. Una extensión de esta herramienta podría incluir componentes para el debbuging y monitoreo de ejecución de las pruebas.
Soporte de Drools para este estado	Guvnor brinda la posibilidad de armar escenarios de testeo para hacer pruebas de caja negra sobre las reglas. El cambio de estados es lo mismo que en lista para revisar.

Estado: Lista para Deploy	
Actores	Experto de negocio, Especialista técnico.
Descripción	Este estado indica que la regla ya fue testeada y esta pronta para formar parte de las reglas en producción. La regla va a estar en el siguiente estado cuando el experto de negocio apruebe la decisión y el especialista técnico realice ésta acción.
Soporte sistema	El sistema deberá proveer una interfaz amigable para que el experto del negocio informe de su decisión, y además una forma amigable de deployar la regla en el motor. Es posible que debido a cambios en las políticas de negocio la regla deba ser modificada o retirada ya que puede quedar obsoleta en la nueva política, por eso la interfaz debe brindar las opciones de enviar al estado de "A Corregir" o al estado "Retirada".
Soporte de Drools para este estado	Mediante el cambio de estados explicado en el estado "Lista para revisión" es posible marcar una regla como "Lista para deploy". Para deployar la regla en el motor se debe realizar un "build" de las reglas que están estado "Lista para deploy" y se debe crear un "snapshot" correspondiente a producción.

Estado: Deployada	
Actores	Especialista técnico, Desarrollador.
Descripción	Cuando una regla llega a este estado es porque la misma pasa a estar en el ambiente de producción. Estas reglas pueden desde este estado pasar a estar "Retirada" o al estado "A Corregir" debido a cambios en las políticas de negocio.
Soporte sistema	El sistema debe proveer soporte para la obtención de reportes y monitoreo en tiempo real de las reglas que se están ejecutando. El Especialista técnico debe contar con una forma sencilla de poder marcar una regla como "Retirada" o para que mueva a "A corrección".
Soporte de Drools para este estado	Las reglas en Guvnor quedan con estado deployadas. Luego cuando una regla cambie de estado se debe realizar un nuevo snapshot. Pudiendo tener un snapshot de producción, otro de testeo y tantos como se considere necesario. En el build de cada snapshot se puede indicar las reglas con que estados serán incluidas.

Estado: A Corregir	
Actores	Experto de negocio, Analista de Negocio, Desarrollador.
Descripción	En este estado residen las reglas que han sido marcadas para corregir. Una regla puede ser marcada a corregir por distintos motivos: - Las reglas que no son aprobadas por el experto en el proceso de revisión, y son marcadas para ser posteriormente corregidas. - En caso de encontrar errores en la regla en la fase de testeo, esta puede ser marcada para que se le realizan las correcciones que corresponda. - Una vez que la regla esta "Lista para deploy" o "Deployada" pueden haber cambios en políticas de negocio que impliquen que la misma sea corregida.
Soporte sistema	La interfaz de escritura para la definición de las reglas debe proveer un mecanismo de edición de las reglas de negocio para todas las formas de expresión enumeradas en el estado "Nueva". Una vez realizadas las correcciones en la regla que correspondan, se debe brindar un mecanismo para marcar a la misma para ser revisada. En caso de que la regla haya sido corregida por un desarrollador, éste debe tener la posibilidad que marcarla como lista para testear, de lo contrario será enviada para que el experto la vuelva a revisar en "Lista para revisar". A su vez se pueden dar cambios de política del negocio que impliquen que la regla ya no sea vigente por lo que la herramienta debe brindar la posibilidad de cambiar el estado a "Retirada".
Soporte de Drools para este estado	Las reglas se pueden editar para cualquiera de las tres formas de expresión. Guvnor permite realizar los cambios de estados que se necesitan para proceder como se recomienda en este ciclo de vida.

Estado: Retirada	
Actores	Especialista técnico, Experto de negocio, Desarrollador.
Descripción	Las reglas se pueden transformar en obsoletas debido a cambios en las políticas de negocio. Una regla puede retirarse una vez que esta deployada o incluso cuando recién está lista para deployar. Existe la posibilidad de que una regla que se encuentra retirada, también debido a políticas de negocio vuelva a estar vigente, donde pasaría en este caso nuevamente al estado "Lista para testear" o al estado "A corregir".
Soporte sistema	Se debe proveer una forma de marcar una regla que está retirada para que pase al estado "Lista para testear" ya que una vez más debe ser probada con el resto de las reglas, debido a que pueden haber nuevas reglas y las existentes pudieron haber sido modificadas mientras esa regla estaba en desuso. Además es necesario contar con la posibilidad de enviar una regla retirada a que sea corregida ya que esta puede requerir modificaciones antes de ser testeada e incorporada al sistema.
Soporte de Drools para este estado	Como ya se ha mencionado se puede pasar la regla a diferentes estados en Guvnor.

4.3.2 Gestión de la configuración

Es necesario realizar una Gestión de la Configuración para mantener la calidad de las reglas de negocio durante cualquier de las etapas de la misma. Esto se debe llevar a cabo controlando los cambios realizados sobre las reglas y donde siempre tiene que haber disponibilidad de una versión estable para cada persona que corresponda. De esta forma se facilita el manejo de las reglas de negocio al proporcionar una imagen detallada de las mismas en cada una de sus etapas.

Para cada una de las reglas de negocio, será importante mantener la siguiente información:

- Nombre
- Versión
- Estado
- Comentario (asociado al estado)
- Localización
- Fecha de Creación
- Usuario de Creación
- Fecha de Última Actualización
- Usuario de Última Actualización

Además se debe disponer de un historial con las diferentes versiones de las reglas que se han ido registrando y en donde sea posible saber los datos del usuario que registro los cambios y cuando fue que se realizaron.

La herramienta por consiguiente debe dar soporte para brindar los servicios descritos anteriormente, a su vez ésta debe proveer una interfaz amigable para que cada uno de los

actores involucrados pueda llevar a cabo la gestión correspondiente. En ese sentido Guvnor da soporte para el versionado a nivel de regla brindando las siguientes funcionalidades:

- Número de versión de la regla.
- Historial de versiones, en el que se pueden ver la fecha y comentarios correspondientes a la versión.
- Visualización y edición de determinada versión, con la posibilidad restablecer la regla a la misma, ya sea que esta se haya modificado o no.

5 Editor de reglas de negocio.

Como ya hemos aclarado, en el marco de este proyecto se utiliza el BRMS de JBoss, éste contiene entre otras cosas un motor de reglas y un gestor/editor de reglas de negocio (Guvnor). Sucede que éste se promociona como una herramienta apta y manipulable por cualquier tipo de usuario para su uso, ya sean estos técnicos o no. Luego de una investigación detallada de Guvnor resultó ser que desde nuestro punto de vista y del cliente, la generación de las reglas de negocio no es tan sencilla, esto es lo que puntualizaremos en la subsección 5.1. Luego en la subsección 5.2 describiremos el diseño de la solución planteada para cumplir con los requerimientos del Editor, y posteriormente veremos la arquitectura y la implementación en las subsecciones 5.3 y 5.4 respectivamente. Para finalizar el capítulo hablaremos del testing en la subsección 5.5.

5.1 Guvnor

Como mencionamos al inicio de este capítulo la interfaz Guvnor para la gestión de reglas tiene varias formas para expresarlas y gestionarlas, pero no resultan realmente amigables para el analista de negocio. Quizás sí se pueda llegar a que un analista de negocio pueda utilizarla pero sólo si se realiza una fuerte capacitación técnica, lo cual no siempre es viable y no permite la inserción de forma inmediata de nuevos usuarios para la definición y gestión de reglas.

El analista de negocio deberá para poder escribir las reglas, hacerse poseedor de un modelo de negocio desarrollado en Java, encapsulado en un Jar, para así poder adjuntarlo. Por lo que desde el comienzo aparecen conceptos técnicos.

Comenzaremos a ver algunas capturas de Guvnor, que nos sirvan de ejemplo para justificar nuestra postura acerca de la no amigabilidad para gestionar las reglas en dicha herramienta.

Luego de crear cierto ambiente de trabajo, como establecer el modelo de negocio a utilizar, el usuario comienza a crear sus reglas en Guvnor. Hay cuatro formas de expresar las reglas de negocio y esas son:

- Editor de reglas asistido gráficamente
- Escritura de reglas con DSL
- Escritura de reglas en DRL
- Escritura de reglas en tablas de decisión

El Editor de Reglas asistido es la imagen que se presenta a continuación en la figura 9 donde se ayuda a quien escribe la regla con las entidades y atributos en un entorno gráfico.

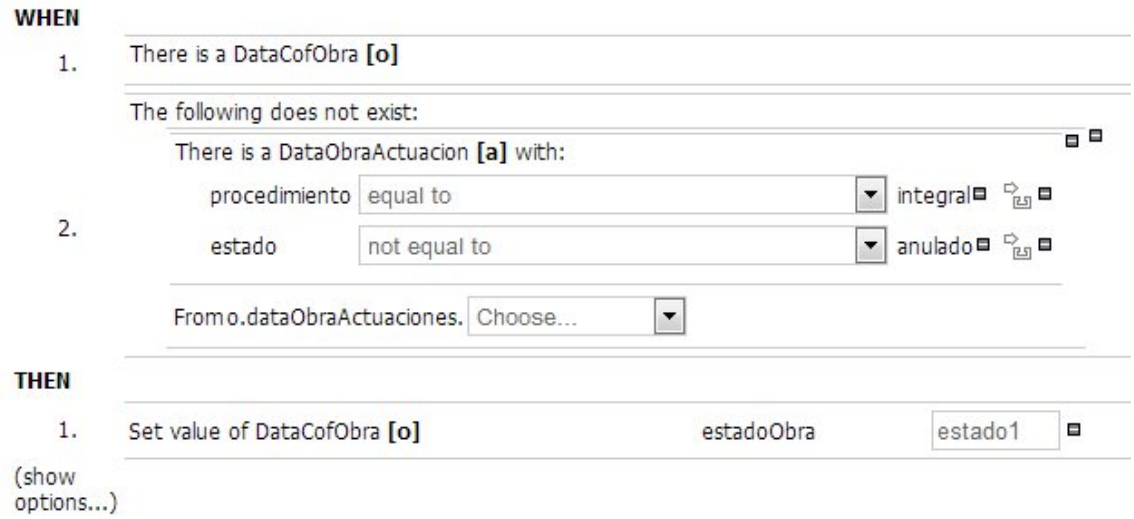

Figura 9: Imagen del ambiente grafico de Guvnor

En este caso vemos que existe una sección WHEN y otra sección THEN. En la primera se definen las condiciones que se deben cumplir para que se ejecuten las acciones definidas en la segunda sección. Hay una cantidad de variantes muy amplias para especificar condiciones, pero estas suelen ser poco amigables. Por ejemplo, en la figura 9 se puede ver en la sección del WHEN como se expresa una condición sobre un elemento de una colección (DataObraActuacion) perteneciente a una entidad principal (DataCofObra), esto no es visualizable por un usuario no técnico. Entre otras cosas, utiliza navegabilidad entre entidades utilizando el punto.

La segunda forma que mostraremos de definir reglas es mediante DSL (domain-specific language). Esto permite crear oraciones en lenguaje natural que tengan una representación lógica y nemotécnica de lo que significa, mostramos como se haría esta definición en Guvnor:

[when]
La cantidad de días trabajados de la obra tiene que ser menor o igual a {num:[1-9]?[0-9]} = o : DataCofObra(cantidadDiasTrabajados <= "{num}")*
[then]
La obra pasa al estado {estado:[A-Z]?[a-z0-9]} = o.setEstadoObra("{estado}")*

Como se puede observar, existe una parte que es en lenguaje natural (la primera parte) representada por la expresión regular que se encuentra entre corchetes, esto tanto en la sección *when* como en la *then*. Aquí aparece el concepto de *expresión regular* la cual no es trivial para un usuario funcional, por lo que a la hora de la definición de estas expresiones es necesaria la presencia de especialistas técnicos. Otro detalle no menor, es que de acuerdo a lo que hemos estudiado, estas oraciones no necesariamente cumplen con los estándares para las expresiones de reglas de negocio en lenguaje natural (plantilla RuleSpeak), queda por cuenta del autor de la regla cumplir con la plantilla o cuidar las ambigüedades. En la figura 10 se observa cómo quedaría la regla de forma definitiva.

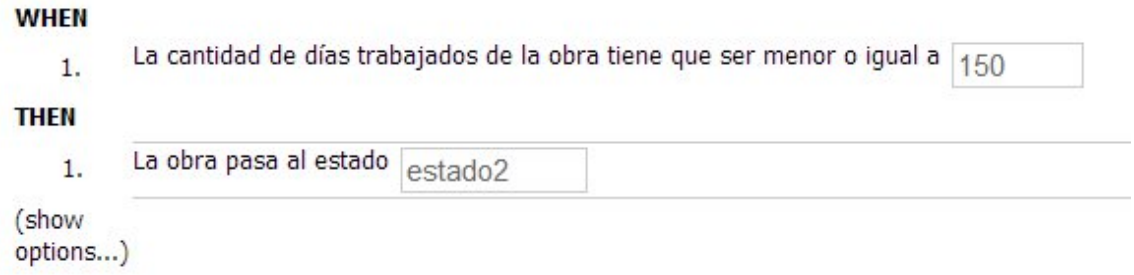

Figura 10: Imagen del ambiente de Guvnor donde se utiliza el DSL definido.

La tercera forma de definir reglas es escribiendo directamente en DRL, esto sería algo como lo que se puede observar en la figura 11.

```
rule "ReglaCorridaConObras"
    when
            datacorrida : DataCorrida ()
            datacofobra : DataCofObra (datacorrida.idCorrida == idCorrida
                    && estado != 2
                    && estadoObra == "Inicial")
    then
            datacorrida.setEstado(7);
end
```

Figura 11: Ejemplo de regla en DRL.

Es claro que esto es un lenguaje especial con determinada sintaxis y requiere de una fuerte capacitación para utilizarlo, hasta para un usuario técnico.

La cuarta forma de definir una regla y la última que presentaremos es mediante tablas de decisión, esta forma parece ser bastante poco ligada a la expresión de reglas en lenguaje natural y a su vez resulta ser bastante limitada. Mostramos en la figura 12 una tabla de decisión, donde se ponen las condiciones y acciones en cada columna y una regla queda determinada por lo que sería una fila.

Decision table

#	Description	diasTrabajados	tipoObra	estado
1		100	privada	Mayor a 100
2		150	publica	Mayor a 150
3		200	privada	Mayor a 200

Figura 12: Tablas de decisión en Guvnor.

Un elemento más a tener en cuenta es que una vez definidas las reglas, encontramos que aparecen conceptos como “Build” para la generación de un paquete de reglas listo para ser ejecutado, también se exige al usuario que se cree un “Snapshot” del paquete que generó para que el motor de reglas lo utilice, lo cual se suma a la argumentación de que aparecen conceptos técnicos.

Por último, el testeo de las reglas en Guvnor se realiza insertando objetos del modelo en memoria y llenando los atributos que correspondan para probar las reglas, todo esto a través de una interfaz gráfica. No es del todo imposible para el analista de negocio, pero al igual que en la creación de las reglas sería necesaria mucha capacitación o la ayuda continúa de algún técnico, ya que por ejemplo los mensajes de error no son del todo claros como para un analista de negocio. Este fue uno de los puntos flacos que también fue notado por el cliente (BPS).

5.2 Requerimientos

Todos los argumentos mencionados en la sección anterior nos motivaron a crear una solución distinta que permita la fácil expresión de las reglas por parte de un analista de negocio. Esto sería:

1- Poder escribir las reglas en un editor en lenguaje natural, utilizando términos del negocio y validando que cada una de las reglas escritas satisfacen con el estándar para la expresión de reglas en lenguaje natural (RuleSpeak). Además el proceso de escritura debe incluir intellisense de forma de ayudar con sugerencias al autor de las reglas.
2- Contar con un procedimiento para poder testear unitariamente la regla de forma dinámica utilizando cierto universo de datos, para que de esta forma se permita encontrar errores en la misma más tempranamente mediante la agilización del ciclo entre desarrollo y testeo unitario. Por ejemplo, la regla puede ser modificada y probada inmediatamente, o cambiar el universo de datos y volver a testear.
3- Acoplarse a las herramientas que se vienen manejando, es decir al BRMS de Jboss, para no perder todas las funcionalidades que ya brinda.

Para esto será necesario crear diferentes artefactos que interactúen entre sí y con las herramientas estudiadas brindando una solución unificada para la gestión de reglas de negocio que tenga como cualidad el fortalecimiento de los puntos débiles detectados durante el transcurso de investigación del proyecto.

En la figura 13 mostramos la interacción entre los artefactos a desarrollar para cumplir con los requerimientos mencionados anteriormente y las herramientas utilizadas en BPS.

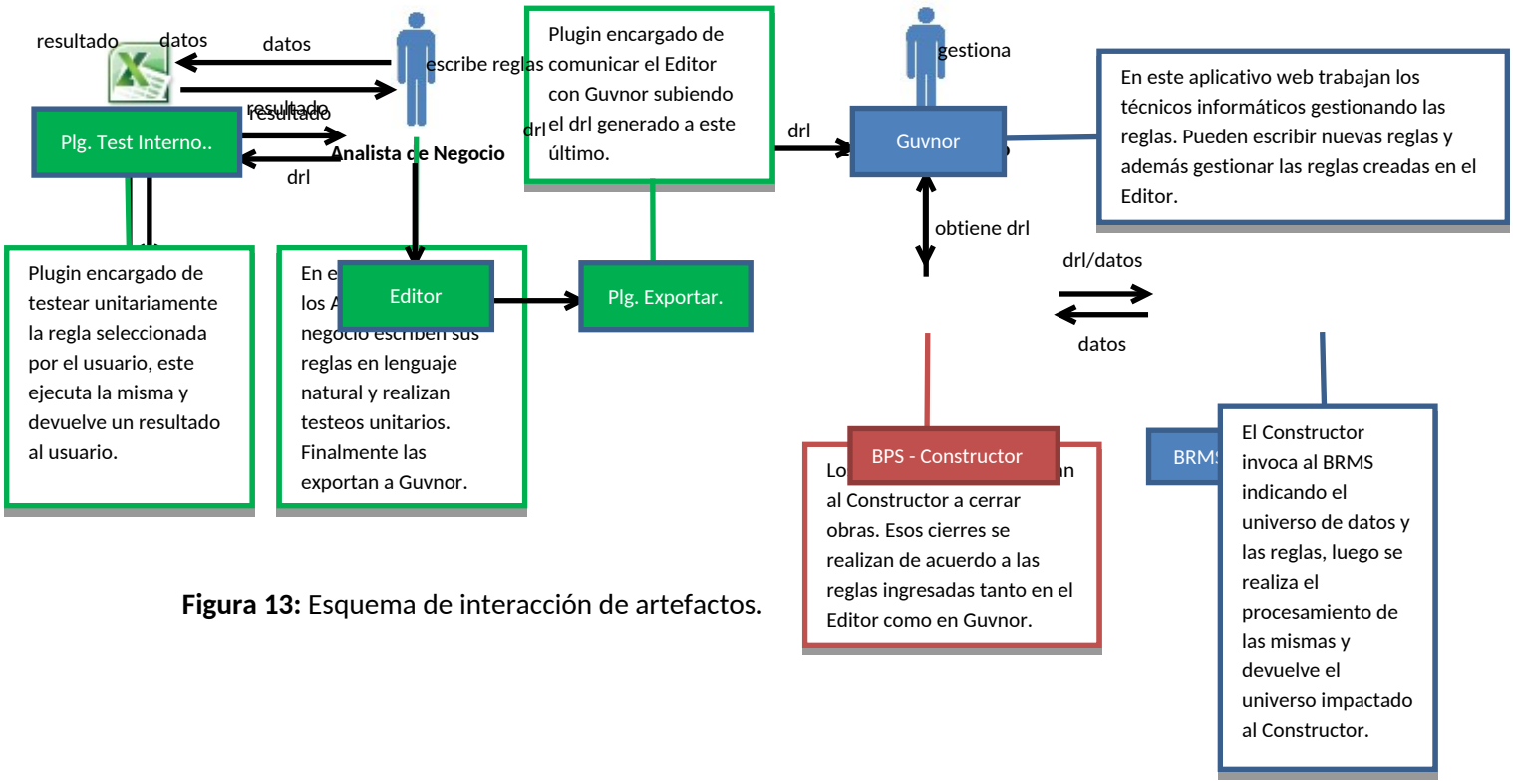

Figura 13: Esquema de interacción de artefactos.

El diagrama anterior contiene notas adjuntas que explican cada una de las partes. En el mismo hemos hecho diferenciación entre los diferentes componentes por colores. El color verde representa los artefactos implementados durante el proyecto de grado, los de color azul son artefactos de Jboss ya utilizados por el Constructor y con el cual se complementan y acoplan los componentes verdes, y por último, la caja roja es el Constructor, aplicativo desarrollado por BPS para el cierre de obras.

5.3 Diseño de la solución

El Editor de Reglas de Negocio será una aplicación que permitirá la expresión de reglas de negocio en lenguaje natural por el Analista de Negocio. Nos basamos en las recomendaciones de la plantilla RuleSpeak para ir confeccionando el texto que debería sugerir o escribir el usuario.
Los artefactos a implementar fueron tres principalmente uno de integración con Guvnor, el editor de reglas y hablaremos de ellos de forma más completa en las siguientes subsecciones.

5.3.1 Integración con Guvnor

En esta subsección hablaremos de la interacción del Editor de Reglas con Guvnor, este último contiene un modelo de negocio previamente definido el cual es descargado automáticamente.

También permite que una vez generado el DRL se pueda exportar el mismo a Guvnor, nuestro repositorio de reglas. Allí podrá ser manipulado al igual que cualquier otra regla creada desde esa herramienta. En la figura 14 se puede ver un diagrama que representa lo descrito anteriormente.

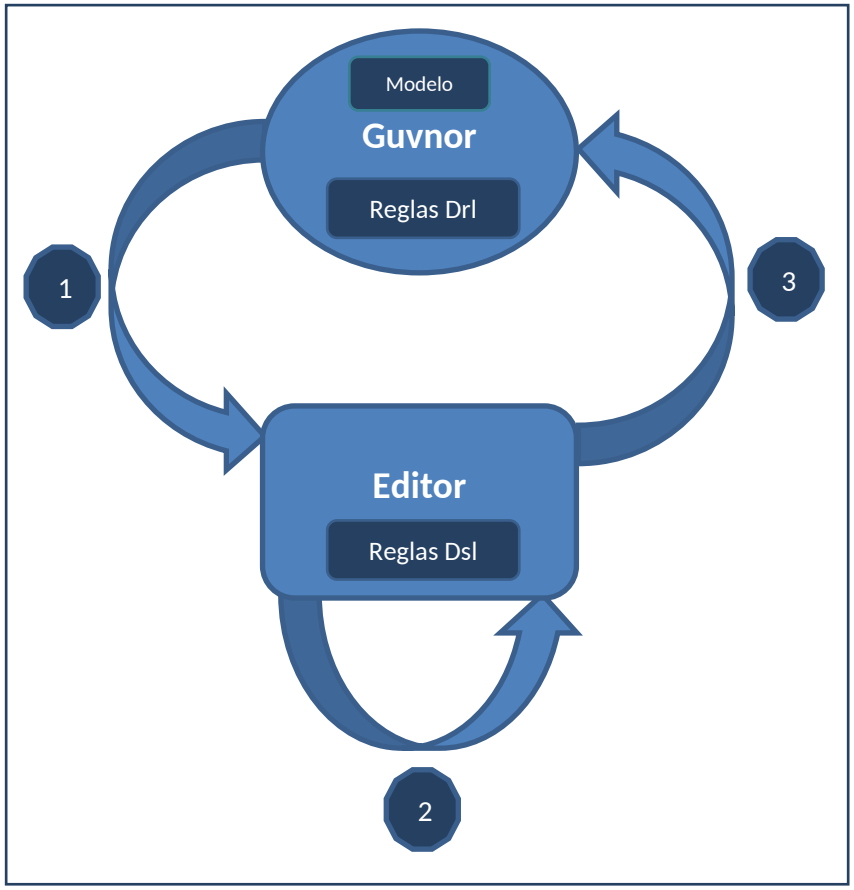

Figura 14: Diagrama interacción Editor-Guvnor.

Esto es lo que sucede en cada transición:

1) El modelo de Guvnor es descargado al Editor una vez que el mismo es abierto.
2) El usuario funcional escribe sus reglas de negocio, se generan los respectivos drl.
3) Una vez que el usuario funcional lo crea conveniente sube las reglas a Guvnor.

Para exportar las reglas será necesario hacer clic derecho sobre el drl de la regla que se quiere subir y elegir la opción del menú llamada ”Exportar regla a Guvnor”, una vez realizada esta acción se mostrará un mensaje al usuario donde se indica si la regla ha podido ser subida correctamente. La regla residirá en el paquete de Guvnor indicado.

Luego de realizada esa acción la regla ya forma parte de Guvnor, y esta pasa a cumplir su ciclo dentro de esta herramienta. Podrá ser ahora modificada, movida a otro paquete o pasada al ambiente de producción.

5.3.2 Especificación de la gramática para la edición de reglas

Este artefacto se basa en especificar una gramática definida en base a la plantilla RuleSpeak, que permite la escritura de las reglas en lenguaje natural que cumplan con este estándar. A su vez debe incluir un mecanismo de sugerencias que ayuden al usuario a expresar las reglas, donde se mezclaría el estándar con los términos propios del modelo de negocio. En caso de que las reglas no cumplan con el estándar se debe marcar con un error este problema. Una vez escrita la regla en forma correcta se generará el archivo DRL correspondiente a la misma, el cual es procesable por DROOLS. Buscando tecnologías que simplifiquen el desarrollo de este artefacto encontramos a Xtext.

Xtext es una herramienta open source para diseñar e implementar un lenguaje de programación propio o lenguajes específicos de dominio (DSL – Domain Specific Language), que cubre todas las necesidades planteadas en el requerimiento número 1. Cubre todos los aspectos de una infraestructura completa para la construcción de un lenguaje, incluye analizadores sintácticos, validadores, le da formato al código, contiene un compilador e intérprete generador de código, todo eso completamente integrado a la IDE de Eclipse. No tiene limitaciones como en el XML, se puede definir la sintaxis tan concisa y sugestiva como se quiera para mapear el dominio particular con el que se trabaja. Es desarrollado en el Eclipse Project como parte del Eclipse Modelling Framework, está incluido en el Eclipse como plug-ins.

Existió un intento de hacer un Editor a partir de la modificación del código fuente de Guvnor, pero luego de varios días de investigación llegamos a la conclusión de que esto requería de un costo de dificultad mayor.

Por lo que el Editor de Reglas se define como una aplicación de escritorio basada en Eclipse, y los tres principales artefactos fueron implementados como Plug-ins del mismo.

El primer paso de este proyecto es escribir una gramática para el reconocimiento de reglas de negocio en lenguaje natural guiados por las plantillas de RuleSpeak. Luego se implementan algunas validaciones sobre la escritura de las reglas, como por ejemplo validar que los atributos sobre los que se escribe en la regla pertenezcan a las entidades correspondientes. Además de las sugerencias a nivel del editor para ayudar al usuario a escribir las reglas, y por último el generador que traduce las reglas escritas en lenguaje natural a reglas de negocio en DRL, todo usando Xtext.

5.3.2.1 Descripción de la gramática básica

La plantilla de RuleSpeak plantea la forma de expresar todas las reglas de la clasificación del BRG en lenguaje natural, nosotros estudiamos como escribir las de nuestra clasificación y vimos que también se pueden escribir con lo planteado allí.

Concluimos que todas las reglas de la clasificación se pueden expresar indicando que algo es obligatorio o no lo es, y que algo está permitido o no lo está, utilizando las siguientes palabras en la regla:

- tiene que
- tiene que … si …

- ha de ser calculado como…
- ha de ser entendido como …
- tiene que ser ejecutado…cuando…
- tiene/está prohibido…
- tiene/está prohibido…si…
- tiene/está autorizado… sólo si…
- no tiene por qué…

Explicaremos el estudio para la regla que utiliza las palabras clave “tiene que” y el resto será ampliado en el anexo K.

La sentencia “tiene que” indica que algo es obligatorio de forma incondicional, y definimos que las reglas de este tipo deberían escribirse así:

Toda entidad **tiene que** tener el atributo <|>|<=|>=|==|!= que el atributo de entidad.

Siendo entidad alguna de las entidades descargadas del modelo de Guvnor y atributo algún atributo de esa entidad, los símbolos de comparación marcan las diferentes opciones a elegir.

Ejemplo: Toda obra **tiene que** tener la fecha desde <= fecha hasta.

A continuación presentamos algunas variantes:

Variante 1: Toda entidad **tiene que** tener el atributo. En la que se marca que el atributo no puede ser nulo, por ejemplo: Toda obra **tiene que** tener id de obra.

Variante 2: Toda entidad **tiene que** tener el atributo <|>|<=|>=|==|!= que valor. En donde la condición se evalúa con un valor ingresado en la regla que puede ser un texto, un número, una fecha, etc. Por ejemplo: Toda obra **tiene que** tener el id de obra ’>‘ que ‘0’.

Variante 3: Toda entidad **tiene que** tener (el atributo <|>|<=|>=|==|!= que el atributo de entidad) (**y** (atributo es <|>|<=|>=|==|!= que valor))*, en la que se pueden agregar otras condiciones de atributos y entidades con el agregado “y” la cantidad de veces que se necesite, y lo mismo agregando “o” entre las condiciones.

Variante 4: Toda entidad **tiene que** tener el atributo <|>|<=|>=|==|!= que la suma/multiplicación de atributo (+/* atributo)$^{+}$, pudiendo operar con otros atributos o valores.

En el editor que desarrollamos están incluidas todas estas variantes para las reglas que utilizan:

- tiene que
- tiene que … si …
- ha de ser entendido como aquel…que/cuyo…
- tiene que ser ejecutado…cuando…
- tiene/está prohibido…
- tiene/está prohibido…si…

Estas reglas serían de validación, derivación y ejecución de acciones, el editor es totalmente extensible para desarrollar los otros tipos de reglas y las otras palabras claves de la plantilla.

5.3.2.2 Intellisense

Intellisense viene de intelligent sense (sentido inteligente) y lo que hace es proporcionar una gama de opciones para facilitar la escritura al usuario y reducir malentendidos, errores tipográficos y otros errores comunes. Uno de los requerimientos del proyecto es que el Editor de reglas implemente intellisense, es decir, que a medida que el usuario va escribiendo las reglas vaya obteniendo ayudas.

Primero elige el tipo de regla que va a escribir y luego el editor va dando sugerencias sobre la estructura de la regla, también va a sugerir la lista de entidades obtenidas del modelo y los atributos que correspondan. Va a ir marcando los errores en la regla subrayando en rojo donde está y cuál es el error.

A continuación presentaremos un ejemplo de cómo sería la escritura de reglas, usando el modelo de negocio que utilizan en BPS. La regla que se escribirá es: "**Toda** DataCofObra **tiene que tener** ((**la propiedad** fechaHastaObra **mayor (>) que la propiedad** fechaDesdeObra) y (**la propiedad** estadoObra **igual al valor** "Iniciado")) **fin**".

Lo primero que se escribe es el nombre de la regla de negocio, eso es a elección del usuario. Luego se despliegan las opciones de las distintas reglas implementadas, para este proyecto implementamos cuatro tipos que son los que se ven a continuación:

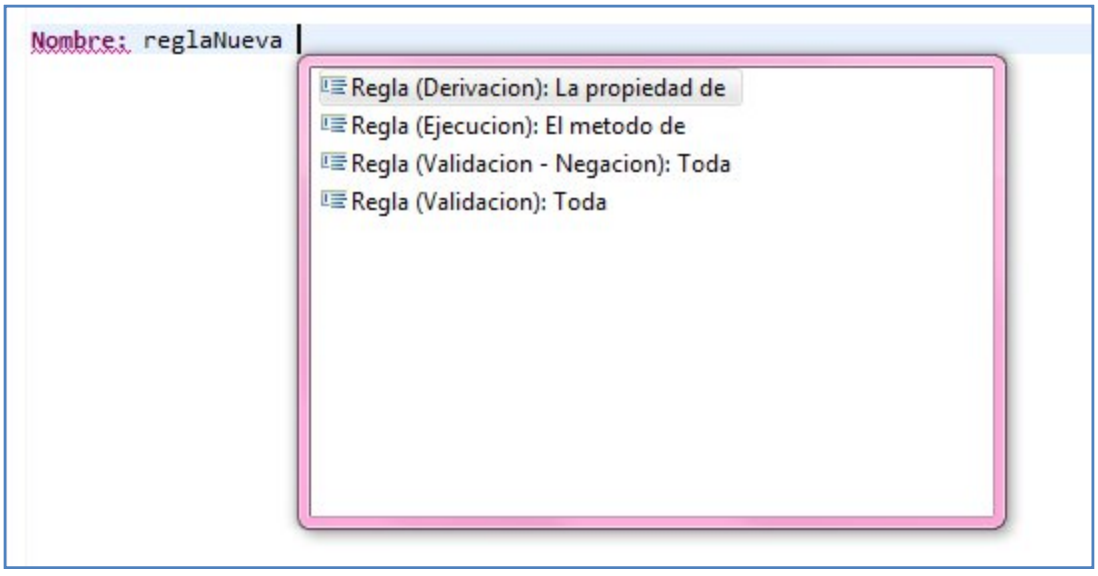

Figura 15: Elección en el Editor del tipo de regla a definir.

Luego de elegido el nombre de la regla, corresponde elegir la entidad del modelo para la cual se van a establecer condiciones. Se muestra en la figura 16 como se despliega una lista con todas las entidades existentes en el modelo:

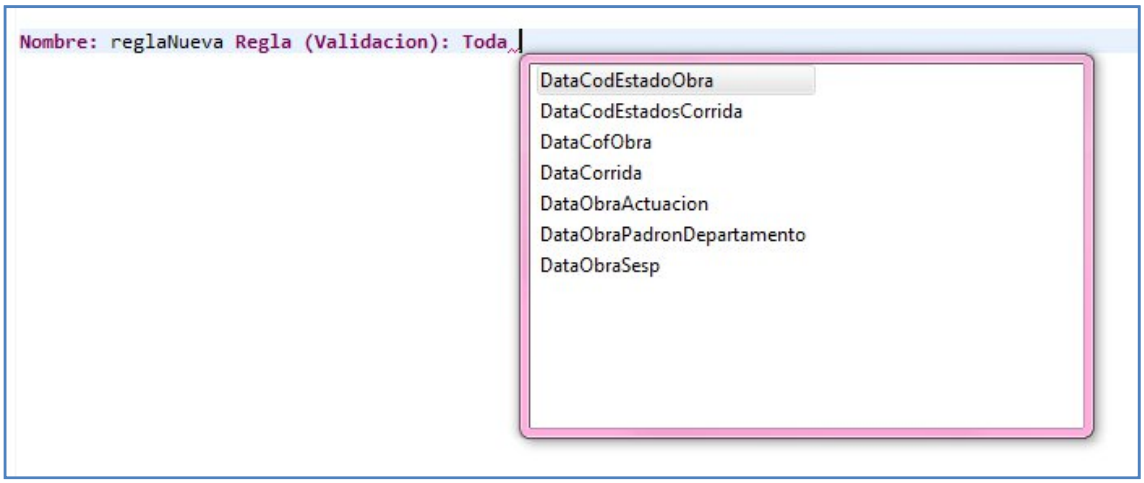

Figura 16: Elección en el Editor de la entidad sobre la cual se va a definir la regla.

Luego de elegida la entidad principal sobre la cual se tienen que cumplir las condiciones viene la palabra clave “tiene que tener” la cual se genera automáticamente, y comienzan a escribirse las condiciones. Las mismas se pueden escribir sobre un atributo de la entidad elegida, llamado “propiedad”, o sobre algún atributo de otra entidad distinta a la primera, llamando a las entidades “concepto”. Estas palabras que pusimos entre comillas fueron elegidas de forma de usar términos más entendibles para el Analista de Negocio, evitando así utilizar términos como entidad o atributo directamente en el lenguaje, los cuales son propios del contexto del dominio informático.

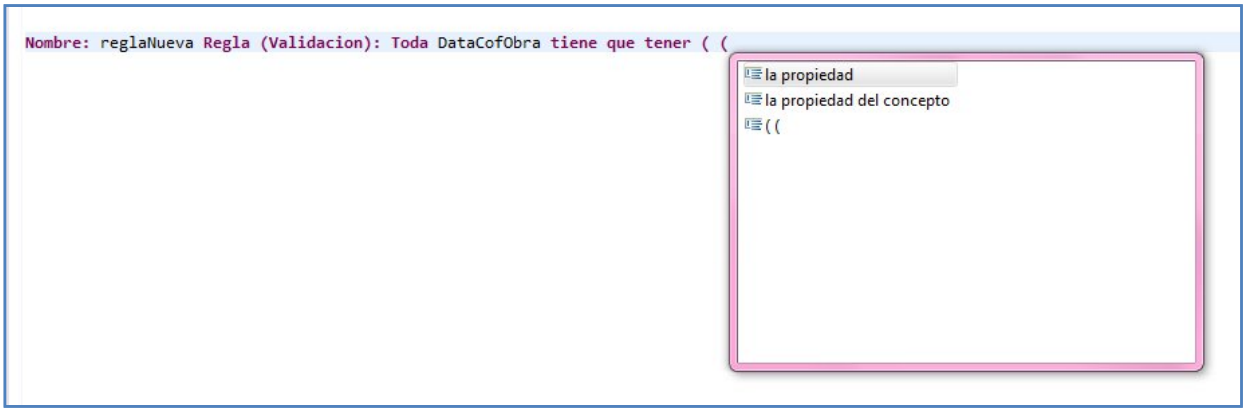

Figura 17: Elección en el Editor del tipo de atributo al cual se le va a exigir condiciones.

Cuando se selecciona escribir una condición sobre un atributo o propiedad, el editor despliega la lista de atributos de la entidad correspondiente:

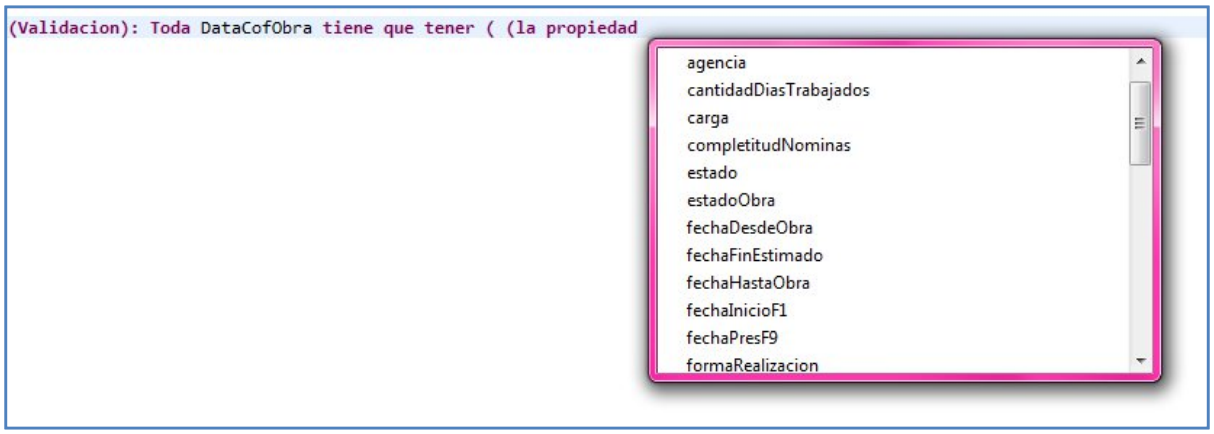

Figura 18: Elección del atributo para comenzar la condición.

Y luego de elegido el atributo se despliegan las opciones de comparación de ese atributo y luego las opciones para la segunda parte de la condición.

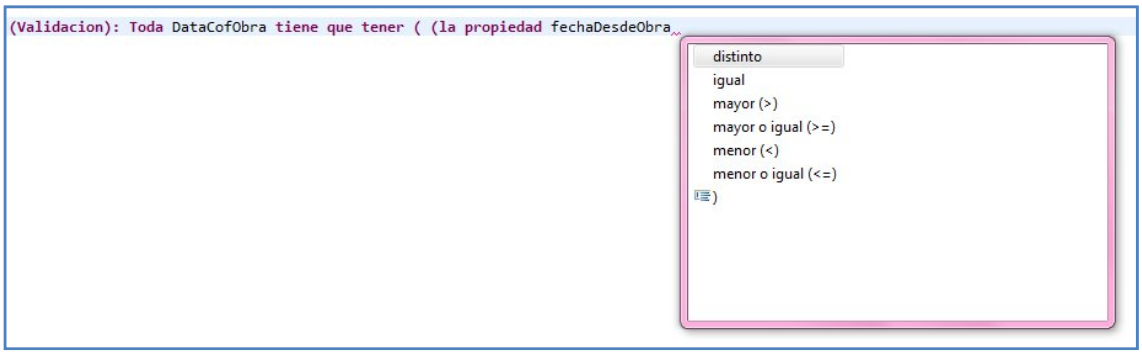

Figura 19: Elección del símbolo para la comparación de la condición mayor que.

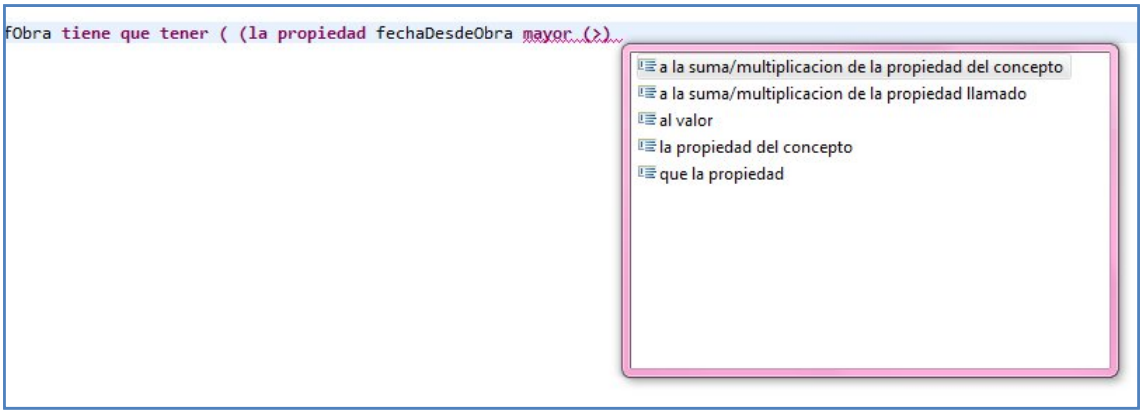

Figura 20: Elección de la segunda parte de la condición mayor que.

El operando de la derecha puede tomar los siguientes valores:

- a la suma/multiplicación de la propiedad del concepto: es para la comparación del total de una operación que incluye sumas o multiplicaciones y atributos de otra entidad diferente a la que se venía utilizando.
- a la suma/multiplicación de la propiedad llamado: es para la comparación del total de una operación que incluye sumas o multiplicaciones sobre atributos de la misma entidad que se estaba utilizando para las condiciones anteriores.
- al valor: comparar con un valor literal que va a ser ingresado por el usuario.
- la propiedad del concepto: compara la condición con un atributo de una nueva entidad distinta a la que se estaba utilizando.
- que la propiedad: se utiliza para comparar con otro atributo de la misma entidad que se venía trabajando.

5.3.2.3 Diseño de gramáticas compuestas

Consideramos que una gramática es compuesta cuando contiene más de una condición sobre la misma entidad, que implicaría agregar al finalizar la primera condición una "o" o una "y".
Notar en el ejemplo de la imagen 19, que en el Editor se abre un paréntesis al iniciar la condición y que luego de terminada una condición, se debe cerrar el paréntesis correspondiente. Una vez cerrado se pueden agregar otras condiciones con "o" o "y".

Los paréntesis no son algo natural para el usuario pero debimos utilizarlos dado que al día de hoy no se encontró una solución a nivel de lenguaje natural que permita agregar la disyunción y conjunción, las cuales son necesarias en las reglas, y que no genere un problema de ambigüedad.

Por ejemplo si tuviéramos la regla:

"Toda OBRA tiene que tener la propiedad estado igual a la propiedad de ACTUACION llamada estadoActuacion y la propiedad de ACTUACION llamada oficina menor o igual a 4 o la propiedad estado distinto la propiedad de ACTUACION llamada estadoActuacion y la propiedad de ACTUACION llamada oficina mayor a 6."

En el caso anterior no se puede distinguir si la condición deseada es:

- El estado de la obra es igual al estadoActuacion de la actuación y la oficina de la actuación es igual a 4 o (el estado de la obra es distinto al estadoActuacion de la actuación y la oficina de la actuación es mayor a 6).
- El estado de la obra es igual al estadoActuacion de la actuación y (la oficina de la actuación es igual a 4 o el estado de la obra es distinto al estadoActuacion de la actuación y la oficina de la actuación es mayor a 6).
- (El estado de la obra es igual al estadoActuacion de la actuación y la oficina de la actuación es igual a 4 o el estado de la obra es distinto al estadoActuacion de la actuación) y la oficina de la actuación es mayor a 6.

Entre otras variantes, por lo que la parentización sería necesaria.

Un ejemplo de cómo quedaría escrita una regla con dos condiciones en el Editor es la figura 21.

```
Nombre: reglaNueva Regla (Validacion): Toda DataCofObra tiene que tener
    ( (la propiedad fechaHastaObra mayor (>) que la propiedad fechaDesdeObra)
    y (la propiedad estadoObra igual al valor "Iniciado" ) ) fin.
```

Figura 21: Regla completa en el Editor de Reglas.

5.3.3 Test Unitario

El tercer artefacto nos permite realizar el testeo de forma unitaria de las reglas, esto se realiza desde el editor, donde es posible elegir una regla y testearla, este testeo tomará como universo un Excel donde estarán cargados los datos que correspondan para poder instanciar las entidades del modelo de negocio. El resultado del testeo quedará en el mismo Excel.

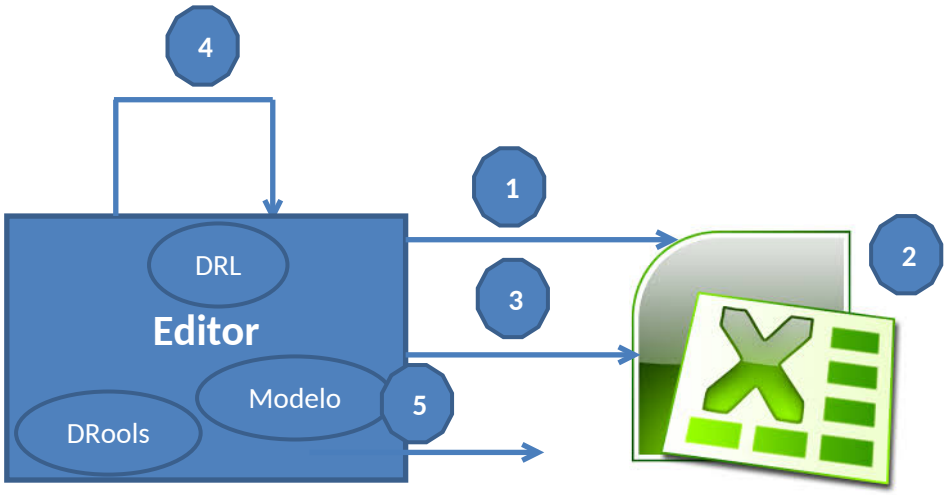

Figura 22: Ciclo de testeo de regla unitaria.

Describimos de qué consta cada paso de este ciclo:

1) En primer lugar el Editor crea un Excel con la descripción del modelo en su primera hoja.
2) Luego el usuario funcional carga los datos en la segunda hoja del Excel, con los cuales se realizará una instancia del modelo. Esto será llamado el universo de datos.
3) Posteriormente el Editor lee los datos del Excel y genera con ellos una instancia del modelo.
4) Una vez instanciado el modelo, utilizando DRools (que reside en el Editor) se ejecuta la regla correspondiente al DRL que está siendo testeado.

5) Finalmente, se escribe el resultado de la ejecución en la tercera hoja del Excel para que el usuario pueda realizar el análisis que corresponda.

Entraremos más en detalle en el diseño del artefacto.

En el momento en que se abre el Editor de Reglas de Negocio se disparan una sucesión de eventos, entre ellos se descarga el modelo desde un paquete de Guvnor. Otro evento que se dispara es la creación de una carpeta en el directorio donde se encuentra el Editor llamada *excelModelo*, dentro de esta se crea automáticamente una planilla Excel con el objetivo de que se carguen los datos del universo a testear en la misma. Esta planilla contiene en su primera hoja la especificación del modelo que se ha descargado, el fin de la misma es mostrar todas las clases del modelo, todos los atributos de cada clase y el tipo de datos que tienen asociados. El orden en que aparecen las clases y sus atributos servirá al usuario para poder expresar el universo de datos en la segunda hoja de la planilla. El tipo de datos que se especifica en los atributos de las clases deberá coincidir con el tipo de datos de la celda en el que se cargaran los valores que sean pertinentes.

Para ver de forma gráfica cual será el flujo de eventos y acciones por parte del usuario presentamos de Diagrama de Secuencia del Sistema en la figura 23.

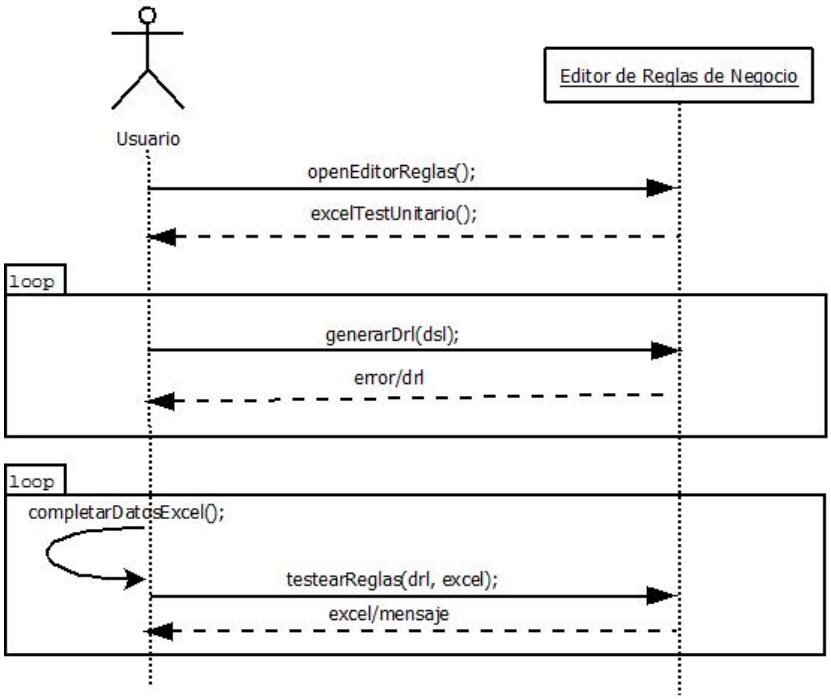

Figura 23: DSS del Testing Unitario.

Mostramos un ejemplo de lo que sería la primera hoja del Excel con el modelo de datos:

Objeto	DataCofObra				
Atributo	idObra	Simple	Integer		
Atributo	agencia	Simple	Integer		
Atributo	cantidadDiasTrabajados	Simple	Integer		
Atributo	departamento	Simple	Integer		
Atributo	fechaDesdeObra	Simple	Date		
Atributo	fechaHastaObra	Simple	Date		
Atributo	fechaPresF9	Simple	Date		
Atributo	fechaInicioF1	Simple	Date		
Atributo	numeroEmpresa	Simple	String		
Atributo	numeroObra	Simple	String		
Atributo	saldoCuenta	Simple	Integer		
Atributo	padron	Simple	String		
Atributo	unidad	Simple	String		
Atributo	duracionObra	Simple	Integer		
Atributo	dataObraPadronDepartamento	Entidad	DataObraPadronDepartamento		
Atributo	dataObraActuaciones	ColeccionEnt	DataObraActuacion		
Objeto	DataObraPadronDepartamento				
Atributo	codDepartamento	Simple	Integer		
Atributo	nombreDepartamento	Simple	String		
Objeto	DataObraActuacion				
Atributo	idObraAct	Simple	Integer		
Atributo	estado	Simple	String		
Atributo	numeroActuacion	Simple	Integer		
Atributo	tipoEstado	Simple	String		

Figura 24: Ejemplo de un modelo de datos.

Como se puede observar son necesarias las cuatro primeras columnas de la hoja para especificar el modelo y tantas filas como sean necesarias para representarlo, recordar que esta hoja la crea el Editor al iniciarse.

Para profundizar en el orden en que se encuentran los datos en la primera hoja, y el significado de cada columna, dirigirse al ANEXO L: Manual de usuario Testing Unitario.

Esta primer hoja sirve como referencia para que el usuario funcional escriba los datos del universo que se deben encontrar en la segunda hoja del Excel, mostramos una captura de una pequeña instancia de un objeto para mostrar cómo se debe llenar esta hoja.

ent	DataCorrida	1		02/03/2012			01/01/2012	31/12/2012		1	2-2	
ref	DataCofObra	1	100	130			5	01/04/2012	20/12/2012	04/05/2012	10/05/2012	22/12/2012
ref	DataObraActuacion	1	homologado	1	integral		1					
ref	DataObraActuacion	2	homologado	2	integral		1					
ref	DataObraSesp	1		VD5	226	1						
ref	DataObraSesp	2		GR7	227	1						

Figura 25: Ejemplo de instanciación de un modelo de datos.

En el Anexo L también se explica la forma en que se debe llenar esta hoja.
Una vez que se ha llenado la segunda hoja del Excel es posible testear la regla. Para hacer lo anterior será necesario dirigirse al Editor de Reglas de Negocio y realizar clic derecho encima

del drl correspondiente a la regla a testear, se desplegará el menú contextual en donde hay que elegir la opción “Testear regla”. Una vez realizado esto comenzará a ejecutarse la regla utilizando el universo de datos cargados en el Excel.

En caso de que no haya ocurrido ningún error en la ejecución entonces se desplegará el siguiente mensaje en el Editor: “*La regla se ha ejecutado correctamente*, *verifique los resultados en el Excel*”. El Editor deja siempre en estos casos un resultado en la tercera hoja del Excel.

En caso de que la regla que está siendo testeada sea de *Validación* entonces pueden suceder dos cosas, que la condición de la regla se haya cumplido o no. Cualquiera de los dos resultados se verifica en el Excel donde estará alguno de los siguientes dos mensajes: “*La condición se ha cumplido*.” o “*La condición no se ha cumplido*.”.

En caso de que la regla sea de *Ejecución*, o sea, que en la sección *Then* del drl se invoca a un método, entonces en la tercera hoja del Excel aparecerá el mensaje: “*El método se ha invocado correctamente*.”.

Por último, en caso de que la regla sea de *Derivación*, se mostrará en la tercera hoja del Excel la instancia de la entidad principal que debió ser modificada, esto sirve para corroborar que la derivación ha sido exitosa o no. Mostramos una captura de cómo quedará expresado el resultado.

DataCorrida	
idCorrida	1
estado	7
fechaUltAct	02/03/2012
numeroSesp	
oficina	0
fechaDesde	01/01/2012
fechaHasta	31/12/2012
usuarioUltAct	
conF9	1
dataCofObras	

Figura 26: Resultado de la ejecución del testing.

Como se puede observar en la figura 26, el orden en que aparecen es el mismo en que aparecen expresados en la primera hoja del Excel.

Todo lo expresado anteriormente es en el escenario de que la ejecución de la regla resulta exitosa, en caso contrario, se desplegarán mensajes de error indicando cual fue el problema. Para profundizar en los diferentes tipos de error que pueden ocurrir dirigirse al Anexo L.

En la siguiente sección estaremos hablando sobre la arquitectura. Se mostrarán diferentes vistas para explicar la arquitectura del sistema.

5.4 Arquitectura

En esta sección, se describirá el mediante diferentes vistas del modelo 4+1 [18], la arquitectura del sistema. Este modelo se utiliza para describir la arquitectura de un sistema software basado en el uso de múltiples puntos de vista. Las vistas que se presentarán son las siguientes:

- Vista de componentes.
- Vista lógica.
- Vista de despliegue.
- Vista de casos de uso.

El modelo 4+1 también incluye una vista de procesos pero entendimos que para este proyecto no es necesario incluirla. Antes de mostrar el diagrama de cada vista haremos una breve descripción acerca de que se constituyen.

5.4.1 Vista componentes

En esta vista se muestra el sistema desde la perspectiva de un programador. Se va a mostrar cómo está dividido el sistema software en componentes y las dependencias que hay entre esos componentes.

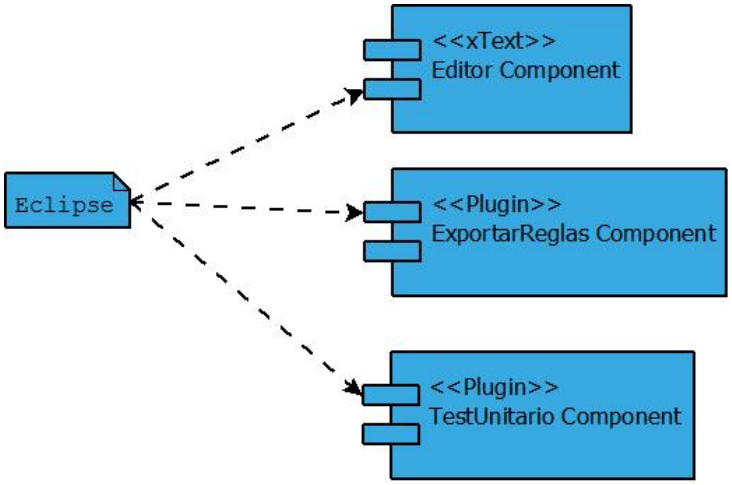

Figura 27: Diagrama de componentes.

La figura 27 muestra una vista donde simplemente se pueden observar los tres componentes principales del Editor de Reglas de Negocio. Cada uno de los tres componentes será incluido dentro de un Eclipse en forma de plugin. Los componentes que fueron representados son:

- Editor Component: Comprende la gramática del Editor de Reglas de Negocio. Contiene toda la lógica del intellisense para la sugerencia asociadas al modelo de Guvnor. Encargado de bajar el modelo de Guvnor y de crear la versión inicial del Excel para el testeo unitario.
- ExportarReglas Component: Resuelve la exportación del drl generado por el componente anterior a Guvnor.

- TestUnitario Component: Encargado de procesar el Excel con datos del universo a testear en las pruebas unitarias y de instanciarlo. Pasa por un motor de reglas ejecutando la regla en cuestión para el universo previamente instanciado. Escribe el resultado de la ejecución de las pruebas.

5.4.2 Vista lógica

En esta vista se representan de qué paquetes y módulos constan las funcionalidades que el sistema proporcionará a los usuarios finales.

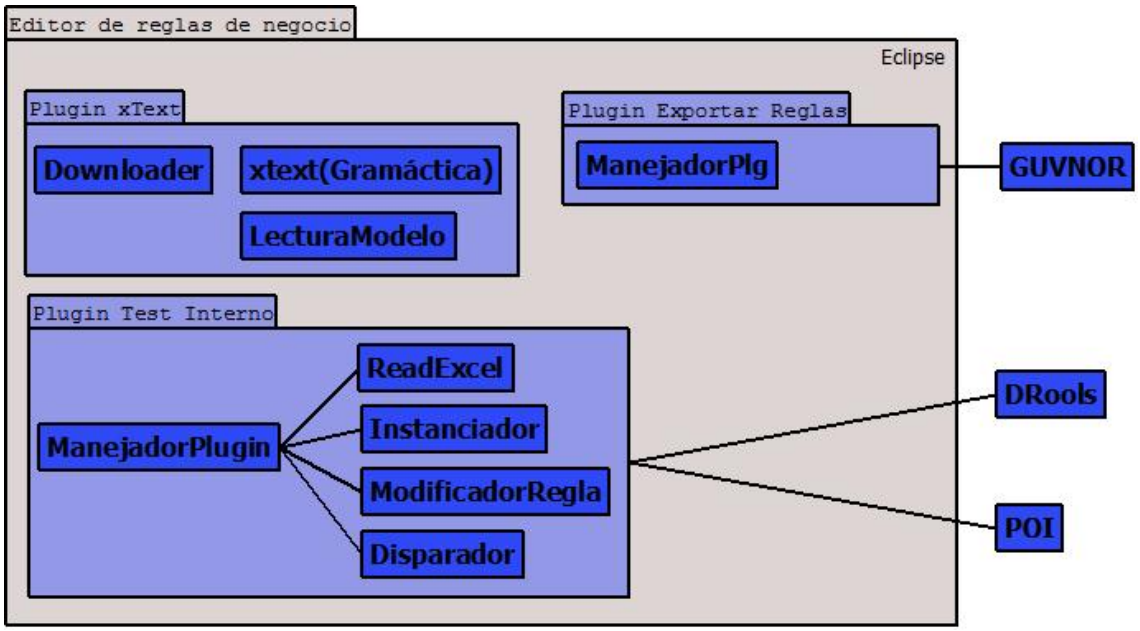

Figura 28: Vista lógica.

La figura 28 representa una vista lógica del Editor de reglas de Negocio. Como se puede ver existe un paquete, que es el Eclipse, que envuelve a cada uno de los tres paquetes desarrollados. Se pueden ver las interacciones con entidades ajenas al Editor de Reglas de Negocio.

El paquete *Plugin xText* se corresponde con el *Editor component* del diagrama de componentes, el mismo contiene los siguientes módulos:

- Downloader: Se encarga de descargar el modelo de Guvnor.
- xText(Gramática): Módulo que contiene la elaboración de la gramática. Encargado de realizar la traducción de las tiras de la gramática a drl. Por otro lado se encuentra la implementación del Intellisense, en esta parte entre otras cosas se incorporan los elementos del negocio del modelo para que puedan ser sugeridos en el momento de definir una regla.
- LecturaModelo: Lee el modelo descargado por el *Downloader*, el mismo será utilizado por la parte de Intellisense para sugerir elementos del negocio en la gramática.

El paquete *Plugin Test Interno* se corresponde con el *TestUnitario Component* el cual contiene los siguientes módulos:

- Manejador Plugin: Es el manejador que captura el evento de 'clic derecho' sobre la regla a testear. Dispara las acciones para que se ejecute la regla sobre el universo de datos que se encuentra en el Excel.
- ReadExcel: Lee y mantiene en memoria los datos leídos del Excel.
- Instanciador: Se encarga de realizar una instancia del modelo a partir de los datos leídos anteriormente.
- ModificarRegla: Modifica el drl de la regla para que esta pueda ser ejecutada.
- Disparador: Ejecuta la regla a testear con el universo de datos proporcionado utilizando DRools.

Este paquete se comunica con DRools desde el módulo Disparador que se encarga de ejecutar la regla para la regla a testear y además, utiliza las librerías POI que permiten realizar la interacción de java con Excel, esto último en el módulo ReadExcel.

Finalmente el paquete *Plugin Exportar Reglas* se corresponde con el componente *ExportarReglas Component* del diagrama de componentes, el mismo contiene un único modulo llamado ManejadorPlg. Este módulo se encarga al igual que el manejador del paquete Plugin Test Interno de capturar el evento de 'clic derecho'. Luego de capturado el evento se subirá a Guvnor el drl seleccionado.

5.4.3 Vista de despliegue

En esta vista se muestra desde la perspectiva de un técnico todos los componentes físicos del sistema a nivel macro, así como las conexiones entre esos componentes que conforman la solución.

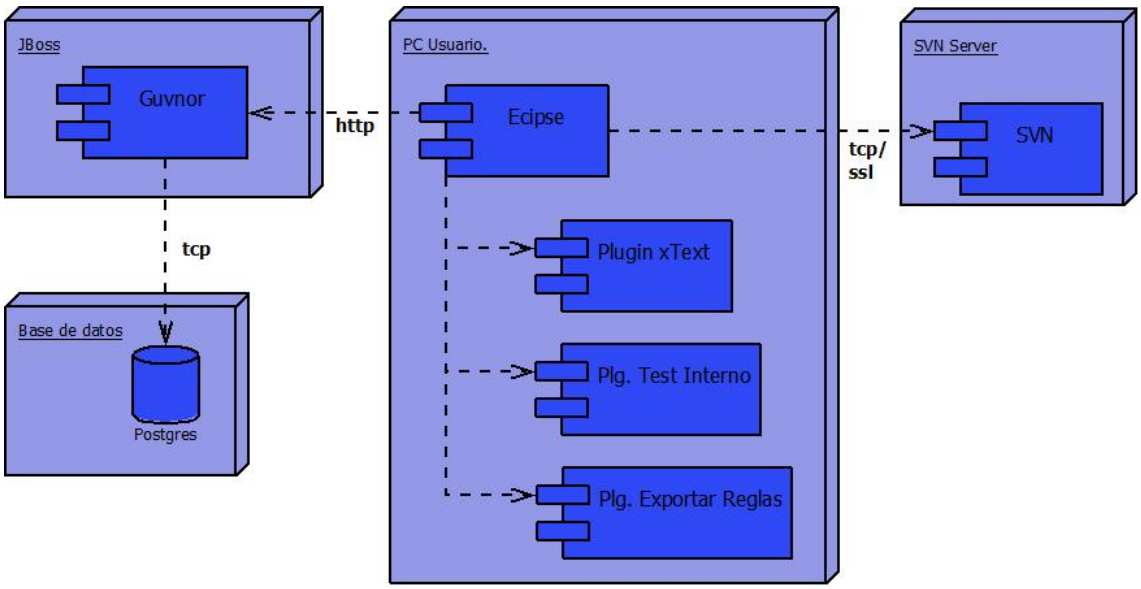

Figura 29: Diagrama de Deploy.

El Diagrama de Deploy consta de 4 nodos físicos los cuales contiene diferentes paquetes. Estos nodos pueden residir en lugares físicos independientes. Vamos a describir cada uno de los nodos describiendo brevemente de que constan:

- PC Usuario: Es el PC donde se encuentra instalado el Editor, en el reside el Eclipse que contiene los tres Plugins en forma de jars que se han desarrollado, ellos son:
  - Plugin xText: Contiene todo lo referente a la gramática y a la generación de las reglas.
  - Plugin Test Interno: Brinda soporte para el testeo unitario de cada regla mediante planillas Excel.
  - Plugin Exportar Reglas: Permite exportar las reglas desde el Editor a Guvnor.

  Se comunica con el nodo *Jboss* mediante http con el uso de WebDav.
- JBoss: Servidor de aplicaciones donde reside Guvnor. En caso de que Guvnor utilice una Base de Datos este nodo tendrá comunicación el nodo donde reside la misma mediante tcp.
- Base de Datos: Reside la Base de Datos utilizada por Guvnor.
- SVN Server: Servidor donde se encuentra el sistema de control de versiones donde residirán las reglas escritas por el usuario.

5.4.4 Vista de casos de uso

Esta vista va a ser representada por los casos de uso software y va a tener la función de unir y relacionar todas las vistas, esto quiere decir que desde un caso de uso podemos ver cómo se van ligando las demás vistas.

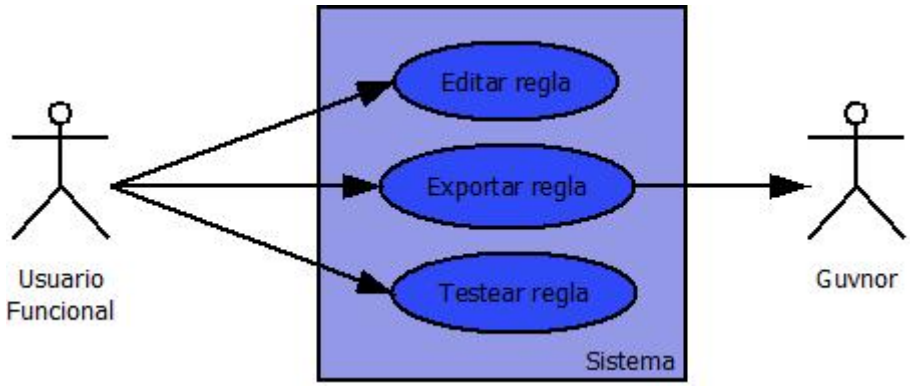

Figura 30: Diagrama de Deploy.

Como se puede observar, el usuario funcional consume los casos de uso:

- Editar regla.
- Exportar regla.
- Testear regla.

Se puede apreciar como el caso de uso *Exportar regla* tiene una interacción con el actor Guvnor.

5.5 Implementación

Como se explicó anteriormente el Editor de Reglas se define como una aplicación de escritorio basada en Eclipse, y los tres principales artefactos fueron implementados como Plug-ins del mismo.

Para desarrollar los Plugins se utiliza el soporte que Eclipse brinda para estos efectos. Tanto el plugin desarrollado para Exportar Reglas como el desarrollado para realizar el Testing Unitario son del mismo tipo, ya que ambos constan de implementar un manejador para despachar el evento que se dispara al elegir una opción del menú contextual desplegado sobre un drl. Cada uno de los Plugin debe incluir en su ClassPath los jars que necesitan, por ejemplo, el plugin de Testing Unitario debe incluir el jar de DRools.
Una vez finalizado el desarrollo de un plugin, este es exportado a un jar el cual se incluye en la carpeta plugins del Eclipse como cualquier otro plugin.

En esta sección lo que estaremos realizando será explicar los detalles de implementación de cada uno de los tres artefactos desarrollados y se explicará cómo se realiza la comunicación entre los diferentes componentes.

Esta sección se dará los detalles suficientes para una posible extensión de los artefactos.

En primer lugar encontramos necesario especificar la existencia de un archivo de configuración a nivel general del Editor de Reglas de Negocio. En el mismo se registran entradas que son consumidas por cualquier de los tres artefactos implementados. Este se encuentra en una carpeta llamada *properties* en el directorio donde está instalado el Editor, el mismo se llama *config.properites*.

Cómo se llama el modelo y en donde está el mismo para bajarlo automáticamente desde Guvnor se encuentran como propiedades del archivo de configuración, por lo que cuando el usuario abre el editor comienza a escribir las reglas con el modelo actualizado.

5.5.1 Implementación Editor de reglas

Como ya hemos desarrollado, el Editor de reglas de negocio está compuesto por un artefacto para la creación de reglas. Ese artefacto es un plugin de Eclipse que llamamos plugin xtext porque fue desarrollado con Xtext. La figura 31 muestra un diagrama con los paquetes de este Plugin.

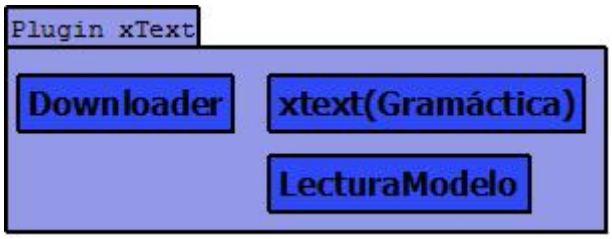

Figura 31: Diagrama de Paquetes del plugin Xtext.

En primer lugar se crea un proyecto del tipo xText(es un proyecto Plugin), que además del nombre y las típicas configuraciones del mismo, se define cual será la extensión del archivo del cual posteriormente se hace la traducción, en este caso traducción a DRL y también el nombre del lenguaje.

En este mismo proyecto, está definido el método init en una clase llamada *MyDslFactoryImpl* que es invocado cuando se abre el Editor, este se encarga de comunicarse con la clase *Downloader* para descargar el modelo de Guvnor a una carpeta del Eclipse utilizando WebDAV. Una vez descargado el modelo las clases *UnZip* y *LeerClasesAtributos* se encargan de levantar en memoria una estructura con todos los nombres de las entidades y todos los atributos de las mismas. Finalmente la clase *CreadorExcel* crea el Excel para el testeo unitario con los datos del modelo en su primera hoja.

En la figura 32 mostramos un diagrama de comunicación donde queda más claro el proceso que ocurre al abrirse el Editor.

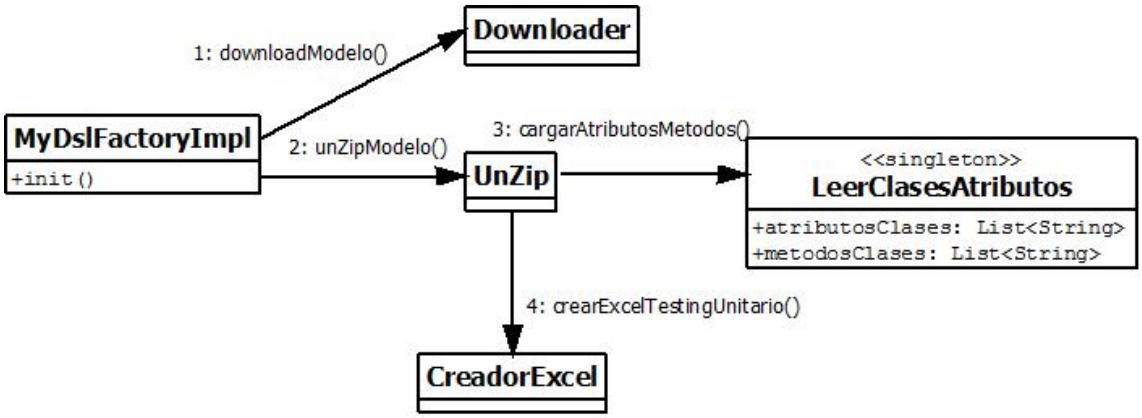

Figura 32: Diagrama comunicación métodos init de la clase MyDslFactoryImpl.

Este proyecto xText creado es el que contiene el DSL, es decir la definición de la gramática y todos los componentes para la ejecución (analizador sintáctico, léxico, linker, validación, etc). Cuando se crea el mismo, además de crearse el proyecto con el nombre elegido, se crean otros dos automáticamente. Uno de ellos es el nombre del proyecto más un punto seguido de la palabra tests, donde van los test unitarios. Y el otro es el nombre del proyecto más un punto seguido de la palabra ui, donde como su nombre lo indica, contiene todo el código centrado en la interfaz de usuario, es decir el editor, el entorno, el contenido de ayuda, etc.

5.5.1.1 Creación de la gramática

En el marco del proyecto de grado creamos un proyecto xtext llamado MyDsl, cuando se crea este proyecto se crea un archivo de ese mismo nombre y extensión xtext, en ese archivo es donde se definen reglas para la gramática que en nuestro caso van a generar reglas de negocio.

La primer regla que se define es la regla que se utiliza como inicio para el lenguaje, en este caso la llamamos regla porque para esta implementación de DSL son reglas de negocio. Como

ya dijimos en un mismo archivo se pueden crear varias reglas para eso tiene el símbolo de + al final que indica que el lenguaje acepta una o más de esas reglas. Otra cosa que ya hemos comentado es que para el alcance del proyecto implementamos cuatro variantes, pero es posible la extensión de todas las restantes simplemente agregando nuevas reglas al lenguaje. A continuación se presenta la regla inicial con las cuatro variantes del editor:

```
Regla:
    regla += (TieneQueTener | TieneProhibido | HaDeSerEntendidoComo | HaDeSerEjecutadoCuando)+
;
```

Si tomamos como ejemplo la regla para la variante que usa “tiene que”:

```
TieneQueTener:
        "Nombre:" nombreRegla=QualifiedName
        "Regla (Validacion): Toda" entidad=Entidad "tiene que tener"
        condEntPpal=(CondicionEntidadPpal) andorExter+=AndOrNuevaEntidad*
        si=(CondicionSi)? "fin."
;
```

Primero se exige que el usuario escriba un nombre para la regla de negocio, que es totalmente libre al usuario y esto es común para todas las variantes. Luego en el editor aparece la opción de elegir cuál de las variantes se escribirá, por lo que si se elige el texto “Regla (Validacion): Toda” se estaría eligiendo la regla que utiliza la palabra clave “tiene que”. La regla del lenguaje llamada *Entidad* permite que el usuario ingrese un texto, que luego es validado en otra clase que explicaremos más adelante. En el capítulo 4 se mostró como es el uso del editor, mostramos que se pueden escribir condiciones sobre una entidad o comparar atributos de una entidad con otra, la regla *CondicionEntidadPpal* se encarga de la escritura de las condiciones referidas a la entidad ingresada en la regla Entidad, y la regla *AndOrNuevaEntidad* se encarga de las condiciones que son de otras entidades y tienen que manejar el “y” y “o” para agregar otras condiciones. Por último la regla *CondicionSi* permite agregar más condiciones y se termina la regla con el terminal “fin”.

Sin continuamos mirando las reglas de la gramática que forman la regla de validación, la primera que aparece es la de condición:

```
CondicionEntidadPpal:

                "( (" ( atributoEntidad=CondicionAtributoEntidad | ( "la propiedad "
                atributo=Atributo))

                  ((comparacion=ComparacionInterna? ")" andor+=(AndOrInterno)*) |
comparacion=ComparacionExterna ")" andorext+=(AndOrInternoNuevaEnt)*)")"
;
```

En el primer renglón de la regla aparece la bifurcación entre escribir una condición sobre otra entidad o escribir un atributo de la entidad que se había elegido anteriormente en la regla *Entidad*. Si se eligiera la rama de la regla *Atributo*, el usuario escribiría un texto que será validado entre la lista de atributos de la entidad seleccionada. Luego en la regla comparación interna se termina de definir la condición:

```
ComparacionInterna:
        simbolo=Simbolo
        comparaciones=(AtributoAtributo|AtributoValor|OperacionAtributo);
```

Donde la regla *Simbolo* es un terminal con los símbolos de las comparaciones:

```
Simbolo:
        "igual a" | "distinto" | "mayor o igual (>=)" | "mayor (>)" | "menor (<)" | "menor o
igual (<=)"
;
```

Y las comparaciones pueden ser de un atributo con otro atributo, un atributo con un valor literal, o un atributo con el resultado de una operación entre atributos y valores:

```
AtributoAtributo:
        "que la propiedad" atributo=Atributo;

AtributoValor:
        "al valor" valor=ValidValue
;

ValidValue:
        INT|STRING
;

OperacionAtributo:
        "a la suma/multiplicacion de la propiedad llamado" atributo=Atributo
operaciones+=(OperacionComplemento)+
;

OperacionComplemento:
        op=("+" | "*") atributo=(Atributo | Valor)
;
```

Luego en la regla *CondicionEntidadPpal* continua la regla *AndOrInterno* que se encarga de agregar la disyunción y conjunción al lenguaje. Ésta regla reutiliza las reglas anteriores para formar más condiciones de comparación.

```
AndOrInterno:
        cond=("y (" | "o ("(atributoEntidad=CondicionAtributoEntidad|("la propiedad "
atributo=Atributo)) comparacion=ComparacionInterna? ")"
;
```

Al finalizar la escritura de las condiciones referidas a la entidad principal, es decir a la primera que se mencionó, se puede continuar agregando condiciones sobre otras entidades que tienen que evaluarse simultáneamente a la entidad principal, eso se puede agregar gracias a la regla *AndOrInternoNuevaEnt*.

Para ver la gramática completa diríjase al Anexo F.

Dentro del proyecto MyDsl, se genera una clase automáticamente llamada *MyDslJavaValidator* en la cual se realizan validaciones sobre la tira que se va generando de la gramática. En nuestro caso se implementamos en esa clase validaciones sobre tres reglas, la llamada *Entidad*, *Atributo* y *Metodo*. Cuando se inicializa el plugin de Xtext se descarga de Guvnor el modelo indicado en el archivo de configuraciones ya mencionado en el capítulo anterior, y se listan por ejemplo las entidades en ese modelo. La validación se realiza verificando que lo que ingreso el usuario en la regla exista realmente en el modelo.
En el proyecto MyDsl.ui se genera automáticamente una clase llamada *MyDslProposalProvider* en las que implementamos las sugerencias que se le brinda al usuario en pantalla. Esta clase extiende de *AbstractMyDslProposalProvider* y para implementar sugerencias se debe sobrescribir el método que corresponda, por ejemplo para dar las sugerencias de la lista de entidades validas en la regla *TieneQueTener*, se debe sobrescribir el método llamado *completeTieneQueTener_Entidad*. Uno de los parámetros en ese método es el *ICompletionProposalAcceptor acceptor*, en el cual se agregan todas las palabras aceptadas para la regla *TieneQueTener* en la variable *Entidad*.

En el ambiente de ejecución, luego de ser escrito y validado el archivo con la extensión deseada para la gramática, se genera la traducción al lenguaje deseado, en este caso DRL.

5.5.1.2 Generación de DRL a partir de la gramática con Xtend.

Dentro del proyecto MyDsl, luego de generar las reglas de la gramática hay que dirigirse al paquete llamado *generator* y dentro del mismo a la clase llamada *MyDslGenerator*. Para generar la traducción se debe implementar el método *doGenerate*, y para eso se itera sobre todos los contenidos, y a medida que se van identificando las reglas se va imprimiendo en un archivo con extensión drl. El ejemplo para la regla “TieneQueTener” sería:

```
override void doGenerate(Resource resource, IFileSystemAccess fsa) {
	for (r : resource.allContents.toIterable.filter((typeof(TieneQueTener)))) {
		fsa.generateFile(r.nombreRegla +".drl", r.compile)
	}
}
```

Y luego se define el método compile para la regla TieneQueTener:

```
def compile(TieneQueTener r)'''
	/* Regla: Validacion */
	rule "«r.nombreRegla»"
		when «calcularAlias(r.entidad.nombre)»
		«IF r.si != null» «r.condEntPpal.compile(r.si, r.entidad.nombre)»
		«ELSE»«r.condEntPpal.compile(r.entidad.nombre)»
		«ENDIF»
		«FOR regla:r.andorExter»
			«regla.compile(r.entidad.nombre.toLowerCase+contadorEntidades.get(r.entidad.nom
		bre.toLowerCase))»
		«ENDFOR»
		then
			System.out.println("Se ejecuto la regla «r.nombreRegla» con exito");
	end
'''
```

Donde el texto azul es lo que va directamente en el archivo generado, y el resto son iteraciones sobre las condiciones de la regla y llamados a métodos compile que traducen de

forma recursiva de la misma manera que la regla. Para cada regla de la gramática xtext genera una clase java, y se utilizan los métodos get y set para ir obteniendo que ingresara el usuario y a partir de eso decidir que se traduce. A estos métodos se invoca automáticamente una vez que el usuario guarda el archivo con extensión mydsl en el editor de reglas, por lo que automáticamente se genera el archivo drl.

5.5.2 Implementación del Artefacto Test Unitario.

La implementación de esta herramienta fue realizada como un plugin del Eclipse. Esto se resolvió de esta forma ya que pretendíamos que fuera una funcionalidad incluida dentro del Editor de Reglas de negocio, que como ya explicamos está basado en Eclipse. El esquema de la figura 33 representa un paquete con las clases del artefacto para el testeo de las reglas.

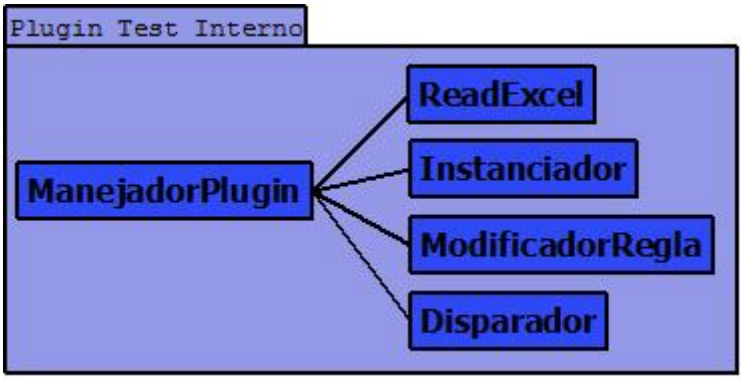

Figura 33: Diagrama de paquete de Plugin Test Interno.

Como se puede observar la arquitectura es bastante simple y pasaremos a describir cada uno de los módulos representados en el esquema anterior.
La clase que se encuentra a la izquierda del diagrama es el llamado "*ManejadorPlg*", este manejador se encarga de capturar el evento correspondiente al 'clic' en la opción "*Ejecutar prueba unitaria*" del menú contextual del drl generado. El manejador lo que hace es invocar uno a uno los demás elementos que aparecen en el siguiente diagrama de la figura 34 en este orden:

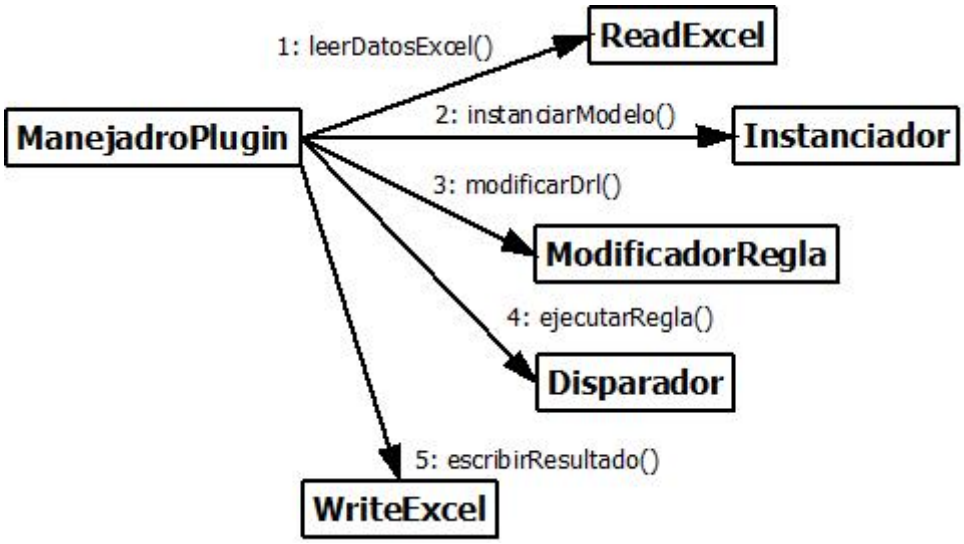

Figura 34: Diagrama de comunicación testear regla.

Describiremos una a una las clases para describir la implementación de los mismos.

ReadExcel: Esta clase se encarga de leer la primera hoja del Excel utilizando las librerías POI [25] para el manejo de planillas en Java, esto abarca tanto la lectura como la escritura. Como explicábamos en secciones anteriores la primera hoja del Excel contiene una descripción del modelo de datos que se encuentra en Guvnor, será entonces necesario leer esta hoja y almacenar una descripción de cada una de las entidades para luego poder hacer la instanciación correspondiente mientras se lee la hoja de datos. El modelo que se utiliza para almacenar esta información se puede observar en la imagen 35.

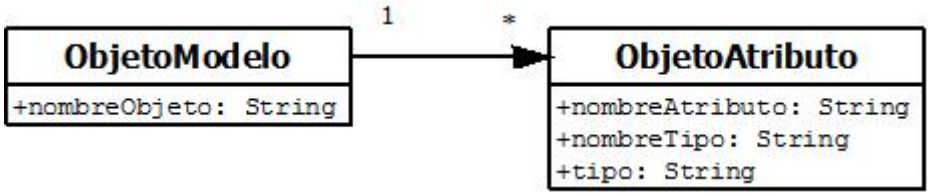

Figura 35: Representación del modelo en Excel.

Se mantiene una colección de *ObjetoModelo* donde esta clase representa a cada una de las entidades del modelo. Un *ObjetoModelo* contiene una colección de *ObjetoAtributo* por cada atributo que tenga la entidad que se está describiendo. De cada atributo(*ObjetoAtributo*) nos interesa preservar es el nombre, el nombre del tipo(Simple, Entidad, EntidadColeccion) y el tipo propiamente dicho.

Instanciador: El Instanciador comienza a leer la segunda hoja del Excel, en cuanto encuentra una fila con la entrada *ent* se encarga de instanciar el objeto que corresponda. Una vez leído el nombre de la entidad se recupera la información del *ObjetoModelo* que corresponda (cargado por el *ReadExcel*) y se crea una instancia de la entidad utilizando *Reflection*. Se procede a leer una a una las columnas con los atributos del objeto y con la información de los *ObjetoAtributo* se van realizando las instancias por *Reflection* y seteos necesarios, recordar que el orden del listado de los *ObjetoAtributo* se corresponden con el orden en el que están expresados los

atributos de un objeto en la hoja 2. La instanciación se realiza de forma recursiva ya que tanto en el caso de que el atributo se corresponda con una entidad o con una colección será necesario instancias otros objetos, con lo cual se invoca al Instanciador de forma recursiva para la fila que corresponda.

ModificadorRegla: Se encarga de tomar el archivo drl al que se le ha realizado cilc derecho y copiarlo a una carpeta local del Editor de Reglas de Negocio. Esta copia será el drl que se ejecutará el cual tendrá algunas variaciones que se pasaron a describir. La primera modificación que se realiza es la inclusión de los *import* de las entidades de negocio que pertenecen al drl, eso es necesario para la ejecución de la regla. Hemos creado una nueva entidad que formará parte del universo de datos a ser testeado, esta entidad se llama *EntidadResultado* y nos sirve para almacenar el resultado de la ejecución de nuestras reglas, para que esto suceda se debe agregar código tanto a la sección del when como en el then por lo que nuestras reglas deben contener los agregados que se muestran en cursiva:

```
rule ……..
when
        er: EntidadResultado();
        ……..
then
        er.setMensajeResultado("La condicion se ha cumplido satisfactoriamente");
        ……….
end;
```

El constructor de EntidadResultado contiene el mensaje del resultado vació, esto nos permite comparar una vez ejecutada la regla si está cumplió la condición del *when* o no.

Disparador: Encargada de interactuar con Drools aparte del *Instanciador*, este último se encargó no solo instanciar sino de agregar cada nueva instancia y una *EntidadResultado* a la sesión donde se insertarán los objetos para que formen parte del universo de a ser evaluado por el motor de reglas. Al *Disparador* solo le resta disparar la regla que corresponda capturando cualquier tipo de error que pueda llegar a ocurrir.

WriteExcel: Encargada de plasmar el resultado del testeo en la tercera hoja del Excel. En caso de que la regla no sea de derivación consulta la *EntidadResultado* y según el contenido de esta escribe si la condición se ha cumplido en forma satisfactoria o no. Por el contrario, si la regla es de derivación comienza leer el universo de datos, que a esta altura ya está impactado por la ejecución de la regla, y escribe el contenido de las entidades de forma que el usuario pueda corroborar que el universo se haya impactado de forma correcta. Para el alcance de este proyecto, solo se escriben los datos de la entidad principal del modelo, que es la primera entidad que aparece descripta en el Excel.

DataCorrida	
idCorrida	1
estado	7
fechaUltAct	02/03/2012
numeroSesp	
oficina	0
fechaDesde	01/01/2012
fechaHasta	31/12/2012
usuarioUltAct	
conF9	1
dataCofObras	

Figura 36: Resultado de la ejecución del testing en el Excel.

5.5.3 Implementación del Artefacto Exportar Reglas a Guvnor.

Como en el caso de la herramienta anterior esta fue desarrollada como un Plugin de Eclipse, en este caso su desarrollo fue bastante más sencillo.

Figura 37: Paquete del plugin de exportación de reglas.

El componente *ManejadorPlg* es el encargado de realizar todo el trabajo. Lo primero que hace es identificar el archivo sobre el cual se está realizando clic derecho, una vez detectado el mismo se procede a leer su contenido y levantarlo en memoria. Finalmente lo que se hace es incorporar la regla levantada a memoria a Guvnor utilizando WebDAV. Hay que tener en cuenta que la implementación utiliza el archivo de configuración *config.properties*, las entradas que utiliza en este caso son las siguientes:

- paqueteGuvnor=Prototipo

Este es el paquete de Guvnor al que se sube la regla.

- pathGuvnor=localhost:8080/guvnor-5.4.0.Final-jboss-as-7.0/org.drools.guvnor.Guvnor

Esta es la url donde se encuentra deployado Guvnor.

- userGuvnor=admin

Usuario de Guvnor.

- passGuvnor=admin

Password de Guvnor.

Luego de que la regla ha sido subida le aparecerá un mensaje al usuario indicando que la misma ha sido exportada de forma exitosa.

5.6 Testing

El testeo en el desarrollo del Editor de Reglas de Negocio estuvo basado principalmente en testear la gramática que genera el drl a partir de una regla de negocio escrita en el Editor.

Mostramos a continuación un caso de cómo se llevó a cabo el testeo para una regla de validación a modo ejemplo. Como se puede ver en la imagen 33 lo primero que hicimos fue definir la regla.

```
Nombre: testRules Regla (Validacion): Toda DataCofObra tiene que tener
    ( (la propiedad cantidadDiasTrabajados menor o igual (<=) al valor 150 ) )
fin.
```

Figura 38: Regla creada en el Editor de Reglas de este proyecto.

Una vez escrita la regla verificamos que se haya generado el drl correspondientes en el Editor:

```
/* Regla: Validacion */
rule "testRules"
    when
    datacofobra1 : DataCofObra (
    cantidadDiasTrabajados <=
    150
    )
    then
        System.out.println("Se ejecuto la regla testRules con exito");
end
```

Figura 39: Regla traducida del lenguaje a DRL.

Luego exportamos la regla a Guvnor, verificamos que la misma se haya incorporado al paquete indicado en el archivo de configuración del Editor:

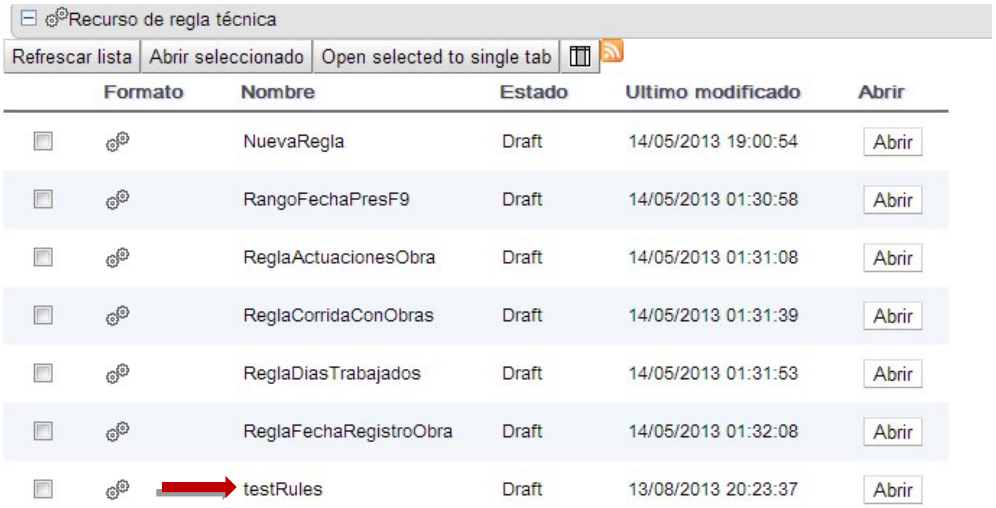

Figura 40: Regla 'testRules' exportada correctamente a Guvnor.

El siguiente paso es verificar la sintaxis de la regla en Guvnor para asegurarse de que lo que generó lo gramática no contiene errores:

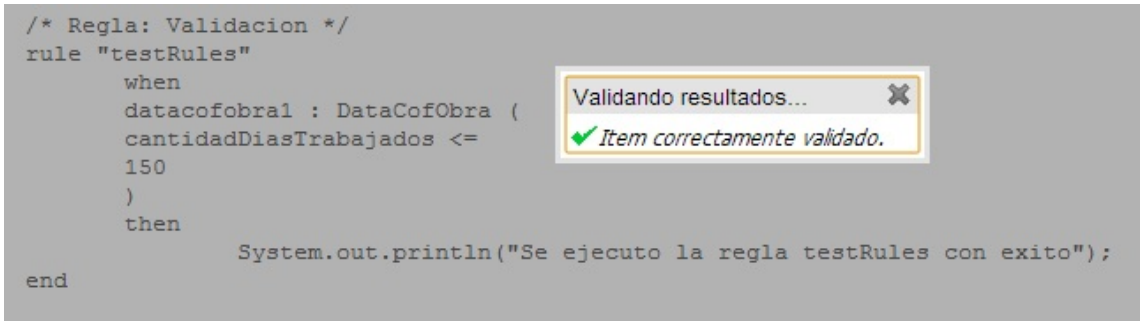

Figura 41: Regla correctamente validada en Guvnor.

Con estos pasos bastaría para decir que la gramática fue bien generada para esta variante de tipo Validación. El conocimiento del lenguaje drl nos permite saber por otro lado que a nivel semántico lo que está expresado en el drl es lo que se escribe en el dsl, lo importante es que la gramática genere el drl correcto sintácticamente, eso es lo verdaderamente importante a la hora de testear el Editor como software. Lo que queremos decir es que no debe confundirse este testeo con el testeo de las regla a nivel semántico.

Es importante comentar que este testeo se llevó a cabo para los diferentes tipos de regla: Validación, Validación – Negación, Ejecución y Derivación.

6 Instanciación para el Constructor

Hoy en día Atyr cuenta con un sistema llamado Constructor para el cierre de obras, para mostrar el resultado de incorporar reglas de negocio al sistema utilizamos alguna de las entidades que se utilizan. Modificamos la parte web del Constructor para mostrar el resultado de la ejecución de las reglas. El modelo subido a Guvnor surge de exportar las entidades del constructor, y estas son:

- **DataObraSesp**: Datos del trámite asociado a la obra.
- DataCodEstadoObra: Estado de la obra, estos pueden ser por ejemplo: “Cerrada”, “No pasó los filtros”, “No hay datos en recaudación”.
- **DataObraActuacion**: Se registra un número de “Actuación” en caso de que exista.
- **DataCorrida**: Una corrida es una ejecución de un cierre de obras, en esta entidad se registra un número asociado a la misma, la fecha de ejecución y otros datos que permiten saber si la misma fue exitosa.
- **DataCodEstadosCorrida**: Estado de la corrida, estos pueden ser por ejemplo: “Iniciada”, “Cargando”, “Finalizada”.
- **DataCofObra**: Se registran los datos correspondientes a una obra, está asociada a las entidades: DataCodEstadoObra, DataObraActuacion, DataCodEstadosCorrida, DataObraPadronDepartamento.
- **DataObraPadronDepartamento**: Datos del padrón y el departamento correspondientes a una obra.

En la siguiente figura mostramos un modelo con las clases de forma de poder visualizar mejor cómo se relacionan entre sí.

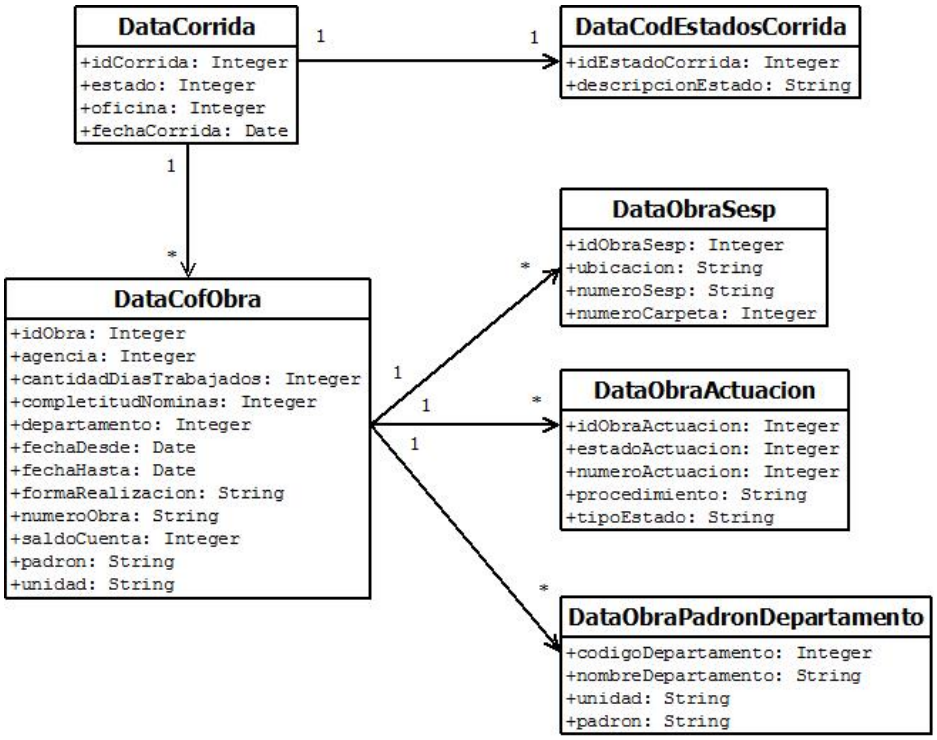

Figura 42: Modelo de datos del prototipo para Constructor.

Al proyecto web que utiliza Richfaces se le agregó una grilla para mostrar los datos de las obras, y de esta forma además mostrar si la regla modifico algún valor, o si se cumplió o no alguna validación.

Se implementaron en el Editor desarrollado para este proyecto reglas que usan en Atyr hoy en día para el cierre de obras, se subieron a Guvnor, y desde un archivo de configuraciones se indicó de que paquete descargar las reglas que van a ser utilizadas en el Constructor.

A continuación presentamos las reglas implementadas en el Editor:

// Las obras candidatas a ser cerradas tienen prohibido tener actuaciones integrales no anuladas en el sistema Guia-Gdt
Nombre: ReglaActuacionesObra **Regla (Validacion - Negacion):**

Toda DataCofObra **tiene prohibido tener (**
(la propiedad idObra **igual la propiedad del concepto** DataObraActuacion **llamado** idDataCofObra **)**
y (la propiedad del concepto anterior estado **distinto al valor** "anulado" **)**
y (la propiedad del concepto anterior procedimiento **igual al valor** "integral" **)) fin.**

// La cantidad de días trabajados de la obra tiene que ser menor o igual a 150.
Nombre: ReglaDiasTrabajados **Regla (Validacion):**
Toda DataCofObra **tiene que tener ((la propiedad** cantidadDiasTrabajados **menor o igual (<=) al valor** 150 **)) fin.**

// La duración de la obra tiene que ser menor o igual 12 meses si la obra es privada.

Nombre: ReglaDuracionObra **Regla (Validacion):**
Toda DataCofObra **tiene que tener ((la propiedad** duracionObra **menor o igual (<=) al valor** 12 **)) si ((** tipoObra **igual al valor** "privada" **)) fin.**

// La obra tiene prohibido tener algún trámite en las oficinas 226, 229 y 2004.
Nombre: ReglaTramitesOficina **Regla (Validacion - Negacion):**
Toda DataCofObra **tiene prohibido tener ((la propiedad** numeroObra **igual la propiedad del concepto** DataObraSesp **llamado** idDataCofObra)
y (la propiedad del concepto anterior ubicacion **distinto al valor** 226 **)**
y (la propiedad del concepto anterior ubicacion **distinto al valor** 229 **)**
y (la propiedad del concepto anterior ubicacion **distinto al valor** 2004 **)) fin.**

// La fecha desde registrada de la obra tiene que ser mayor o igual octubre de 2006.
Nombre: ReglaFechaRegistroObra **Regla (Validacion):**
Toda DataCofObra **tiene que tener ((la propiedad** fechaInicioF1 **mayor o igual (>=) al valor** "01-oct-2006" **)) fin.**

// Las obras candidatas a ser cerradas tienen que tener la fecha de presentación de F9 comprendida entre la fecha desde y la fecha hasta definidas por el usuario si éste ha optado por realizar cierres "ConF9".
Nombre: RangoFechaPresF9 **Regla (Validacion):**
Toda DataCofObra **tiene que tener ((la propiedad** idCorrida **igual la propiedad del concepto** DataCorrida **llamado** idCorrida **)**
y (la propiedad del concepto anterior fechaDesde **menor (<) que la propiedad** fechaPresF9 **)**
y (la propiedad del concepto anterior fechaHasta **mayor (>) que la propiedad** fechaPresF9 **)**
y (la propiedad del concepto anterior conF9 **igual al valor** 1 **)) fin.**

Y además agregamos estas reglas para probar el Editor y la aplicación con reglas distintas a las de validación:

Nombre: ReglaConF9 **Regla (Ejecucion):**
El metodo de DataCorrida **llamado** obtenerObrasConF9 **ha de ser ejecutado cuando ((la propiedad** conF9 **igual al valor** 1)) **fin.**

Nombre: ReglaCorridaConObras **Regla (Derivacion):**
La propiedad de DataCorrida **llamado** estado **ha de ser entendido como (valor)** 7 **si ((la propiedad** idCorrida **igual la propiedad del concepto** DataCofObra **llamado** idCorrida)
y (la propiedad del concepto anterior estado **distinto al valor** 2)
y (la propiedad del concepto anterior estadoObra **igual al valor** "Inicial")) **fin.**

Luego de utilizar los filtros que ya tienen en el Constructor hoy en día, se obtiene la lista de obras que lo cumple. En la aplicación del BPS no se muestra cual fue el resultado de aplicar los filtros, lo cual fue una de las modificaciones que hicimos para poder mostrar el resultado de ejecutar las reglas. Luego de filtrar se procede con el botón *Ejecutar* inicializando la llamada al motor de reglas.

Figura 43: Resultado de filtrar las obras existentes con las condiciones indicadas en la parte superior.

En la acción del botón *Ejecutar* se procede creando la base de conocimiento en donde se cargan las reglas indicadas en un archivo de configuración:

```
KnowledgeBase knowledgeBase = createKnowledgeBase();
```

Luego se crea una sesión stateless y se invoca al motor de reglas de jboss:

```
StatelessKnowledgeSession session = knowledgeBase.newStatelessKnowledgeSession();

ArrayList<Command> commands = new ArrayList<Command>();

commands.add(CommandFactory.newFireAllRules());

session.execute(CommandFactory.newBatchExecution(commands));
```

Como se puede apreciar por el código anterior, incluir en una aplicación el comportamiento a través de reglas no agrega complejidad alguna. Simplemente se agrega Drools a las librerías del proyecto y las líneas de código que se mostraron en esta sección.
El Editor de reglas implementado para este proyecto tampoco agrega complejidad dado que las reglas se consumen de Guvnor que es indistinto al origen de las mismas.

Para finalizar se carga en pantalla el resultado de la ejecución, cerrando la obra si se cumplen todas las reglas, o mostrando en la columna *Estado* de cada obra que reglas no se cumplieron.

Figura 44: En la columna Estado queda el resultado de ejecutar las reglas.

Figura 45: Cuando se selecciona el botón '+' de la columna Estado se muestra en detalle que reglas no se ejecutaron.

El código fuente del Constructor con las modificaciones anteriormente mencionadas se encuentra en la carpeta "Instanciación Constructor".

7 Gestión del proyecto

Más allá de lo planificado como alcance al comienzo del proyecto, a medida que se fue investigando y avanzando sobre el tema, se fueron realizando reuniones frecuentemente. En estas reuniones se fue volcando encima de la mesa lo investigado en cuanto a la temática de las reglas de negocio, de forma de tal de que junto al cliente se fueron trazando nuevas líneas de investigación. Esta mecánica de reuniones con el cliente fue conformando el alcance final del proyecto y se pudo definir cuáles eran los resultados esperados del proyecto.

Luego de esta etapa comenzó la etapa de elaboración de los entregables, los cuales eran enviados al cliente. El cliente leía lo documentado para finalmente reunirse a realizar la devolución correspondiente, en algunos casos de algunos documentos se realizaron varias iteraciones.

Una vez finalizados los entregables de documentación se comenzó desarrollar el Editor de Reglas de Negocio y la herramienta para realizar el testeo unitario. En una primera instancia se intentó incorporarlo a Guvnor pero el costo en tiempo de modificar esta herramienta es muy elevado. Durante esta investigación también hubo reuniones con el cliente donde se lo fue poniendo al tanto.

Finalmente se optó por xText para el desarrollo del Editor y de ahí en más se comenzó a desarrollar, realizando reuniones cada vez más esporádicas donde se fueron mostrando los avances del desarrollo, y donde se re planificó el alcance inicial de acuerdo a los tiempos que se manejaban para el proyecto.

8 Conclusiones

En los últimos años el manejo de las reglas de negocio a nivel de las organizaciones ha sido foco de atención de los distintos proveedores de software, esto ha generado que ellos brinden soluciones para poder realizar este manejo y eso marca una tendencia en la relevancia que cobra una buena gestión de las reglas. Las ventajas de poder brindar a los Analistas de Negocio la posibilidad de definir sus propias reglas resulta en la aparición de diferentes herramientas que pueden llevar a cabo este objetivo. No obstante no resulta sencillo encontrar una herramienta que nos permita el no involucramiento de usuarios técnicos en el proceso de gestión de las reglas.

En este trabajo hemos podido realizar una investigación en el cual se pudo estudiar el estado del arte referido a las reglas de negocio, se pudo mostrar lo que existe hoy en día a nivel de definición de regla de negocio encontrándose una gran variedad al respecto, de hecho no se encontró una definición estándar. Se proporcionó una definición propia teniendo en cuenta todo lo estudiado y lo mismo sucedió con las definiciones de ciclo de vida. Existe una variedad bastante extensa y dependiendo de los enfoques de cada herramienta, lo que se hizo en este trabajo fue exponer estas variantes a modo de resultado de la investigación. Posteriormente, al igual que en la definición propia de regla de negocio, se proporcionó una definición propia de ciclo de vida, esta está ajustada de forma tal que sea soportada por Guvnor, herramienta que es utilizada por BPS para realizar la gestión de las reglas. Por otro lado se realizó una investigación sobre el ciclo de vida de las reglas de negocio pero a nivel de la organizacional y no de la regla en sí, en esta parte fue muy poco lo que se encontró pero si logramos hallar una investigación en *paper* referido a la temática, esta nos sirvió como base para un planteo de cómo llevar a cabo la incorporación de las reglas en una organización.

Una realidad similar nos encontramos cuando investigamos posibles clasificaciones de reglas de negocio, como se comentaba en el capítulo correspondiente, no existe una clasificación universal de reglas, todas dependen del enfoque y están realizadas por organizaciones o por personas individuales. En función de lo investigado planteamos una definición para clasificación de reglas de negocio propia.

Con estos tres primeros estudios, (el de definición de regla de negocio, definición de ciclo de vida y clasificación de reglas de negocio) nos dimos cuenta de que la gestión de las reglas de negocio llevadas a cabo por software pese a que cada vez cobra más relevancia en las organizaciones, aún no se encontraron soluciones unificadas o estándares que permitan la rápida incorporación del manejo de las mismas. A su vez las herramientas que brindan soporte para la gestión necesitan una previa capacitación para su uso. Estas razones nos hacen pensar que la madurez de la temática avanza pero muy lentamente.

En este proyecto se realizó un estudio del área de negocio referente al Cierre de Obras del área Fiscalización de Atyr basado en lectura de documentos así como en consultas puntuales con él equipo técnico, con el fin de realizar un relevamiento de las reglas de esta área. Esto resultó en un documento que con las reglas de negocio relevadas expresadas en RuleSpeak, expresiones a la postre pueden terminar resultando útiles para expresarlas en el Editor de Reglas de Negocio que se desarrolló. Pudimos ver que esta forma de expresar las reglas utilizando un mismo criterio y cumpliendo las bondades de RuleSpeak, esto parece ser más ordenado que la forma en la que se encontraba en la documentación original.

En cuanto a la investigación realizada referente a las herramientas en las que específicamente se pueden expresar reglas de negocio, pudimos concluir que no hay herramientas disponibles de forma libre que permitan expresar reglas en lenguaje natural. Tampoco nos encontramos con una herramienta en la que fuese posible que el Analista de Negocio sea capaz que expresar las reglas por sí solo, sin la intervención de un técnico. En el Anexo E de tecnologías comentamos acerca de los diferentes BRMS, luego de una investigación nos focalizamos en Guvnor y DRools ya que es código abierto y además ya existía un antecedente de intento de uso en BPS. Surgió la necesidad de contar con una forma de expresar las reglas de negocio en lenguaje natural y como alternativa inicial se manejó la posibilidad de modificar los fuentes de Guvnor para poder agregar una funcionalidad para poder expresar las reglas. Tras varios días de investigación se constató que la documentación del código es casi nula y poder realizar modificaciones tenía altos costos por lo que se buscaron otras alternativas.

XText resultó ser la herramienta que estábamos necesitando. Nos permitió generar una gramática para poder generar tiras en lenguaje natural que cumplan con la plantilla RuleSpeak y desarrollar un Editor de Reglas de Negocio en lenguaje natural que permite expresarlas. A su vez, permite la integración con Guvnor y poder reutilizar todas sus capacidades nativas. Es bueno saber que las reglas una vez exportadas a Guvnor pueden continuar su ciclo de vida dentro del mismo de forma normal.

El único software que se estudió con soporte para el testeo de reglas de negocio de forma unitaria fue Guvnor, como se comentó el Testeo Interno no resultó ser algo amigable para el Analista de Negocio realizar un testeo allí. Por esta razón se proveyó al Editor de herramientas para poder realizar el testeo unitario, a partir de planillas de Excel y datos ingresados por el usuario, ejecutar la regla desde el Editor y corroborar el resultado nuevamente en el Excel.

Por lo tanto, se pudo desarrollar un software que permitiera la edición de reglas de negocio en un lenguaje estándar como RuleSpeak; proveer un ambiente de testing para dichas reglas que los usuarios funcionales puedan validar y brindar los mecanismos para la integración de éstas a la plataforma de JBoss utilizada por BPS. De esta forma, entendemos se cumplieron adecuadamente cada uno de los objetivos planteados al inicio del proyecto por el cliente.

9 Trabajo a futuro

El trabajo a futuro pasa fuertemente en mejoras al Editor de Reglas de Negocio en Lenguaje Natural. Se podría definir el trabajo a futuro en dos tipos, el primero de ellos enfocado a lo que es la parte visual y de usabilidad del Editor, el segundo enfocado en lo referido a la definición de las reglas y al testeo de las mismas.

9.1 Visualización y usabilidad

Comenzaremos marcando algunos puntos que nos han quedado en el tintero de la parte de usabilidad. Considerando que se optó por desarrollar el Editor basado en el Eclipse dado el soporte que encontramos para llevar a cabo nuestra idea, pues entonces hubiese sido bueno haber tenido más tiempo para hacer una remasterización más completa del mismo, con remasterización nos referimos al hecho de ocultar ciertas funcionalidades de Eclipse que pueden llegar a confundir y que no le son útiles al Analista de Negocio.

Otro punto a mejorar y que en cierta forma está incluido en lo comentado anteriormente, pero que por su importancia merece mencionarlo aparte, es el hecho de que el manejo de versiones de los *dsl* escritos por el Analista de Negocio sea manejado con Subversion desde el Editor. Sería bueno brindarle una funcionalidad un tanto más amigable.

Antes de cerrar con los comentarios referentes a la parte visual, comentamos que una de las cosas a mejorar que quedará para futuras versiones y que sería un cambio significativo para el Editor, sería poder implementar una versión Web del mismo.

9.2 Integración con Guvnor

Mirando ahora al Editor desde el punto de vista de una herramienta que será distribuida en varios lugares, será necesario agregar un sistema de autenticación para los usuarios finales. Esto impediría que cualquier usuario pueda modificar las reglas de negocio, más allá de que estas no pasan a producción de forma directa cuando son subidas a Guvnor.

9.3 Gramáticas

Enfocándonos en la gramática, tenemos varios puntos a comentar para trabajo a futuro. Como se mencionaba en el capítulo 4, el Editor de Reglas de Negocio no tiene soporte para las siguientes palabras clave:

- tiene/está autorizado… sólo si…
- no tiene por qué…

Esto en determinado momento del proyecto quedó fuera del alcance por un tema de tiempos y sería bueno agregarlo para futuras versiones.

Entrando en los comentarios referentes a la expresión de las reglas propiamente dicho, sería bueno que no solo se validen que la entidad sea válida o que determinado atributo pertenece a determinada entidad, sino que además se pueda dar un mensaje de error al usuario si este

realiza comparaciones de tipos que no se corresponden entre sí, de forma que no pueda por ejemplo comparar un atributo de tipo Entidad con un atributo de tipo String. En caso de realizar una comparación que no sea correcta se obtendrán resultados inciertos a la hora ejecutar la regla. Por un tema de alcance también se dejó afuera la posibilidad de expresar reglas con condiciones sobre los elementos de una colección, esto sería un agregado de mucho valor que quedará para futuras versiones.

Si bien el editor que desarrollamos es un acercamiento del analista de negocio a las reglas, aun faltaría algo fundamental para su independencia de usuarios técnicos, y eso es la definición sobre el modelo el cual trabajan. Hoy deben trabajar con el modelo que les brindan los técnicos, pero sería necesario que los definan los analistas también. Algo que estudiamos en el desarrollo del proyecto fue incorporar Ecore al editor, Ecore es un producto de eclipse que pertenece al Eclipse Modelling Framework que es perfectamente compatible con la solución que desarrollamos, ya que el mismo xtext utiliza Ecore para las clases que genera a partir de la gramática. Ecore sería la solución para que tanto analistas como técnicos crearan el modelo y trabajaran en conjunto. Es una herramienta en la que crean el modelo utilizando drag and drop y luego se generan las clases java a partir de él, esto podría ser una funcionalidad más del editor.

9.4 Testing

Acerca de la herramienta para realizar el testo de las reglas, sería importante poder corroborar de alguna manera que las reglas de derivación modifican correctamente los elementos de una colección, cosa que hoy no sucede, solo pueden observarse en el resultado de una regla de este tipo los atributos de la entidad principal que se está testeando.

En el testeo unitario solo se ve el impacto de regla que modifican los atributos de la entidad principal.

10 Referencias

[1] - Terry Moriarty - The Next Paradigm, Database Programming and Design.

[2] - Ronald G. Ross - The Business Rule Book (First Edition)

[3] - GUIDE Business Rules Project, Final Report - http://www.businessrulesgroup.org/first_paper/BRG-whatisBR_3ed.pdf, útlima fecha de acceso: 07/09/2013.

[4] - The Business Rules Group, Organizing Business Plans, The Standard Model for Business Rule Motivation, v1.0, 11/2000 - www.BusinessRulesGroup.org - Última fecha de acceso: 07/09/2013.

[5] - Ronald G. Ross and Gladys S. W. Lam - BRS Business Rule Methodology, BRSolutions.

[6] - Morgan T - Business Rules and Information Systems. Aligning IT with Business Goals.

[7] - Are all Rules Business Rules? Not!, Are Integrity Constraints Business Rules? Not!, Are Software Requirements Rules? Not! - http://www.brcommunity.com/b345.php, http://www.brcommunity.com/b335.php, http://www.brcommunity.com/b341.php - Última fecha de acceso: 07/09/2013.

[8] - A Lifecycle Approach towards Business Rules Management: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.123.4732&rep=rep1&type=pdf - Última fecha de acceso: 07/09/2013.

[9] – Primatek - http://www.primatek.ca/en/company.html - Última fecha de acceso: 07/09/2013.

[10] - Clasificación Versata - http://ericwworden.com/WP-VersataBusinessLogicDesignerConcepts-20030106.pdf - Última fecha de acceso: 07/09/2013.

[11] - Barbara von Halle - "Business Rules Applied", Clasificación Chris Date.

[12] - Clasificación Ronald Ross - http://www.brcommunity.com/b086.php - Última fecha de acceso: 07/09/2013.

[13] - Clasificación BRG - http://www.businessrulesgroup.org/first_paper/br01c3.htm#s3b - Última fecha de acceso: 07/09/2013.

[14] - RuleSpeak - http://www.rulespeak.com/es/ - Última fecha de acceso: 07/09/2013.

[15] - Ronald Ross - PROTEUS, Principles of the Business Rule Approach.

[16] - SBVR - http://www.omg.org/spec/SBVR/1.0/ - Última fecha de acceso: 07/09/2013.

[17] - Banco de Previsión Social - http://www.bps.gub.uy/37/institucional.html - Última fecha de acceso: 07/09/2013.

[19] - Architectural Blueprints, The “4+1” View Model of Software Architecture, http://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf - Última fecha de acceso: 07/09/2013.

[20] - BRG - http://www.businessrulesgroup.org/home-brg.shtml - Última fecha de acceso: 07/09/2013.

[21] - Ciclo de vida de WebSphere - http://www.ibm.com/developerworks/websphere/library/techarticles/0810_fasbinder/0810_fasbinder.html - Última fecha de acceso: 07/09/2013.

[22] - Jboss Rules Life Cicle - http://planet.jboss.org/post/business_knowledge_life_cycle - Última fecha de acceso: 07/09/2013.

[23] - Ronald Ross sobre el ciclo de vida de las reglas - http://dataqualitypro.com/data-quality-pro-blog/business-rules-for-data-quality-ronald-g-ross - Última fecha de acceso: 07/09/2013.

[24] - Integrated Analysis and Design of Knowledge Systems and Processes, Mark Nissen, Magadi Kamel, Kishore Sengupta - http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.111.2122&rep=rep1&type=pdf - Última fecha de acceso: 07/09/2013.

[25] - POI - http://poi.apache.org/ - Última fecha de acceso: 07/09/2013.

[26] - Manifesto de reglas de negocio - http://www.businessrulesgroup.org/brmanifesto/BRManifiesto.pdf - Última fecha de acceso: 07/09/2013.

[26] - Microsoft’s Rule Engines - http://karlreinsch.com/2010/02/05/microsoft-rule-engines/ - Última fecha de acceso: 07/09/2013.

[27] - InRule - http://www.inrule.com/InRule_Overview/ - Última fecha de acceso: 07/09/2013.

[28] - Business Rules Engine, Microsoft - http://msdn.microsoft.com/en-us/library/aa561216.aspx - Última fecha de acceso: 07/09/2013.

[29] – BizTalk Server - http://www.microsoft.com/biztalk/en/us/business-rule-framework.aspx - Última fecha de acceso: 07/09/2013.

[30] - OpenRules - http://openrules.com/overview.htm, http://openrules.com/examples.htm - Última fecha de acceso: 07/09/2013.

[32] – IBM iLog - http://www-01.ibm.com/software/websphere/ilog/ - Última fecha de acceso: 07/09/2013.

[33] – Oracle Business Rules - http://www.oracle.com/technetwork/middleware/business-rules/overview/index.html - Última fecha de acceso: 07/09/2013.

[34] – Smart Business Processes using Oracle Business Rules - http://www.oracle.com/technetwork/middleware/ias/smart-processes-obr-1-129729.pdf - Última fecha de acceso: 07/09/2013.

[35] – Oracle Business Rules - http://beatechnologies.wordpress.com/category/oracle-business-rules/ - Última fecha de acceso: 07/09/2013.

[36] – Oracle Business Rules Using Java - http://beatechnologies.wordpress.com/2011/07/07/testing-oracle-business-rules-using-java/ - Última fecha de acceso: 07/09/2013.

[37] - Business Rules Framework Microsoft – http://msdn.microsoft.com/en-us/library/aa561216.aspx - Última fecha de acceso: 07/09/2013.

[38] – Clasificación del Business Rules Group - http://www.businessrulesgroup.org/first_paper/br01c3.htm#s3b - Última fecha de acceso: 09/09/2013.

[39] – Clasificación de Ronald Ross- http://www.brcommunity.com/b086.php - Última fecha de acceso: 09/09/2013.

[40] - Barbara von Halle - Business Rules Applied

[41] – Clasificación de USoft - http://developer.usoft.com/documentation/80doc/index.htm - Última fecha de acceso: 09/09/2013.

[42] – Clasificación de Versata - http://ericwworden.com/WP-VersataBusinessLogicDesignerConcepts-20030106.pdf - Última fecha de acceso: 09/09/2013.

[43] – Clasificación de IBM - http://publib.boulder.ibm.com/infocenter/adiehelp/v5r1m1/index.jsp?topic=%2Fcom.ibm.wasee.doc%2Finfo%2Fee%2Fbrb%2Fconcepts%2Fcbrb_rtypes.html - Última fecha de acceso: 09/09/2013.

[44] – Clasificación de Primatek - http://www.primatek.ca/blog/2008/12/16/on-business-rules-categorization-part-iv-%E2%80%93-rule-classification/ - Última fecha de acceso: 09/09/2013.

[45] - Resumen de las diferentes clasificaciones - http://fox.wikis.com/wc.dll?Wiki~BusinessRuleTaxonomies - Última fecha de acceso: 09/09/2013.

[46] – Clasificación de Oracle - http://docs.oracle.com/cd/E15523_01/integration.1111/e10228/intro.htm - Última fecha de acceso: 09/09/2013.

ANEXO A: Manifiesto de Reglas de Negocio

Manifiesto de Reglas de Negocio

Los Principios de la Independencia de las Reglas

The Business Rules Group[1]

Artículo 1. Los requisitos como elementos principales, nunca como secundarios

1.1. Las reglas son un ciudadano de primera clase en el mundo de los Requisitos.

1.2. Las reglas son esenciales para los modelos de negocio y para los modelos de tecnología, y una parte separada y especifica de los mismos.

Artículo 2. Independientes de los procesos y no contenidas en ellos

2.1. Las reglas son restricciones explicitas de comportamiento y/o proporcionan soporte para la dirección de las actividades de negocio.

2.2. Las reglas no son procesos ni procedimientos. Y por tanto no deben estar contenidas en ninguno de ellos.

2.3. Las reglas se aplican a lo largo de los procesos y procedimientos. Debe existir un corpus coherente de reglas que se aplique sistemáticamente en todas las áreas de actividad del negocio.

Artículo 3. Proporcionar conocimiento meditado, no un sub-producto

3.1. Las reglas se construyen sobre hechos, y los hechos sobre conceptos tal y como son expresados mediante términos.

3.2. Los términos expresan conceptos de negocio; los hechos realizan afirmaciones sobre estos conceptos; las reglas restringen y apoyan estos hechos.

3.3. Las reglas deben ser explicitas. No se debe asumir ninguna regla sobre ningún concepto o hecho.

3.4. Las reglas son los fundamentos que definen lo que el negocio sabe de si mismo- es decir son conocimiento básico de negocio.

3.5. Las reglas necesitan ser alimentadas, protegidas y gestionadas.

Artículo 4. Declarativas, no de procedimiento

4.1. Las reglas deben expresarse de forma declarativa en sentencias de lenguaje natural, por la audiencia conocedora del negocio.

4.2. Si algo no puede ser expresado claramente, entonces no es una Regla.

4.3. Una serie de enunciados solo es declarativa si no contiene una secuencia implícita.

4.4. Cualquier enunciado de reglas que necesite de otros elementos que no sean términos o hechos, revelan hipótesis sobre la implementación de un sistema.

4.5. Una regla es distinta del nivel de cumplimiento definido para ella. La regla y su nivel de cumplimiento son dos asuntos diferentes.

4.6. Las reglas deben definirse independientemente de la quien tiene la responsabilidad de su cumplimiento, y de donde, cuando o como se refuerzan.

4.7. Las excepciones a las reglas se definen mediante otras reglas.

Artículo 5. Expresiones bien formadas, no expresiones creadas con fines específicas para un caso

5.1. Las reglas de negocio se deben expresar de manera que pueda ser validada su exactitud por el personal conocedor del negocio.

5.2. Las reglas de negocio se deben expresar de manera que se pueda verificar recíprocamente su coherencia.

5.3. Las lógicas formales, como la lógica de predicados, son fundamentales para la expresión

formal de reglas en términos de negocio, así como para las tecnologías que implementan dichas reglas.

Artículo 6. Arquitectura basada en las reglas, no una implementación indirecta

6.1. Un sistema basado en reglas de negocio se construye intencionadamente para permitir el cambio continuo de las reglas de negocio. La plataforma sobre la que el sistema se ejecuta debe soportar esta evolución.

6.2. Es mejor ejecutar las reglas directamente – por ejemplo utilizando un motor de reglas – antes que transcribirlas en alguna forma embebida dentro de un procedimiento.

6.3. Un sistema de reglas de negocio siempre debe ser capaz de explicar el razonamiento por el cual llega a una conclusión o emprende una acción.

6.4. Las reglas se basan en los valores ciertos. La forma en la que certeza de una regla se determina, se mantiene oculta a quienes la utilizan.

6.5. La relación entre eventos y reglas es generalmente de muchos-a-muchos.

Artículo 7. Procesos guiados por reglas, no programación basada en excepciones

7.1. Las reglas definen el límite entre actividad de negocio aceptable y no aceptable.

7.2. Las Reglas requieren a menudo de una gestión especial o especifica de las violaciones detectadas. Cualquier actividad derivada de la violación de una regla es una actividad como cualquier otra.

7.3. Para asegurar la máxima consistencia y reutilización, el tratamiento de las actividades de negocio no aceptables, debe separarse de la gestión de actividades de negocio aceptables.

Artículo 8. Al servicio del negocio, no al de la tecnología

8.1. Las Reglas tratan sobre las prácticas de la gestión y gobierno del negocio, por lo tanto son motivadas por las metas y los objetivos de negocio y se les da forma a través de varios factores internos y externos a la empresa.

8.2. Las reglas suponen siempre un coste a la empresa.

8.3. Este coste de la aplicación de las reglas debe valorarse y balancearse, teniendo en cuenta los riesgos asumidos por el negocio, y las oportunidades perdidas en caso de no aplicarlas.

8.4. "Más reglas" no es mejor, la abundancia de reglas no beneficia a su aplicación. Normalmente es mejor un número limitado de reglas bien reflexionadas.

8.5. Un sistema eficaz puede estar basado en un pequeño número de reglas. Adicionalmente, se pueden añadir reglas más discriminatorias, por las que ha medida que pasa el tiempo el sistema mejore y se hace más inteligente.

Artículo 9. "De, por y para" el personal de negocio. No "de, por y para" el personal de IT.

9.1. Las reglas deben provenir del personal con conocimiento de negocio.

9.2. Los expertos de negocio debe tener disponibles herramientas que les ayuden a formular, validar y gestionar reglas.

9.3. Los expertos de negocio deben tener disponibles herramientas que les ayuden a verificar la coherencia reciproca entre las reglas de negocio.

Artículo 10. Gestionando la lógica de negocio, no las plataformas de Hardware/Software

10.1. Las reglas de negocio son un patrimonio vital del negocio.

10.2. A largo plazo, las reglas son más importantes para el negocio que las plataformas Hardware/Software.

10.3. Las reglas de negocio deben organizarse y salvaguardarse de forma que puedan ser re-desplegadas a nuevas plataformas de Hardware/Software.

10.4. Las reglas, y la habilidad para cambiarlas de forma eficaz, son factores clave para mejorar la adaptabilidad de las empresas.

ANEXO B: Plantillas básicas de RuleSpeak

La versión original en inglés ha sido desarrollada por Ronald G. Ross [37]

Definición y propósito

Una plantilla básica es una estructura o patrón básico de sentencia de regla de negocio en lenguaje natural que puede emplearse para expresar un tipo de regla de un modo consistente y bien organizado. Cada plantilla básica está orientada para un tipo de guía/directriz del negocio.

El propósito de las plantillas básicas es asegurar que las reglas escritas pueden comprenderse con facilidad. También ayuda a asegurar que los diferentes profesionales que trabajan sobre un gran conjunto de reglas expresen las mismas ideas, de la misma manera. Esta consistencia no se podría lograr si las reglas se expresaran de forma libre.

Las plantillas básicas no tienen una naturaleza tecnológica. En otras palabras, no representan un lenguaje formal para representar reglas a nivel de implementación, como lógica del negocio basada en tecnología (por ejemplo, motores de reglas). Más bien se concentran en expresar las reglas del negocio para mejorar la comunicación a nivel del negocio.

Factores de éxito en el empleo de las plantillas

¿Qué es importante y qué no?, ¿cuándo aplicamos las plantillas básicas de las sentencias de reglas? Es obvio que no es importante cómo se implementaran las reglas, usando un lenguaje de programación o una tecnología de lógica de negocio automatizada particular. Lo que sí importa es la comunicación efectiva de negocio. A veces, no perder la perspectiva puede ser difícil para aquellos responsables de la implementación técnica. A menudo, suelen tener tendencia a querer obtener las versiones de técnicas directamente. Pero hacer esto no ayuda a capturar las reglas desde la perspectiva de negocio.

El empleo de plantillas básicas para expresar las reglas del negocio requiere de una modesta dosis de disciplina. Esta disciplina se puede conseguir mediante la compresión de ciertos aspectos fundamentos sobre reglas y una relativa moderada práctica. Estos conceptos fundamentos se explican a lo largo del documento.

También es importante comprender a qué nivel de reglas deben aplicarse las plantillas básicas. Las plantillas básicas no están pensadas para el diseño tecnológico de sistema o automatización. Tampoco están pensadas para usarse al nivel más alto de las guías directivas (es decir, el lenguaje de las leyes, regulaciones, contratos, políticas y demás).
Más bien, las plantillas básicas se centran en lo que llamamos Sentencia Practicable. Las reglas del negocio practicables son las que se usan para tomar decisiones de negocio operacionales, en lo que se considera el día-a-día del negocio. Si un trabajador del negocio está adecuadamente autorizado, y comprende el vocabulario de negocio, debería ser capaz de aplicar las reglas de negocio practicables al realizar las actividades continuas del negocio.

A menudo, estas reglas de negocio practicables se interpretan de las formas más elevadas como las directrices directivas, para poder clarificarlas, hacerlas no ambiguas y solidificar la intención de negocio. Las plantillas básicas facilitan esta interpretación proporcionando patrones de expresión predecibles. Lo mismo aplica si las reglas practicables se obtienen mediante ingeniería inversa desde los sistemas existentes, la documentación, o se capturan mediante sesiones de facilitación con expertos en la materia.

Uso de las plantillas

La obligación y la prohibición pueden expresarse con todos sus grados y matices en lenguaje natural de forma afirmativa o negativa. Este abanico de posibilidades es tan amplio y presenta tantos matices y connotaciones que resulta necesario acotar para simplificar y comunicar de forma no ambigua.

- **"Debe"** - Se emplea a menudo el "debe" en las formas más elevadas de guía/directriz de negocio (es decir, el lenguaje de las leyes, regulaciones, contratos, políticas y demás). En RuleSpeak, se prefiere el "tiene que" debido a que el "debe" está contaminado por "debe de". Esto hace que una perífrasis de obligación atenuada "debe + infinitivo" tenga sentido también de probabilidad al estar muy contaminada por "debe de + infinitivo" (perífrasis de probabilidad). Esta ambigüedad es inoportuna cuando se expresan reglas.

- **"Debería**" - Cualquier regla del negocio puede declararse usando "debería" y "no debería" en vez de "tiene que" y acompañando a "está/tiene prohibido" ("debería estar/tener prohibido"). Esta elección refleja una asunción o decisión sobre el nivel de cumplimiento/refuerzo de la regla. En RuleSpeak, se prefiere documentar cualquier sentido de cómo de estrictamente va a ser reforzada de forma separada de la expresión misma de la regla. Por lo tanto siempre se prefiere la palabra clave "tiene que" o "está/tiene prohibido" en vez de "debería" o "debería estar/tener prohibido".

- "**Puede" -** Desafortunadamente, es posible usar de muchas maneras la palabra "puede" en castellano. La ambigüedad resultante es particularmente inoportuna cuando se expresan reglas. Por ejemplo "*Las mujeres pueden ingresar en el Ejército*" tiene hasta 6 interpretaciones distintas:
    - Lo tienen permitido/están autorizadas.
    - Son capaces.
    - Están obligadas.
    - Se las anima a ello.
    - En el futuro, es posible que lo hagan.
    - Advertimos de su posible ingreso (admonición).

  Debido a estos problemas de interpretación, RuleSpeak desaconseja el uso de "puede"/"no puede" en todas las sentencias de reglas del negocio. Por tanto siempre usaremos la palabra clave de sentencia de permiso "está/tiene autorizado".

Conceptos básicos

Comprender algunas ideas fundamentales sobre cómo expresar las reglas del negocio nos ayudará a expresarlas de forma efectiva. Brevemente, las ideas son:

Cada sentencia de regla del negocio debería usar una palabra clave de regla

Cada regla del negocio tiene que ser expresada usando una de las siguientes palabras.

- "tiene que"
- "tiene prohibido" o "está prohibido"
- "ha de ser"

Estas palabras fundamentales son llamadas en RuleSpeak las palabras clave de regla. Es muy importante que cada sentencia de regla incluya exactamente una de estas palabras clave de regla.

Las guías o directrices practicables tienen dos variantes

Una regla del negocio siempre elimina algún grado de libertad. La presencia de una de las palabras clave de regla en una sentencia indica claramente esta intención.

La gran mayoría de sentencias que se expresan para dar directrices o guías son normalmente reglas de negocio. A veces, sin embargo, las guías practicables no eliminan ningún grado de libertad. Este tipo de sentencia, que simplemente clarifica que algo está permitido, o que algo no es obligatorio, se denomina *sentencia de permiso*.

Una sentencia de permiso no debería incluir ninguna de las palabras clave de regla. En vez de esto, debería incluir exactamente una de las dos siguientes palabras clave de permiso.

- "tiene autorizado" o "está autorizado"
- "no tiene por qué"

Las sentencias de permiso y las sentencias de reglas de negocio son ambas: sentencias guía.

Cada sentencia guía tiene una plantilla básica

Una plantilla básica refleja el tipo de guía/directriz que el negocio quiere expresar. Los ejemplos de las sentencias guía se incluirán más adelante en el documento.

Tabla 5. Plantillas básicas de RuleSpeak.

Plantilla básica	Descripción	*Tiene que ser usada para:*
1.0 Una regla del negocio indicando que algo es obligatorio		
... tiene que ...	Indica que algo es obligatorio.	Requiere/obliga a que a ciertos pedidos se les compruebe el crédito.
... ha de ser calculado como ...	Indica que la formula que incluye algunas operaciones matemáticas (por ejemplo: suma, multiplicación, media, etc.) ha de ser usada para calcular algún resultado.	Proporciona una fórmula para calcular el volumen de pedidos anual de un cliente.
... ha de ser entendido como aquel ... que ...	Indica que algo ha de ser clasificado o derivado de cierta forma si ciertas condiciones son verdaderas.	Expresa las circunstancias bajo las cuales un cliente de crédito tiene un riesgo alto.
... tiene que ser ejecutado ... cuando ...	Indica que algún proceso o procedimiento ha de ser ejecutado cuando cierta condición sea verdadera.	Pedir stock cuando la cantidad en el almacén descienda de un determinado nivel.
2.0 Una regla del negocio indicando que algo no está permitido		
... tiene/está prohibido ...	Indica que algo no está permitido.	Obliga a que un producto no sea vendido a un cliente de crédito con riesgo alto.
3.0 Una regla del negocio indicando que algo está permitido condicionalmente		
... tiene/está autorizado ... solo si ...	Indica que algo está permitido solo bajo ciertas condiciones.	Requiere/obliga a que a ciertos pedidos se les compruebe el crédito, pero establece un permiso para otros.
4.0 Una sentencia de permiso indicando que algo está permitido		
... tiene/está autorizado ...	Indica que algo está permitido.	Clarifica que cualquier cliente está autorizado a poseer una cuenta.
5.0 Una sentencia de permiso indicando que algo no es obligatorio		
... no tiene por qué ...	Indica que algo no es obligatorio.	Clarifica que a ciertos pedidos no se les tiene que comprobar el crédito.

Cada sentencia guía debería ser una sentencia completa

El castellano asegura sentencias completas mediante las reglas gramaticales sobre cómo formar las oraciones. Sigamos estas reglas al escribir las sentencias guía.

Cada sentencia guía debería tener un sujeto

Las oraciones lingüísticamente correctas en castellano constan casi siempre de un sujeto. Aunque en ocasiones puede ser un sujeto implícito o el orden de la sentencia puede estar invertido, lo más habitual es que el sujeto aparezca explícitamente como primera palabra o frase en el cuerpo principal de la oración. Las sentencias con un sujeto claro, si están bien escritas, son fáciles de seguir. (Por ejemplo, el sujeto de la oración anterior, es “sentencia”).

RuleSpeak prescribe que las sentencias guía tengan un sujeto al inicio de la oración. Este sujeto debería ser un nombre, a veces cualificado. Además se prefiera el sujeto en singular. Esta aproximación aumenta la claridad y consistencia en general.

Muchas notaciones, a menudo en el ámbito de las tecnologías de lógica de negocio automatizada y en el ámbito de programación – ofrecen la sintaxis “si… entonces…”. En estas notaciones el sujeto no aparece hasta después de la cláusula “si”.

Además de ser “poco amigable” para la comunicación de negocio, esta sintaxis es problemática para expresar ciertos tipos de guía. Por ejemplo: ¿Qué es el “si” en una regla del negocio como “Todo empleado tiene que tener 1 nombre”?

Cualquier sentencia guía debería referenciar a una instancia

Una instancia es una única cosa, en contraposición a una clase de cosas. Las instancias pueden ser referenciadas por las sentencias guía al igual que pueden ser referenciadas las clases de cosas. Si hay dudas de si un término en una sentencia guía se refiere a una instancia o a una clase, la aproximación más prudente es usar las comillas simples con el término.

Ejemplo: La fecha de devolución de declaraciones de renta de las personas físicas en EE.UU. es el 15 de abril.

En este ejemplo, solo hay un “15 de abril” en el calendario, por lo que hay poca probabilidad de confusión. De forma similar, solo hay un país con siglas “EE.UU.” por lo que esta referencia también está clara.

Ejemplo: 1 indicativo de “inflamable” tiene que ser mostrada en cualquier tanque que contenga combustible.

En este ejemplo, “inflamable” se refiere a una instancia de una clase de cosas, en este caso “indicativos”, en vez de a una clase o categoría de pegatinas indicativas (por ejemplo, pegatinas indicativas inflamables). Por lo tanto la palabra “inflamable” se ha incluido con comillas simples.

Cualificando las sentencias guía

La mayoría de sentencias guía incluyen alguna condición indicando las circunstancias bajo las cuales hay que hacer cumplir la regla. A menudo esta condición se expresa siempre mediante una subordinación condicional. La subordinada debería empezar únicamente con "si", evitando en lo posible otras opciones ("cuando", "mientras", "dónde"…) que pueden crear problemas y nunca mejoran la comunicabilidad.

Ejemplo: Todo envío tiene que ser asegurado si el valor del envió es mayor que 500 €.

Algunas cualificaciones se pueden expresar sin usar "si".

Ejemplo: Cualquier empleado retirado tiene prohibido estar asignado a un proyecto.

Aquí la cualificación "retirado" se usa para modificar directamente a "empleado". Esta forma natural de cualificación es perfectamente aceptable.

Ejemplo: Cualquier pedido ordenado por un cliente nacional tiene autorizado el no incluir los gastos de envío en el precio total.

Aquí la cualificación "ordenado por un cliente nacional" modifica de forma natural "pedido", siendo también aceptable.

Guía que se aplica solo a cierto(s) momento(s) en el tiempo

La palabra "si" indica que la cualificación es continúa en el tiempo. La palabra "cuando" debería ser usada si la sentencia guía se aplica sólo en cierto momento temporal.

Ejemplo: El total a pagar correspondiente a la matrícula de 1 estudiante para 1 semestre tiene que ser 5,065 € cuando el estudiante se registre para ese semestre.

Como hemos comentado, la regla del negocio se aplica solo en el momento en el tiempo en que un estudiante se matricula para algún semestre. Asumimos que en cualquier otro momento temporal, la regla no tiene por qué aplicar (¡probablemente se incrementará!).

Permitiendo algo de forma condicional

La plantilla básica "… tiene/está autorizado… si…" (sin el "solo") *debería evitarse*. Las sentencias que usan esta plantilla básica casi siempre tiene truco y se pueden malinterpretar fácilmente.

Ejemplo: Todo producto está autorizado a ser devuelto si se proporciona el ticket de compra.

Esta sentencia expresa un consejo. (Esta interpretación la dicta la lógica.) Cubre la situación en la que tiene que proporcionarse alguna prueba de la compra. Literalmente *no dice* nada acerca de si un producto está autorizado a ser devuelto si *no* se proporciona el ticket. Probablemente, la intención de negocio es no permitir devoluciones sin una prueba de la compra. Asumiendo que este sea el caso, la palabra clave de regla "solo" tiene que usarse tal y como se indica en el siguiente ejemplo, transformando el consejo en una regla del negocio.

Ejemplo: Todo producto está autorizado a ser devuelto solo si se proporciona el ticket de compra.

Las cualificaciones en sentencias de permiso basadas en la plantilla básica "… tiene/está autorizado …" expresadas sin él "si" no son tan fáciles de malinterpretar. Por lo que son aceptables si se usan cuidadosamente.

Ejemplo: Cualquier empleado retirado está autorizado a trabajar a tiempo parcial en Alberta.

Esta sentencia tiene que interpretarse como un consejo. No dice nada sobre empleados no retirados, o sobre empleados que no estén trabajando a tiempo parcial, o empleados que no estén trabajando en Alberta. Si hay alguna otra regla de negocio que necesita cubrir otras circunstancias tiene que ser expresada de forma separada, tal y como ilustra la siguiente sentencia de regla.

Ejemplo: Cualquier empleado retirado tiene prohibido trabajar a tiempo parcial fuera de Alberta.

Cómo decir que algo no es una regla del negocio bajo ciertas circunstancias

Si algo no es una regla del negocio bajo ciertas circunstancias, la mejor plantilla básica para expresar consejo es normalmente "… *no tiene por qué* …".

Ejemplo: La comprobación de crédito de cualquier pedido no tiene por qué ser solicitada si el importe del pedido es menor de 1,000 €.

Observemos que la sentencia de permiso deja abierta la cuestión de si la comprobación de crédito es obligatoria para pedidos de 1,000 € o más. Si se requiere esta comprobación de crédito, se tiene que establecer una regla del negocio por separado. Lo que debemos recordar sobre expresar guías/directrices es que nada es requerido (todo está permitido) a menos que una regla del negocio elimine de forma explícita algún grado de libertad.

Ejemplos de sentencias guía

La tabla 6 proporciona ejemplos para las plantilla básicas de *RuleSpeak*.

Tabla 6. Ejemplos de las plantillas básicas de RuleSpeak

Tipo de guía	Plantilla básica	Ejemplos de sentencias guía
1.0 Una regla del negocio indicando que algo es obligatorio		
1.1.1 Indica que algo es obligatorio de forma incondicional.	*... tiene que ...*	Todo pedido tiene que tener una fecha garantizada de envío. Todo envío tiene que tener 1 estado. El importe pagado por 1 pedido tiene que ser mayor o igual que el importe adeudado por el pedido.
1.1.2 Indica que algo es obligatorio de forma condicional usando el "si".	*... tiene que ... si ...*	Todo envío tiene que ser asegurado si el valor del envío es mayor que 500 €. El impuesto aplicable a ventas para todo pedido tiene que ser 6.25% si el pedido es realizado en Texas y el envío es nacional.
1.1.3 Indica que algo es obligatorio de forma condicional sin usar el "si".	*... tiene que ...*	Todo estudiante tiene que estar inscrito en al menos 2 asignaturas al cierre de inscripción. Toda orden de compra efectuada durante una tormenta de nieve tiene que ser aprobada por 2 o más directores.
1.2 Indica que algo ha de ser calculado de cierta forma.	*... ha de ser calculado como ...*	Coste del producto ha de ser calculado como la suma del coste de todos sus componentes. El importe pagado por 1 pedido ha de ser calculado como la suma de todos los importes de pago relacionados con el pedido.
1.3 Indica que algo ha de ser derivado de cierta forma.	*... ha de ser entendido como aquel ... que/cuyo ...*	Cliente de alto riesgo ha de ser entendido como aquel cliente cuyo saldo a pagar excede los 1000 € en cada una de las tres últimas facturas del cliente. Artículo de precio elevado ha de ser entendido como aquel artículo cuyo coste excede los 500 €. Persona adulta ha de ser entendida como aquella persona cuya edad es 21 años o superior. Profesor asociado ha de ser entendido como aquel profesor que imparte menos de 7 horas lectivas semanales.
1.4 Indica que es obligatorio que sea	*... tiene que ser ejecutado*	'Enviar notificación de envío' tiene que ser ejecutado para 1 pedido cuando dicho pedido se expide.

ejecutado en un momento determinado.	... *cuando* ...	
2.0 Una regla del negocio indicando que algo no está permitido		
2.1 Indica que algo no está permitido de forma incondicional.	... *tiene/está prohibido* ...	El número de asientos para todo sector de la cancha está prohibido que exceda de 30.
2.2 Indica que algo no está permitido usando el "si".	... *tiene/está prohibido* ... *si* ...	Todo pedido tiene prohibido ser expedido si el saldo a pagar de la cuenta que posee el cliente que realiza el pedido supera la autorización de crédito del cliente.
2.3 Indica que algo no está permitido sin usar el "si".	... *tiene/está prohibido* ...	Cualquier pedido a crédito de más de 1,000 € tiene prohibido ser aceptado sin una autorización de crédito. Todo cliente de alto riesgo tiene prohibido realizar cualquier pedido de cualquier artículo de precio elevado. Toda sustitución de cualquier miembro del equipo de proyecto está prohibida hasta que el proyecto no ha terminado.
3.0 Una regla del negocio indicando que algo está permitido condicionalmente		
3.1 Indica que algo está permitido solo bajo ciertas condiciones usando "si".	... *tiene/está autorizado* ... *sólo si* ...	Todo cliente está autorizado a adquirir cualquier pesticida de cualquier proveedor sólo si el proveedor comercializa ese pesticida. Todo cliente tiene autorizado realizar cualquier pedido sólo si posee cuenta de crédito. Cualquier retirada de dinero de cualquier cuenta está autorizada sólo si la cuenta está activa.
3.2 Indica que algo está permitido solo bajo ciertas condiciones sin usar "si".	... *tiene/está autorizado* ... *sólo* ...	El uso de la alarma está autorizado sólo en situación de emergencia.
4.0 Una sentencia de permiso indicando que algo está permitido		
4.1 Indica que algo está permitido de forma incondicional.	... *tiene/está autorizado* ...	Cualquier persona de cualquier edad tiene autorizado el mantenimiento de cualquier cuenta.
4.2 Indica que algo está permitido solo bajo ciertas condiciones sin usar "si".	... *tiene/está autorizado* ...	Cualquier empleado retirado está autorizado a trabajar a tiempo parcial en Alberta. Cualquier pedido a crédito de 1,000 € o menos tiene autorizado ser aceptado sin autorización de crédito.
5.0 Una sentencia de permiso indicando que algo no es obligatorio		

5.1 Indica que algo no es obligatorio, de forma incondicional.	*... no tiene por qué ...*	El cliente no tiene por qué realizar ningún pedido.
5.2 Indica que algo no es obligatorio bajo ciertas condiciones usando "si".	*... no tiene por qué ... si ...*	La comprobación de crédito no tiene por qué ser solicitada si el pedido es de menos de 1,000 €.
5.3 Indica que algo no es obligatorio bajo ciertas condiciones sin usar "si".	*... no tiene por qué ...*	El cliente preferente no tiene por qué pagar en metálico para cualquier pedido con destino internacional.

ANEXO C: Ciclo de vida de la Regla de Negocio

En la sección **¡Error! No se encuentra el origen de la referencia.** hablábamos acerca del ciclo de vida de las reglas de negocio en sí y mostrábamos en la figura 2 el ciclo de vida planteado por Primatek.

En este anexo mostraremos cómo este ciclo de vida puede ser implementado utilizando la herramienta iLog de IBM. Se puede recurrir al Anexo E para informarse más con respecto a iLog.

ILog

ILog [32] cuenta con tres grandes módulos, ellos son:

- Business rule applications development
- Business rule management and authoring
- Enterprise application

Nos focalizaremos en los dos primeros que son los módulos en donde es posible manipular reglas de negocio.

En el módulo *Business rule applications development* trabajan arquitectos y el equipo de desarrollo. Tanto desarrolladores como analistas de negocio pueden participan en la definición de vocabulario específico. Se implementan aquí reglas complejas que no son posibles de expresar por un analista de negocio.

En el módulo *Business rule management and authoring* se provee tanto a los Policy Managers como a los analistas de negocio las herramientas necesarias para expresar reglas. El Policy Manager es entendido como un experto en la lógica de negocio encargado de definir y de hacer cumplir las políticas a nivel empresarial.

Nos centraremos ahora en este último módulo, el *Business rule management and authoring*, se presentará entonces una tabla por estados en donde se muestran los actores y una breve descripción de cómo se podría realizar la implementación, esto en base a las herramientas provistas iLog.

Los actores que entran en juego son:
Policy Manager: Explicado en párrafos precedentes cuál es el rol de este actor.
Analista de negocio: Traducen las políticas de negocio a una especificación aceptable para desarrolladores, realiza validaciones de la especificación formal con los Policy Manager
Administrador: Se encarga de que el entorno ejecute normalmente.
Desarrollador: Los desarrolladores están encargados de desarrollar, testear, debuggear y desplegar el aplicativo. Crean e implementan vocabulario para utilizar en las reglas de negocio. Crean e implementan reglas de negocio complejas que no es posible que sean expresadas por los usuarios.

Estado	Actor	Descripción
New	Policy Manger, Analista de negocio, Desarrollador.	Los Policy Managers crean y editan las reglas de negocios ayudados por los analistas de negocio y desarrolladores.
Ready for review	Policy Manager, Analista de negocio.	Crean snapshots utilizando líneas base de las reglas en las que ellos estaban trabajando antes de que sean desplegadas, se escriben consultas y se crean reportes utilizando los resultados de las mismas. Se puede realizar una revisión de reglas definidas por otros. Es una instancia para que se verifique que las reglas no violen ninguna política de negocio. Se asocia con la actividad "Review" del iLog.
Reviewed	Policy Manager	Se ha pasado por la etapa anterior de forma exitosa.
Ready to test	Policy Manager	Utilizando el *Decision Validation Services* es posible validar que las políticas de negocio fueron bien definidas mediante la utilización de escenarios. Se pueden crear suites de testeo y ejecutar pruebas dentro de ese entorno para detectar posibles errores provenientes de la regla de negocio. Se asocia con la actividad "Validate" del iLog.
Rejected	Policy Manager, Analista de negocio	Determinada regla ha sido rechazada en el estado anterior.
Tested	Policy Manager	Se han pasado los test ejecutados anteriormente.
Ready to deploy	Administrador	La regla está en manos del administrador donde se encarga de instalarla y dar permisos a los usuarios para poder manipularla. Se asocia con la actividad "Deploy" del iLog.
Deployed	Policy Manager, Analista de negocio	La regla se encuentra en ejecución.

Tabla 7: Posible implementación del ciclo de Primatek.

Ronald Ross

Para Ronald Ross [23] el ciclo de vida posee las siguientes fases:

1) Estudiar las actividades de la empresa
2) Capturar los datos
3) Analizar los datos
4) Desplegar los cambios realizados y volver a la actividad de la empresa

Muchos profesionales en sistemas de información solo se concentran en las fases de captura y análisis, pero el valor de real se encuentra en el ciclo de vida como un todo. Como siempre se ha sugerido, ¿de qué sirve tener los datos bien discernidos si luego no se puede actuar fácilmente en consecuencia? Con actuar en consecuencia nos referimos a desplegar reglas de negocio en las operaciones del día a día.

De esta forma es aquí donde muchos profesionales focalizados en los proceso de negocios pierden el tren, ya que no entienden que si realmente se necesita agilidad en el negocio es necesario una alta calidad en los datos y una gestión de decisiones eficaz.

JBoss

Drools [22] es un sistema experto para la gestión de reglas de negocio, se le llama sistema experto porque intenta imitar las decisiones o acciones tomadas por un experto humano en un dominio específico. En vez de orientado a las reglas, Drools está orientado al conocimiento ya que trata de derivar una respuesta basándose en una base de conocimientos. Dentro de la documentación encontrada de Drools, se considera el ciclo de vida del conocimiento del negocio el cual se puede ver en la figura 46.

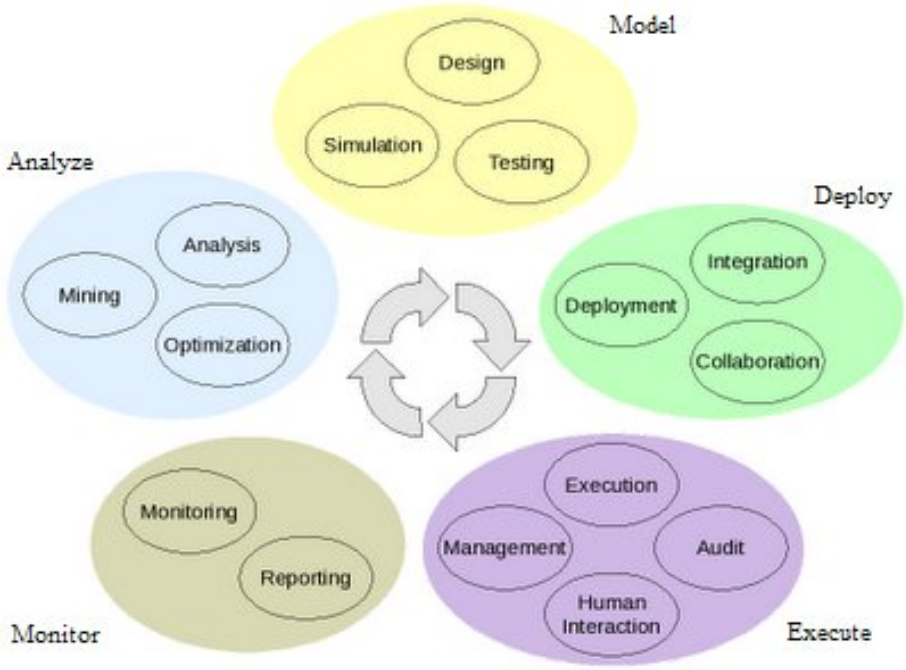

Figura 46: Ciclo de vida de las reglas en Jboss

MODEL (Modelado):

Design (Diseño): El actor crea o actualiza el proceso de negocio en alguna interfaz gráfica, esto incluye dibujar el control del flujo, definiendo propiedades para cada nodo del proceso, hay herramientas especializadas según el actor que desarrolle esta tarea, puede ser un analista de negocio o un desarrollador.

Testing: Una vez que el proceso está definido se deben testear escenarios específicos del proceso y guardar ese conjunto de test para repetirlos a lo largo del ciclo. Esto incluye

debuggear para ver qué ocurre durante la ejecución paso a paso e ir inspeccionando los estados.

Simulation (Simulación): Se simulan completamente casos de uso específicos, es una extensión al testing. Se debe tener una visualización amigable para todo tipo de usuarios, se debe tener la posibilidad de disparar eventos, controlar el flujo del tiempo, realizar validaciones, etc.

DEPLOY:

Integration (Integración): Los procesos definidos deben ser integrados a las aplicaciones, esta integración debe mantener la integridad del sistema y no debe dificultarse el mantenimiento y actualizaciones futuras. Se recomienda utilizar un repositorio de conocimiento para guardar todos los procesos definidos, la herramienta a utilizar debe simplificar la integración de los procesos guardados con el ambiente de programación. Drools tiene como repositorio de conocimiento a Guvnor.

Collaboration (Colaboración): Se debe incorporar también los procesos definidos al software de colaboración, Guvnor también brinda soporte a esto. El software de colaboración hace referencia a los métodos y herramientas de software que facilitan el trabajo en grupo y mejoran su rendimiento, incluye herramientas de comunicación electrónica que envían mensajes, archivos, datos o documentos entre personas, y herramientas de gestión colaborativa que facilitan las actividades del grupo, como por ejemplo calendarios electrónicos.

Deployment (Despliegue): Luego de tener todo integrado se realizan las actividades, configuraciones y compilaciones para dejar el sistema listo para ser instalado y usado.

EXECUTE (Ejecutar):

Execution (Ejecución): la fase de ejecución implica la aplicación real de los procesos de negocio. En esta etapa todos los elementos del proceso, como actividades automáticas e interacción humana, están integrados y conectados para que puedan interactuar y comunicarse y así lograr los objetivos del negocio.

Audit (Auditoría): El sistema de la organización puede tener muchos procesos activos y en diferentes etapas de ejecución, por lo que es esencial tener un registro de auditoria para encontrar pedidos o información de la ejecución. Se debe guardar esta información de auditoria, incluso en algunas organizaciones puede ser un requisito legal, y también brindar herramientas para la búsqueda y eliminación de registros de auditoria.

Human Interaction (Interacción humana): La interacción humana es muy importante en los sistemas de la organización, algunos procesos se ejecutaran automáticamente pero otros pueden necesitar de actores humanos. Se le asigna tareas a ciertos usuarios, los usuarios pueden ejecutarlas o re asignarlas, se debería contar con una herramienta para armar listas de usuarios, permitir las asignaciones, además de tener alguna herramienta para el testeo y demos.

Management (Administración): se debe proveer una interfaz que permita la administración de los procesos en ejecución, solicitar las tareas asignadas a cierto usuario, realizar tareas de auditoria, es decir administrar todas las fases de esta etapa del ciclo de vida.

MONITOR (Monitoreo):

Monitoring (Seguimiento): se monitorea lo que ocurre en los procesos durante la ejecución, se controlan distinto tipo de eventos como por ejemplo instancias de un proceso que inician o que son completados. Estos eventos se pueden observar dinámicamente o pueden ser utilizados para crear logs y guardarlos en la base de datos para posteriormente analizarlos.

Reporting (Reportes): Si se guardan registros históricos de los eventos ocurridos en los procesos en ejecución, con esa información se puede presentar informes. Esos reportes pueden ser usados para monitorear o realizar análisis de ejecución, se pueden encontrar problemas en el proceso o de performance del sistema.

ANALYZE (Analizar):

Analysis (Análisis): con herramientas y técnicas de diagnóstico a partir de los reportes de la fase anterior se controlan los procesos ejecutados en búsqueda de problemas, ya sea de performance como cuellos de botella o de reglas ejecutadas incorrectamente.

Mining (Extracción): Se estudia y extrae cuál es el problema en la ejecución del proceso, puede ser una regla mal definida o alguna variante de una tarea ejecutada por un humano. Si hay problemas en el sistema de performance se busca donde se produce.

Optimization (Optimización): se plantea o planea el rediseño de la regla o el proceso que haya presentado problemas en el estudio.

Desde el punto de vista del usuario no hay diferencia entre las etapas, no se distingue para describir el conocimiento del negocio si el sistema esta implementado solo con reglas de negocio o reglas basadas en eventos o incluso CEP. El uso de Drools brinda una sola herramienta para el manejo de todas estas etapas unificándolas y brindando la oportunidad de elegir que paradigma es más conveniente para especificar la base de conocimiento de su negocio.

Business Rules Framework de Microsoft

Business Rules Framework [37] forma parte de BizTalk Server, en la documentación encontrada se hace referencia al ciclo de vida más general a nivel de gestión de las reglas de negocio en su conjunto. Se pueden ver los siguientes estados correspondientes a la gestión de las reglas de negocio propuesto por Microsoft.

- *Incorporación:* Planificar cómo incorporar las reglas de negocio en la aplicación.

- *Identificación de lógica de negocio*: Identificar la lógica de negocio que se quiere representar utilizando reglas de negocio en la aplicación.
- *Identificación de fuentes de datos*: Se identifican las fuentes de datos de los elementos de las reglas. Si se desea se puede definir vocabulario específico del dominio.
- *Definición de reglas*: Se definen las reglas desde el vocabulario definido o directamente desde las fuentes de datos.
- *Testeo:* Testear y debuggear las reglas con datos de ejemplo. Se proveen distintos mecanismos que permiten la generación del ambiente para poder realizar el testeo.
- *Publicación:* Publicar la versión de la regla en el almacenamiento de las reglas.
- *Despliegue*: Realizar el despliegue de la regla.
- *Monitoreo*: Realizar el monitoreo de las regla de negocio cuando sea necesario.

WebSphere de IBM

El producto WebSphere [21] de IBM, dentro de los servicios que ofrece se encuentra el de modelado de procesos de negocio, y dentro de este se da soporte al manejo de reglas de negocio, mostraremos a continuación como maneja este producto el ciclo de vida de las reglas.

El ciclo de vida lo define para permitir llevar a cabo el manejo de la sincronización de las reglas de negocio, es decir, como las reglas de negocio son cambiantes a lo largo del tiempo y a su vez estas tienen que impactar el sistema en tiempo real, ven necesario definir el ciclo de vida para poder llevar a cabo esto.

Se manejan cinco estados para las reglas de negocio residentes en WebSphere, ellos son:

- *Estado inicial*: estado en el que se permite la definición de la regla mediante el Business Rules Manager, ésta es almacenada en un archivo zip y luego integrada al proceso de negocio utilizando el WebSphere Integration Developer.
- *Testeo de la regla*: permite en este estado realizar un testeo de la regla una vez incluida en el proceso utilizando parámetros y datos de prueba.
- *Actualizadas con el manejador de reglas de negocio*: una vez que las reglas de negocio van cambiando, Business Rules Manager permite impactar estas modificaciones en las reglas definidas.
- *Exportadas con las modificaciones*: una vez que sea han hecho las modificaciones de la regla utilizando el Business Rules Manager ésta se exporta a un archivo zip.
- *Sincronizada con el proceso de negocio*: es posible importar el archivo zip utilizando nuevamente el WebSphere Integration Developer al proceso para que la regla sea sincronizada al mismo.

Se puede profundizar más sobre cada producto en el anexo E.

ANEXO D: Clasificación de Reglas de negocio

Existen diferentes clasificaciones según para quién se dirija la clasificación, en este anexo se profundizaran las clasificaciones para los analistas de negocio y para los técnicos informáticos.

Clasificación para analistas de negocio

Clasificación del Business Rules Group

A continuación se resumen las definiciones de los conceptos relativos a Reglas de Negocio propuestos por el Business Rules Group [38].

Regla de Negocio (Business Rule)	Declaración que define o restringe algún aspecto del negocio. Esto debe ser un término o un hecho (descrito más adelante como una afirmación estructural), una restricción (descrito más adelante como una afirmación de acción), o una derivación. Es atómica en el sentido de que no se puede descomponer en reglas más detalladas, si se redujera se perdería información importante del negocio.
Declaración de Regla de negocio (Business Rule Statement)	Afirmaciones declarativas de la estructura del negocio o restricciones que el negocio impone sobre sí mismo. Puede ser la fuente de varias reglas de negocio, es decir, varias reglas de negocio pueden ser basadas en la misma declaración de negocio.
Policy (Política)	Es la base para una o más declaraciones de reglas. Es información que manifiesta la dirección que tiene la organización.

Tabla 8: Conceptos definidos por el BRG.

Las categorías de las reglas de negocio corresponden a las categorías de las declaraciones en las cuales las reglas se basan, el BRG define cuatro categorías:

• ***Definiciones de términos del negocio***
Es el elemento más básico del lenguaje de las reglas de negocio, define cómo la gente piensa y habla. Definir los términos es establecer una categoría de regla de negocio.
En general los términos son definidos en glosarios o como entidades de la organización en los modelos de relaciones.

• ***Hechos que relacionan términos del negocio entre sí***
La estructura de funcionamiento de una organización puede describirse a través de los hechos que relacionan a los términos entre sí. Los hechos pueden ser documentados como frases en lenguaje natural o como relaciones, atributos, y estructuras de generalización de un modelo.

• ***Restricciones*** (ellos la llaman 'declaraciones de acción')
Cada organización restringe comportamientos, esta clasificación se relaciona a restricciones para los datos, restringir cuando un dato puede ser actualizado, o prevenir que ocurra cierta acción.

• ***Derivaciones***
Las reglas de negocio definen como el conocimiento se transforma en algo nuevo.

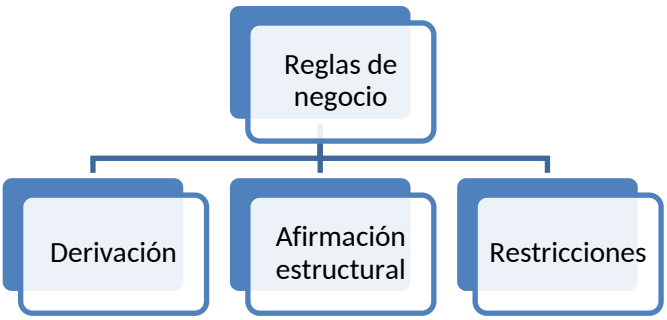

Figura 47: Tipos de reglas según el BRG.

Afirmación estructural (STRUCTURAL ASSERTION)

Es una declaración de algo de importancia para el negocio, puede ser algo que exista como concepto, o una relación con otro interés para el mismo. Detalla específicamente un aspecto estático del negocio. Hay dos tipos de afirmaciones estructurales; los *términos* y los *hechos*.
Un término es una palabra o frase que tiene un significado especial para el negocio, los términos de interés pueden ser términos comunes o términos de negocio. Un término de negocio es aquella palabra o frase que tiene un significado especial para el negocio dentro de cierto contexto del mismo, cada termino de negocio definido debe ser usado al menos una vez en algún contexto del negocio. Ejemplos de términos de negocio en el contexto de una organización que alquila autos podrían ser 'reserva', 'pedido de alquiler' o 'devolución'. Términos comunes son palabras que se usan en el día a día y tienen un significado universalmente conocido y aceptado, por ejemplo 'auto' o 'ciudad'. Los términos pueden ser definidos como un tipo, como es el caso de 'cliente' o 'modelo de auto', pueden ser literales como 'General Motors', o pueden definir relaciones de sinónimos entre dos términos como pueden ser 'auto' y 'vehículo'.

Los términos son usados para construir hechos de la organización, los hechos son una asociación de uno o más términos, un término puede aparecer en más de un hecho.
Por ejemplo un hecho sería "cada CLIENTE puede tener muchos CONTRATOS", siendo 'cliente' y 'contratos' términos previamente definidos. Los hechos son reglas simples como pueden ser asignar un término como atributo de otro o definir relaciones entre términos, relaciones de participación o de generalización. Algunos ejemplos serían:

- Nombre es un atributo de cliente
- Color es un atributo de auto
- Un administrador es un empleado
- El inventario de un modelo de auto está compuesto de los autos de ese modelo que pertenecen a la empresa

- Los autos son alquilados por clientes
- Los clientes pueden pedir determinado tipo de auto

Otros tipos de hechos son definidos como reglas derivadas que serán desarrollados más adelante.

Restricciones (ACTION ASSERTION)

Es una declaración que restringe o una condición que limita o controla las acciones de la organización, representa los aspectos dinámicos del negocio. Especifica los resultados que las acciones pueden producir, mientras que las afirmaciones estructurales describen posibilidades, estas afirmaciones de acción restringen, se utilizan palabras como 'debe' o 'tiene que' en vez de 'debería' o 'podría'. Ejemplos de esta categoría serían:

- Un auto debe tener un número de registro.
- Un auto no puede ser entregado al cliente si no se aprobó el pedido en la tarjeta de crédito del cliente.

Las acciones pueden ser desde la creación de una instancia hasta la ejecución de un método definido en la acción. También se pueden definir acciones que dependan de una variable del tiempo como un timer. Por ejemplo "si un cliente tiene tres meses atrasados, entonces se debe tomar posesión del auto alquilado", agrega una restricción de tiempo (los tres meses) en la condición y se ejecuta la acción de adquirir el auto.

Dentro de las declaraciones de acción se las puede separar en tres categorías:

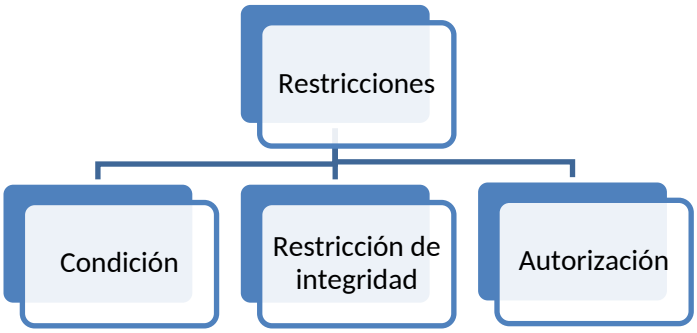

Figura 48: Categorías de las restricciones.

Una *condición* (CONDITION) es una afirmación de que si algo es verdadero se aplica una regla del negocio, es decir otra afirmación de acción. Ejemplos de condiciones podrían ser preguntas como "¿ha mostrado el cliente una licencia de conducir vigente?" o "¿el cliente ha realizado una orden de compra?" y si se cumple se aplica otra regla de negocio que puede ser de su tipo u otro tipo.

Una *restricción de integridad* (INTEGRITY CONSTRAINT) es una afirmación que siempre debe ser verdadera. Se considera una regla con poder de ejecución inmediata, ya que prohíbe cualquier acción que implique un valor falso. Mientras que las reglas del tipo *condición* testea un valor y decide que acción tomar a partir del resultado de ese test, las reglas de *restricción*

de integridad declaran y prohíben cualquier acción si es que se viola la condición. Por ejemplo una *condición* preguntaría "hay un auto registrado?", mientras que una *restricción* declara "un auto debe estar registrado" y si no hubiera un auto registrado prohíbe las acciones ya sea dar de alta un registro o setear el estado del registro del auto en null.

Una *autorización* (AUTHORIZATION) define una libertad específica o permisos sobre cierta regla o cierta acción. Se especifica 'solo X puede hacer Y', siendo X un usuario e Y una acción. Estos permisos se otorgan a 'tipos' independientes en una cierta actividad (por ejemplo usuarios, sectores de la organización, computadoras, etc). Un ejemplo de este tipo de reglas sería "solo el manager de la empresa puede autorizar una devolución del producto".

Derivación (DERIVATION)

Una derivación es una declaración de conocimiento que es resultado de la derivación de otro conocimiento del negocio. Puede ser de dos tipos, un *cálculo matemático* (mathematical calculation) por ejemplo "el total es el costo multiplicado por las horas", o una *inferencia* (INFERENCE) como por ejemplo si una persona ya tiene una deuda y realiza una nueva compra, el total adeudado es lo que debía antes más el costo de lo que compró ahora.

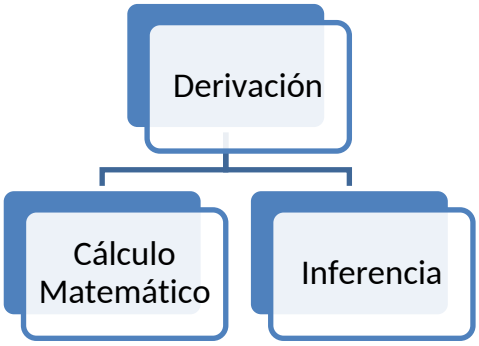

Figura 49: Categorías de la derivación.

Un *cálculo matemático* produce conocimiento de acuerdo a algún algoritmo matemático, mientras que una *inferencia* produce conocimiento usando lógica inductiva o deducciones.

Retomando lo que eran las categorías de las reglas, relacionándolas con el tipo de reglas quedaría:

Categoría	**Tipo de regla**
Definiciones de términos del negocio	Afirmación estructural
Hechos que relacionan términos del negocio entre sí	Afirmación estructural
Restricciones	Acciones
Derivaciones	Derivaciones

Se muestra a continuación En la figura 50 un esquema general de esta clasificación.

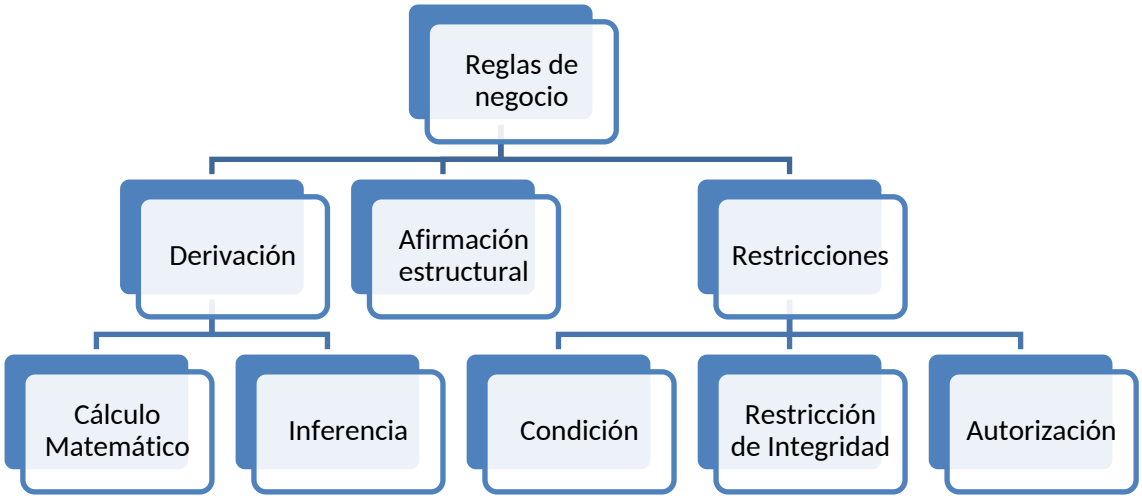

Figura 50: Clasificación Business Rules Group.

Esquema de clasificación de Ronald G. Ross

La clasificación de Ronald Ross llamada BRS (Rule Classification Scheme [39]) tiene las siguientes tres categorías principales:

1) Rechazo (Restricción)
2) Productor
    a. Reglas de cálculo
    b. Reglas de derivación (Reglas de inferencia)
3) Disparador (Estimulación / Respuesta de reglas)
    a. Habilitación
    b. Copia
    c. Ejecución

A continuación se profundizará esta clasificación explicando más a fondo cada una de estas categorías. Se hará una presentación de forma compacta utilizando ejemplos.

El BRS refleja como las reglas reaccionan a diferentes eventos. Una regla dada puede reaccionar de tres formas posibles ante un evento, por lo tanto, hay tres categorías fundamentales de reglas en este esquema.

Estas tres categorías que se desarrollan a continuación, son esenciales, definitivas y mutuamente excluyentes. También hay subcategorías como la tabla lo indica. Las categorías proporcionan una base rica para la organización de las y a su vez permiten obtener una mejor comprensión de las reglas de la empresa.

Describiremos concretamente las categorías y subcategorías.

Categoría Rechazos.

Nombre común: Restricción.

Estas son las reglas que no permiten que ocurra un evento si se violan las condiciones de la misma.
Rechazadas son las restricciones que protegen al negocio de datos incorrectos, es decir, partiendo de la información que hace que se viole la regla de negocio.

Por ejemplo, un *Rechazo* podría ser especificado para impedir que un cliente realice una solicitud de crédito si este no tiene un buen historial de pagos.

Ejemplo de restos tipos de reglas serían:

- Una orden de compra tomadas durante una tormenta de nieve tiene que ser aprobada por al menos dos directores.

- El retiro de una cuenta está autorizado solo si la cuenta está activa.

- Un pedido tiene prohibido ser entregado si el saldo de la cuenta del cliente excede la autorización de crédito del cliente.

Categoría Productor.

Son las reglas que no rechazan ni proyectan eventos sino que simplemente calculan, miden, derivan o infieren de forma automática. Las reglas *Productor* pueden ser categorizadas como *Reglas de cálculo* y *Reglas de derivación (Reglas de inferencia)*.

Reglas de cálculo

Esta sub categoría incluye cualquier tipo de regla que compute valores de forma automática, siguiendo las operaciones matemáticas estándares especificadas de forma explícita (suma, multiplicación, promedio,…)

Por ejemplo, una regla de cálculo puede estar planteada para calcular el “volumen de pedidos anuales” de un cliente.

Otro ejemplo sería: El costo de un producto *ha de ser calculado* como la suma del costo de todos sus componentes.

Reglas de derivación (Reglas de inferencia)

En esta sub categoría se incluyen las reglas que se infieren automáticamente basadas en condiciones especificadas de forma explícita.

Por ejemplo, una regla de derivación se puede dar para expresar si una persona es considerada una mujer, basada en la edad y el sexo.

Un ejemplo concreto sería: Un cliente de alto riesgo *ha de ser entendido como aquel que* su saldo supera los $1000 en cada una de sus últimas tres facturas.

Categoría Disparador (Estimulación / Respuesta de reglas)

Esta categoría abarca las reglas que tienden a tomar alguna acción automáticamente cuando un evento relevante ocurre. Las acciones podrían ser crear o eliminar datos, habilitar o deshabilitar una regla, setear un valor, ejecutar un procedimiento o programa. Un *Disparador* no rechaza los eventos (como lo hace un *Rechazo*) sino que, causa algunos nuevos eventos como resultados de estos. Los *Disparadores* en general describen el comportamiento automático del sistema, proporcionando un aumento de la productividad de los trabajadores.

Un *Disparador* puede ser especificado para reordenar el stock de forma automática si la cantidad disponible cae por debajo de cierto punto.

Habilitación

Estas son cualquieras de las reglas *Disparador* que tiene uno de los siguientes efectos, crear o eliminar una instancia de datos, permitir o no permitir la ejecución de un proceso o procedimiento.

Algunos ejemplos de este tipo de reglas son:

- Los asuntos pendientes tienen que ser eliminados cuando un caso está cerrado.

- La regla “una entrada un asiento” en el Centenario tiene prohibido ser forzada si el dueño del asiento es Alcides Edgardo Ghiggia.

- La regla “Enviar aviso de cita” tiene que ser deshabilitada si no se conoce la dirección del cliente.

Copia

Estas son cualquier tipo de reglas que setea datos. Este tipo de datos puede ser complejo y no necesariamente tiene que ser almacenado. Por ejemplo, una *Copia* se puede utilizar para inicializar la matrícula a un estudiante al principio de semestre cuando el estudiante se matricula. Otros ejemplos son:

- Reglas de inicialización, se refieren al “almacenamiento” de datos
  Regla: El impuesto para las ventas tiene que ser seteado en ‘8.25%’ si la fecha de cumplimiento de la orden es del 2011.

- Reglas de presentación, describe la “salida” de datos
  Regla: Una orden tiene que ser mostrada en rojo en la pantalla si la orden está atrasada.

Ejecución

Son los tipos de regla que causan que un proceso o un procedimiento sea ejecutado, o que una regla sea disparada. Por ejemplo:

- El procedimiento *enviar-nota-de-avance* tiene que ser ejecutado por una orden, cuando la orden es enviada.

- La regla *proyección-de-fecha-envío* debe ser disparada, cuando un envío es desplegado en la pantalla.

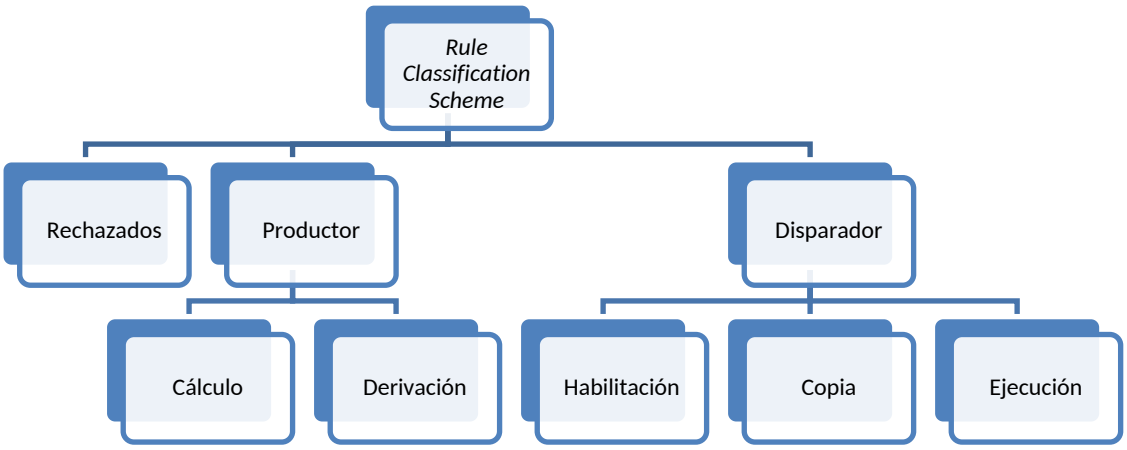

Figura 51: Clasificación Ronald Ross.

Clasificación de Reglas de Barbara Von Halle

Esta clasificación [40] está dirigida hacia los analistas de negocio, con esto en mente, el esquema de clasificación sirve a los siguientes fines:

- Explica toda la gama de reglas de negocio que se está buscando para relevar.
- Simplifica y guía a los analistas del negocio de manera eficiente a través de una secuencia para descubrir reglas.
- Permite a los analistas expresar cada tipo de regla de negocio en su propio vocabulario para una mayor claridad.

La clasificación divide al mundo de las reglas de negocio en sólo tres grandes categorías: los términos, hechos y reglas.

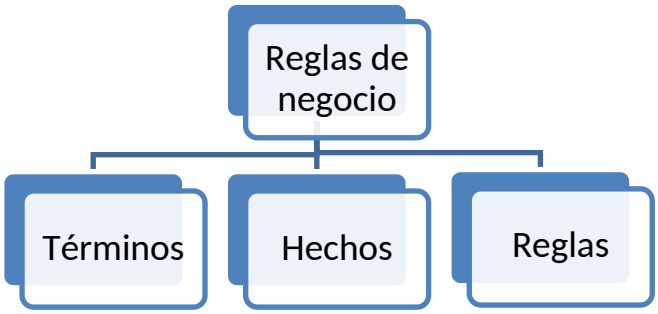

Figura 52: Clasificación de Barbara Von Halle.

Términos (Terms)

Un término es una palabra o frase que tiene un significado específico para el negocio en algún contexto particular, su significado para el negocio debe ser definido. Los términos son referenciados en los otros tipos de reglas y pueden definir lo siguiente:

- Un concepto como “cliente”.
- Una propiedad de un concepto como “código de cliente”.
- Un valor como “femenino”.
- Un conjunto de valores como “días laborables (lunes, martes, miércoles, jueves, viernes)”.

Hechos (Facts)

Un hecho es una afirmación que asocia uno o más términos expresando algún tipo de relación entre ellos. Un término puede aparecer en varios hechos, algunos ejemplos de hechos son:

- Un cliente puede realizar una orden.
- La orden es para un ítem de la línea.
- Un ítem de la línea es un producto.
- Un cliente tiene un código de cliente.

Los términos y hechos son la semántica atrás de las reglas y son la base de la lógica de las mismas.

Reglas (Rules)

Es la tercera parte de la clasificación, es una afirmación declarativa que aplica lógica o computaciones a valores de información. Los resultados de una regla generan nueva información o una acción resultante de la decisión de la regla. Ejemplos de reglas son:

- La cantidad total en dólares es la suma de la cantidad en dólares de cada ítem.
- Un nuevo cliente no puede realizar una compra cuyo total sea mayor a $1000.
- El envío de un pedido de un cliente que no ha pagado su última factura, se retrasará hasta que se reciba el pago.

Cuando un evento de negocio o de casos de uso se lleva a cabo, una persona del negocio puede inyectar reglas para controlar su ejecución y hay al menos cuatro formas posibles en que una regla puede guiar a un evento de negocios. Una regla en el contexto del evento de negocio puede:

- Presentar la información sobre el evento de negocio.
- Restringir la información generada por el evento de negocio.
- Iniciar una acción fuera de los límites del sistema o del evento de negocio.
- Crear nueva información a partir de información existente.

Este tipo de reglas se puede desglosar en la siguiente clasificación:

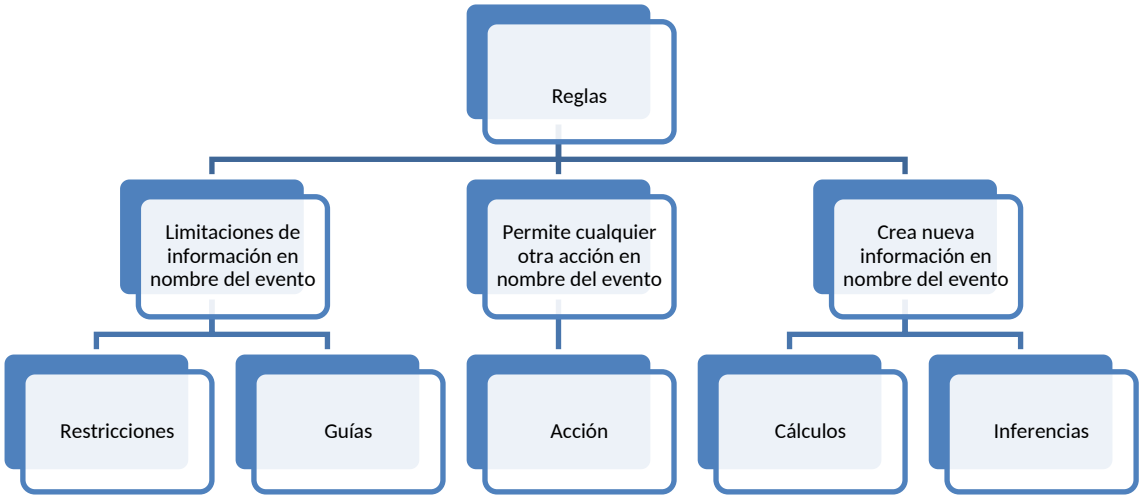

Figura 53: Desglose de la clasificación de Barbara Von Halle.

Restricciones (Constraints)

Una restricción puede ser una limitación obligatoria o un control sugerido sobre el comportamiento de los eventos del negocio. Una restricción obligatoria es una afirmación completa que expresa una circunstancia incondicional que debe ser verdadera o no verdadera para el evento del negocio para completar con integridad. Algunos ejemplos de restricciones obligatorias son:

- Un cliente no **debe tener** más de 10 órdenes abiertas al mismo tiempo.
- El monto total en dólares del pedido del cliente **no debe** ser mayor al límite de crédito de dicho cliente.
- El password ingresado en el sitio web **debe ser** correcto para poder ingresar.

Guías (Guidelines)

Una guía es una declaración completa que expresa un aviso sobre una circunstancia que debe ser verdadera o no. Es una pauta que no fuerza a la circunstancia a ser verdad o no, sino

simplemente advierte al respecto, permitiendo al ser humano tomar la decisión, al ser sólo una guía advierte y no rechaza, lo que proporciona libertad de elección. Un ejemplo de una guía es: Un cliente no debería tener más de 10 órdenes abiertas al mismo tiempo.
Se incluye a las guías como una clasificación de las reglas, porque son a menudo muy importantes para la comunidad empresarial. Algunos productos de software no son compatibles con la gestión y la especificación de las reglas guía lo que significa que el desarrollador las tiene que transformar a mensajes del tipo warning.

Acción (Action Enablers)

Esta categoría corresponde a los habilitadores de la acción, es una declaración completa que prueba las condiciones de la regla y al encontrar que son verdaderas inicia otro evento de negocios, mensaje, u otra actividad, es decir inicia una nueva acción externa. Ejemplos de facilitadores de acción son:

- Si un pedido de un cliente es válido, a continuación iniciar el proceso de pedido.
- Si un cliente es de alto riesgo, entonces notificar al gerente de servicio al cliente.
- Si un miembro es demasiado veterano para el parque temático, iniciar la acción de recomendarlo a otros parques temáticos adecuados.

Activar acciones-reglas puede ser utilizado para crear una secuencia orientada a eventos del negocio, como un workflow. Es opuesto a las restricciones, las restricciones evitan que un evento ocurra, mientras que los activadores de acciones empiezan el evento.

La cuarta categoría incluye a las reglas que crean nueva información de información existente, hay dos tipos en esta categoría, cálculos e inferencias.

Cálculos (Computations)

Un cálculo es una afirmación completa que proporciona un algoritmo para llegar al valor de un término. Este tipo de algoritmos pueden incluir suma, diferencia, producto, cociente, contar, máximo, mínimo, promedio. Un ejemplo de una regla de cálculo es:

- El total de una orden se calcula como la suma del costo de cada producto de la orden más impuestos.
- Las horas a pagarle anticipadamente al guardia de seguridad se calculan como el total de dinero pagado dividido por lo que se gana de entradas por hora.

El resultado de la ejecución de una regla de cálculo es la de crear nueva información, a esta nueva información creada por una regla se le denomina conocimiento.

Inferencias (Inferences)

Una inferencia es una declaración que pone a prueba las condiciones de la regla y al encontrarlas verdaderas establece la verdad de un hecho nuevo. Ejemplos de inferencias son:

- Si un cliente no tiene facturas pendientes de pago, el cliente es un cliente preferencial.
- Si un cliente es preferencial, entonces el pedido del cliente califica para un descuento del 20 por ciento.
- Si el cliente tiene una mala calificación crediticia, el método de facturación se establece como prepago.

El resultado de la ejecución de una regla de inferencia es la creación de una nueva pieza de información, por lo tanto conocimiento. Esta nueva información, en términos de bases de datos, puede ser una nueva instancia de una entidad (por ejemplo una nueva instancia de la entidad de cliente preferencial) o una nueva instancia de un atributo (por ejemplo, un nuevo valor para la cantidad de cliente-descuento). Por lo tanto, a medida que se analiza a las reglas, se debe tener en cuenta que cuando se descubre una acción de base de datos dentro del alcance del sistema se debe clasificar a la regla correspondiente como una inferencia.

Observando la definición de reglas de negocio se puede ver que todas las reglas de negocio son acerca de los datos. Es decir, los términos definen los conceptos de datos y detalles, los hechos definen las asociaciones entre los datos, las restricciones y las guías prueban los valores de los datos, los cálculos llegan a un valor de datos, inferencias llegan a una conclusión de datos, y los habilitadores de acción evalúan los valores de los datos antes de iniciar la acción.

Se muestra a continuación un esquema general para la clasificación:

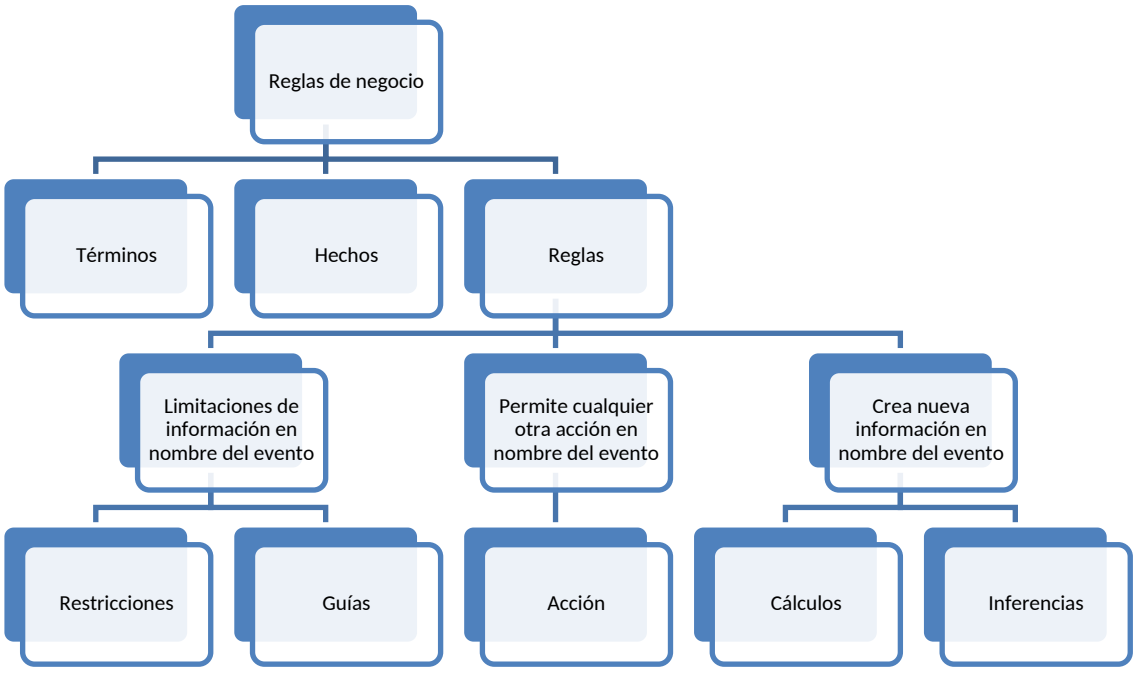

Figura 54: Clasificación completa Barbara Von Halle

Clasificación para desarrolladores de software

USoft

Usoft [41] brinda un producto hecho en Java orientado al cambio en los datos, ellos consideran que el negocio se modela solamente con el modelo de datos y las reglas de negocio. Esta empresa brinda un ambiente de desarrollo completo incluyendo un motor de reglas, y distinguen los siguientes tipos de reglas:

- Reglas de restricción
- Reglas de comportamiento
- Reglas de deducción
- Reglas de instrucciones
- Reglas de presentación

Se muestra a continuación un esquema general de la clasificación:

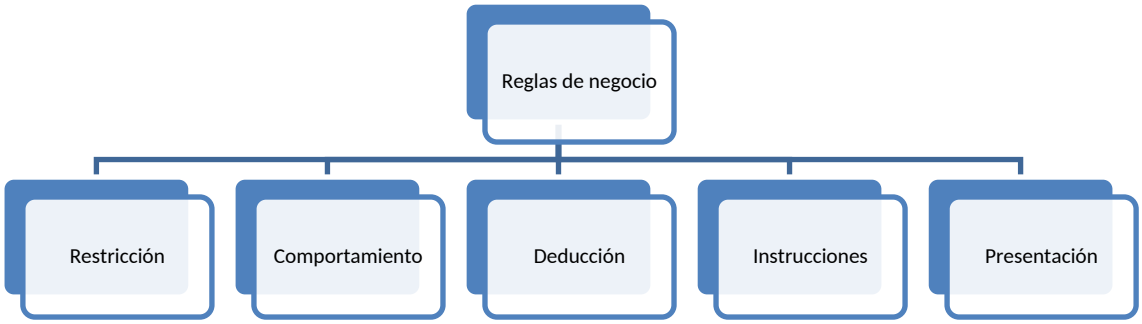

Figura 55: Clasificación USOFT.

Reglas de restricción

Se definen restricciones específicas sobre la información que va a ser guardada, establecen que es lo que no está permitido. Generalmente son valores mínimos aceptados, valores inválidos o valores que deberían ser obligatorios. Un ejemplo de este tipo de regla sería:

“Para realizar una entrega además de la dirección se debe especificar el código postal.”

Estas reglas son similares a las que hay que tener en consideración a la hora de estudiar el modelo de datos. Una categoría especial dentro de este tipo de reglas son las reglas que definen permisos y autorizaciones, es decir, especifican que usuarios o roles tienen acceso a qué partes del sistema.

Reglas de comportamiento

Expresan como se debe comportar el sistema en determinadas situaciones, especifican que debe hacer el sistema automáticamente. Tener esta clasificación nos garantiza que todas las reglas que cambien datos automáticamente van a estar agrupadas en este grupo. Estas reglas solo limitan el comportamiento del proceso, no agregan nada, ejemplos de estas reglas serían:

- Cuando el stock de un artículo se termina, las entregas de las órdenes de compra que incluyen a este artículo deben ser bloqueadas.
- Cuando el stock de un artículo es menor a 10, crear un pedido al proveedor.

Reglas de deducción

Las reglas de deducción definen o calculan como se obtiene información nueva a partir de información existente. Hay dos tipos de deducciones, las que se basan en un cálculo y las que se obtienen a partir de un razonamiento formal. Si la regla guardara esa nueva información no es una regla de deducción sino que es una regla de comportamiento. Ejemplos de esta categoría de reglas son:

- Una regla que calcula el precio total de una orden.
- Una regla que dado el código postal determina la zona de entrega del pedido.

Reglas de instrucción

Indican como se deben manejar los eventos del negocio, por ejemplo que hacer cuando una orden de compra es cancelada. Estas instrucciones se dan a conocer en la organización por ejemplo utilizando el manual de usuario. Son reglas que instruye a los usuarios sobre qué hacer en el sistema en ciertas situaciones, no están implementadas en el sistema, sino en manuales. También son usadas para guiar a los usuarios en las partes más complejas del sistema. Las reglas de instrucciones son escritas a medida que se va desarrollando el sistema para garantizar que no hay pérdidas en el conocimiento.
Un ejemplo de esta regla seria “use EUA en vez de Estados Unidos de América ya que esa abreviación esta hardcodeada en el sistema.”

Reglas de presentación

Estas reglas no refieren a conocimiento del negocio, sino que están estrechamente relacionadas con el flujo de trabajo interno de la organización. Establecen como se le debe presentar al usuario el sistema, describen el layout de la aplicación y los reportes. Ejemplos de estas reglas podrían ser:

- Todas las claves primarias deben ser presentadas en pantalla con un borde azul.
- Información sobre el producto y el precio del mismo deben ser mostrados juntos en la pantalla.

Versata

Versata [42] es una empresa de software que propone una clasificación de reglas de negocio orientada a los desarrolladores. Al igual que Usoft está orientada al cambio en los datos. Esta empresa brinda un software que incluye un motor de reglas que soporta web services, arquitectura orientada a servicios (SOA) y además reglas complejas para la toma de decisiones, procesos, transacciones y lógica de datos. La clasificación de reglas que utilizan es:

- Reglas de derivación
- Reglas de validación
- Reglas de integridad
- Reglas de restricción
- Reglas de acción
- Reglas de presentación

Se muestra a continuación en la figura 56 un esquema de la clasificación general.

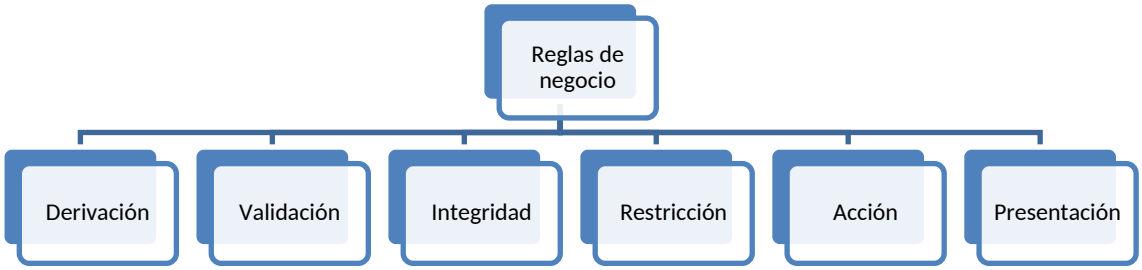

Figura 56: Clasificación VERSATA.

Reglas de derivación

Las reglas de derivación definen como el valor de un atributo es computado cuando un valor es actualizado en la base de datos. Dentro de este tipo de reglas se encuentran:

Suma: una regla de suma obtiene un valor de un atributo sumando otros atributos especificados.

Cuenta: este tipo de regla obtiene el valor derivado de atributo primario contando el número de registros en un objeto especificado. Una regla de recuento puede incluir opcionalmente una expresión que restringe los registros incluidos en el recuento.

Copia: se deriva el resultado copiando un valor especificado de un atributo a otro, por ejemplo un atributo de un objeto hijo a un objeto padre. Esta regla es útil para reducir el número de joins en consultas.

Fórmula: una regla de fórmula deriva un valor de atributo mediante la evaluación de una expresión sobre otros atributos del mismo objeto. Estas reglas pueden hacer referencia a

operaciones de modificación (inserción, actualización, eliminación), incluyen también métodos del sistema o el desarrollador que devuelven un valor, e incluyen la estructura if-then-else.

Predeterminado: una regla por defecto especifica el valor de un atributo cuando un usuario no introduce un valor para el mismo. Esta especificación puede ser un valor numérico literal, un string, o un método que devuelve un valor. Las actualizaciones posteriores del usuario pueden cambiar el valor asignado.

Reglas de acción

Se las puede llamar reglas ECA (Evento - Condición - Acción) porque introducen el comportamiento de procedimientos, dado un *evento* que está siendo monitoreado, cuando la *condición* es verdadera se ejecuta una *acción* definida en la regla. En esas acciones se pueden ejecutar métodos desarrollados internamente o llamados a sistemas externos.

Reglas de validación

Estas reglas definen limitaciones para los datos, se basan en condiciones definidas por el desarrollador. Estas reglas se pueden clasificar en:

Validaciones: cuando cambian los atributos de un objeto las reglas de validación limita los atributos de otros objetos especificados o del mismo, cuidando que no se asignen valores inválidos. Generalmente son para validar la información que ingresa el usuario.

Nulabilidad: son las reglas que obligan a los atributos a tener valores asignados, controlan que lo ingresado por el usuario no sea vacío. En general se especifica que el atributo no puede ser NULL y el diseñador de lógica de negocios establece una regla de validación que si no se cumple muestra un mensaje de error.

Tipos de dato: estas reglas controlan los valores de los tipos de los atributos ingresados o cuando son actualizados. Controlan cambio entre tipos inconsistentes por ejemplo si el tipo es numérico que no se modifique por una letra, o que no se modifiquen atributos si es el atributo que identifica un objeto.

Reglas de integridad

Estas reglas se encargan de establecer la integridad entre objetos, se refieren a integridad referencial. Se definen reglas para establecer que pasa con un objeto hijo cuando el objeto padre es actualizado o borrado, y también para indicar que pasa con el objeto padre cuando se actualiza o crea un nuevo objeto hijo.

Reglas de restricción

Son reglas a nivel de objeto para controlar los cambios en los atributos. Consisten de una condición como por ejemplo si la cantidad de stock de un producto llega a un mínimo especificado, una acción a realizar (en el caso del ejemplo la acción podría realizar un pedido

automático del producto) y un mensaje de error en caso de que sea necesario si al llegar a esa condición no hay acciones a realizar. La condición puede ser el cambio de valor en algún atributo o la invocación de un método del desarrollador que indique si se debe realizar la acción o no.

Reglas de presentación

Son reglas que definen aspectos de las interfaces de usuario de la aplicación. En general no se le da importancia a esta categoría pero estas reglas implican definir cómo va a ver la información el ser humano que utilice la aplicación. Se definen aspectos de color, como el color de fondo, el color de letra, el tipo de letra, la fuente, que cosas deben decir los labels, la alineación de los objetos en la aplicación, etc.

IBM (WebSphere)

WebSphere [43] es una familia de productos de software privado de IBM, está diseñado para configurar, operar e integrar aplicaciones de e-business a través de varias plataformas de red usando las tecnologías del Web. Esto incluye componentes de run-time (como el WebSphere Application Server - WAS) y las herramientas para desarrollar aplicaciones que se ejecutarán sobre el WAS.

Las reglas de negocio para IBM se pueden clasificar en tres tipos de reglas:

- Reglas básicas.
- Reglas clasificadoras.
- Reglas clasificadas.

Se muestra a continuación en la figura 57 la clasificación general.

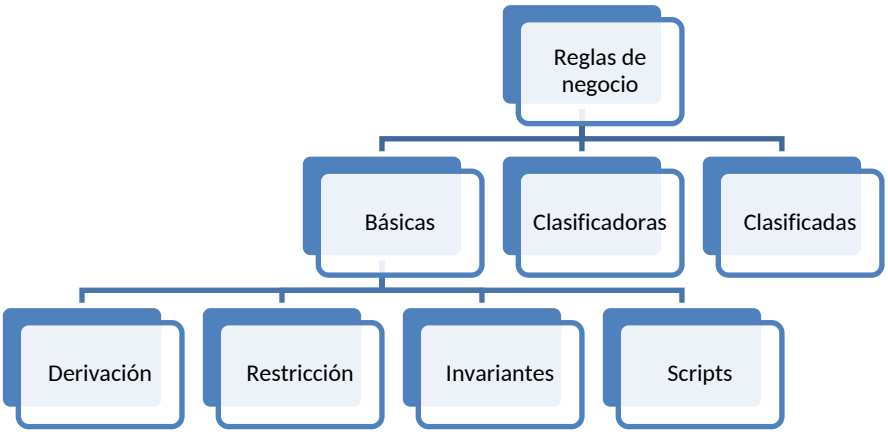

Figura 57: Clasificación IBM (Webphere).

Reglas básicas

Las *reglas básicas* son reglas que no fueron generadas por un clasificador de reglas y tampoco son reglas clasificadas. Las reglas base en general realizan cálculos específicos del negocio, tales como el cálculo de un descuento o determinar si una orden cumple los requisitos para que pueda ser liberada.

Los siguientes son ejemplos de reglas básicas, estas son libres de realizar cualquier función que tenga sentido para el negocio.

Reglas de derivación

Estas reglas se utilizan para retornar un valor. Por ejemplo una regla de derivación se puede utilizar para calcular un descuento o calcular el precio total de un pedido.

Reglas de restricción

Estas reglas confirman que una operación ha cumplido con todas sus obligaciones o que una restricción particular se ha cumplido. Por ejemplo, una regla de restricción se puede utilizar para comprobar que un valor introducido por el usuario se encuentra dentro de ciertos valores límites. Esta tecnología provee un tipo de datos especial llamado `ConstraintReturn` que puede ser devuelto por una regla de restricción, este objeto contiene un valor booleano que puede ser utilizado para producir un mensaje externo que indica que la restricción no se cumplió.

Reglas invariantes

Estas reglas aseguran que los múltiples cambios realizados por una operación están debidamente relacionados entre sí.
Por ejemplo: El balance en la cuenta de ahorro debe ser igual al balance anterior más el crédito menos el débito, si luego de los movimientos bancarios el resultado es distinto entonces el sistema está perdiendo dinero y se debe enviar una excepción.

Scripts de reglas

Estas reglas implementan un “micro-workflow” o un soporte de performance electrónico. Estas son pequeñas piezas variables de un proceso de negocio que provee asistencia a usuarios finales para obtener el máximo rendimiento de la aplicación.

Reglas clasificadoras

Se utilizan para determinar las formas en que las variables pueden ser clasificadas en el negocio. Las reglas de clasificadoras se utilizan para calcular una clasificación en una situación del negocio particular. Por ejemplo, los clientes de un banco se pueden clasificar en oro, plata y bronce en base a su historial de gastos o a la cantidad de dinero que tienen en su cuenta.

Reglas clasificadas

Son muy similares a las reglas base. La diferencia es que las reglas tienen asociada una clasificación que especifica una situación de negocio en la cual se aplica. Por ejemplo, supongamos que se tiene la situación en que los clientes se clasifican en oro, plata y bronce, y se desea calcular un descuento para cada cliente. Se va a tener tres versiones de la regla para calcular el descuento, una para la clasificación de oro, otra para la clasificación de plata y otra para la clasificación de bronce. De esas tres reglas definidas se va a aplicar la que corresponda según la situación que evaluó la regla clasificadora.

Oracle

Este es un enfoque presentado por Oracle [46] para realizar el manejo de reglas de negocio en ADF. ADF es un framework basado en la arquitectura MVC (Modelo-Vista-Controlador) para la creación de aplicaciones empresariales, provee patrones de diseño listos para usar mediante un enfoque visual y declarativo. El esquema de clasificación de este framework ayudará a entender el tipo de lógica de aplicación que se modela con reglas de negocio. Esto también provee un soporte para la compleja tarea de determinar cómo implementar mejor cada regla de negocio en ADF.

El esquema de clasificación está basado en la siguiente definición de regla de negocio:

Una regla de negocio es:
Una restricción que se aplica al estado del sistema,
Un cambio en el estado del sistema,
El uso autorizado del sistema,
O
Una acción automática que es disparada por un cambio en el estado del sistema.

Si se piensa en términos de modelado de UML se puede mapear las partes de este con los siguientes conceptos:

- Invariante: Una restricción que se aplica al estado del sistema en un punto estable.
- Pre condición: Una declaración que debe ser verdad antes de que una cosa en particular pueda tener lugar.
- Post condición: Una declaración que debe ser verdad luego que una cosa en particular haya tenido lugar.

Las invariantes son típicamente incluidas en un modelo de clases UML. Las pre condiciones y las post condiciones normalmente son propiedades de un caso de uso. UML no tiene un concepto que cubra las acciones automáticas, tal vez solo para las operaciones, en este enfoque a las acciones automáticas se las considerará como reglas de negocio.

Mostramos un organigrama en la figura 58 de lo que es la clasificación para tener una visión más global de las mismas.

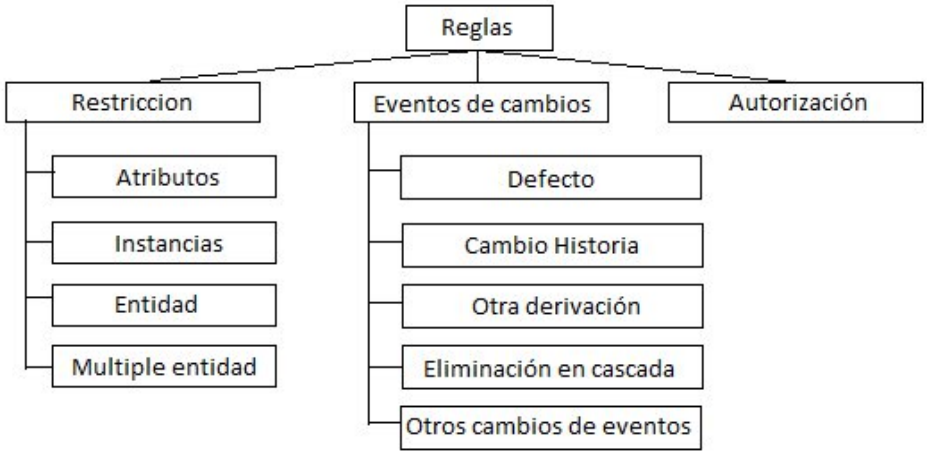

Figura 58: Organigrama de la clasificación de Oracle.

En este enfoque se presentan tres clases principales de reglas de negocio:

- Reglas de restricción.
- Reglas de evento de cambios.
- Reglas de autorización.

Reglas de restricción definen la restricción de un estado del sistema o el cambio de un estado del sistema.

Reglas de eventos de cambios definen acciones que son disparadas automáticamente por los cambios de estado en el sistema. Las acciones automáticas también pueden ser cambios en datos (insert, update, delete) o una acción fuera de la base de datos como puede ser un envío de mails o la impresión de un informe.

Reglas de autorización definen una restricción en la autorización del uso del sistema. En términos de UML las reglas de autorización son las pre condiciones.

Comenzando con las reglas de autorización, hoy en día las soluciones para implementar la autorización no requiere ninguna codificación personalizada en ADF, existen soluciones personalizadas que se pueden incorporar por fuera y por lo tanto no será discutido en este enfoque.

Realizar la distinción entre las restricciones y los cambios de eventos no siempre es sencillo. Esto es porque muchas reglas de restricción pueden ser implementadas como reglas de cambio de eventos. En otras palabras, cuando se produce un error es posible mostrar un mensaje de error (regla de restricción), o el error puede ser corregido automáticamente para el usuario (regla de cambio de evento).

Estos tipos principales de reglas han sido divididos en un número de subtipos. La siguiente sección describe cada uno de estos subtipos.

Reglas de restricción

Las reglas de restricción definen una restricción sobre el estado del sistema o sobre cambios permitidos sobre el estado del sistema, a estas restricciones sobre el estado del sistema se las conoce como Invariantes. Las restricciones que refieren a cambios en el estado del sistema se las conoce como Pre Condiciones. Las Pre Condiciones solo pueden ser comprobadas cuando en realidad se va a realizar el cambio en los datos, ya que se está comprobando si el cambio está permitido. Las invariantes pueden ser chequeadas al realizar una acción, pero también pueden ser chequeadas sobre objetos del sistema.

Es útil distinguir entre Invariantes y Pre Condiciones si se pretende proveer una herramienta para validar objetos existentes. Con respecto a ADF, la implementación de Invariantes y Pre Condiciones es la misma, y ejemplos para ambas serán presentadas en esta sección.

Los siguientes subtipos de reglas de restricción pudieron ser identificados:

Tipo	**Subtipo**
Reglas de restricción	Atributos
	Instancias
	Entidad
	Multi entidad

La clasificación anterior es bastante simple, ya que es muy fácil ver si la regla se refiere a:

- Solo un atributo en una instancia de un objeto de una entidad (Atributos)
- Dos o más atributos en una misma instancia (Instancia)
- Más de una instancia de un mismo objeto de una entidad (Entidad)
- Más de una instancia a través de múltiples objetos de una entidad (Multi entidad)

Esencialmente, esta subdivisión se encuentra en orden creciente de complejidad. Al estimar el tiempo que se necesita para poner en práctica una regla determinada, se debe tener en cuenta que las reglas de atributos son relativamente fáciles de implementar, las reglas de instancias son más difíciles, y así sucesivamente.

Reglas de atributos

Una regla de atributo define los valores que está permitidos para un atributo. Se puede realizar la siguiente sub clasificación de reglas de atributos:

Tipo	**Subtipo**	**Tipo de Regla**
Reglas de restricción	Regla de atributos	Propiedades de atributos
		Dominio
		Validadores de atributos

Propiedades de atributos

Las propiedades de atributos son las reglas que restringen el estado de un atributo, en ADF tienen una propiedad dedicada para registrar esta regla. Por ejemplo, el primer nombre de un empleado es requerido obligatoriamente.

Dominio

Cada vez que se está repitiendo el mismo conjunto de propiedades para atributos similares, por ejemplo, correo electrónico, código postal, o atributos de Si/No, se puede utilizar un *dominio* para definir estas propiedades una única vez, y volver a usarlos para atributos similares.

Validadores de atributos

Los validadores de atributos permiten definir fácilmente validaciones complejas sobre atributos. ADF BC ofrece una serie de validadores predefinidos.

Algunos ejemplos:

Validadores de comparación: Equals, NotEquals, LessThan, GreaterThan, LessOrEqualTo, GreaterOrEqualTo.

Validadores de lista: In, NotIn.

Validadores de rango: Between, NotBetween.

Validadores de largo: Equals, NotEquals, LessThan, GreaterThan, LessOrEqualTo, GreaterOrEqualTo.

Validadores de expresiones regulares: Matches, Not Matches.

Reglas de instancias

Las reglas de instancia dependen del valor de dos o más atributos dentro de la instancia del mismo objeto.

Tipo	Subtipo	Tipo de Regla
Reglas de restricción	Regla de instancias	Validadores de instancias
		Reglas de eliminación

Validadores de instancias

Estos validadores son implementados mediante métodos de validación, es posible crearlos de forma automática y retornar un booleano. Se pueden implementar validaciones sobre la instancia del estilo de: Un empleado cuyo trabajo es “Vendedor” debe tener un valor para la comisión.

Delete rules

Las reglas de eliminación se limitan cuando se intenta realizar una eliminación. Por esa razón se dispara cuando se va a llevar a cabo. Una regla que se encuentra en esta subcategoría es por ejemplo: No se puede eliminar una orden después de que se ha enviado.

Reglas de entidad

Las reglas de entidad dependen de la información de más de una instancia del objeto de la misma entidad.

Tipo	Subtipo	Tipo de Regla
Reglas de restricción	Reglas de entidad	Identificador único.
		Colección
		Otra entidad

Identificador único de colección

Un identificador único de entidad puede ser usado como una combinación de atributos que identifican de forma única a una instancia. Por ejemplo: El nombre del departamento debe ser único.

Colección

Estas son reglas que implican contar, realizar sumas o promedios dentro de una colección de instancias. Una regla de este estilo sería: No pueden haber más de 20 departamentos.

Otra entidad

Otra entidad incluye todas las reglas que involucran múltiples instancias de la misma entidad.

Reglas de múltiple entidad

Las reglas de múltiple entidad dependen de información de más de una instancia de múltiples entidades.

Las reglas de múltiple entidad se pueden sub clasificar de la siguiente forma:

Tipo	Subtipo	Tipo de Regla
Reglas de restricción	Reglas de múltiple entidad	Asociación

		Colección con padre
		Múltiple entidad simple
		Múltiple entidad compleja

Reglas de asociación

Una regla de asociación simple se define sobre una entidad relacionada con otra entidad. Una regla de asociación compleja restringe el conjunto de instancias con la cual la entidad asociada puede hacer referencia. Una regla de asociación simple sería por ejemplo: Cada línea de una orden de compra debe pertenecer a una y solo una orden. Una regla de asociación compleja sería por ejemplo: Un producto debe ser suministrado por una relación de negocios del tipo proveedor.

Colección con padre

Sirve para chequear las mismas condiciones que en la categoría de Reglas de Entidad para colección pero con alguna particularidad.

Estas reglas son disparadas cuando un nace o se modifica una instancia hijo, pero la regla en sí misma es ejecutada en los padres luego de que los cambios han sido escritos. Esto se realiza por un tema de performance, la regla solo necesita ser chequeada una vez por los padres, aún si hay varios hijos que han sido modificados o creados. Ejemplo: No puede haber más de un empleado por departamento.

Esta regla debe ser ejecutada cada vez que se agrega un nuevo empleado o que se modifica uno ya existente. Supongamos que agregamos 5 empleados al departamento 10, si esta regla es chequeada en cada entidad correspondiente al empleado podría mostrar 5 errores uno. Sin embargo, si esta condición es ejecutada en el departamento solamente habrá una falla con un solo mensaje de error.

Múltiple entidad simple

Múltiple entidad simple incluye todas las otras reglas que involucran el chequeo de información de múltiples entidades, pero tienen que ser ejecutadas en una sola entidad. Usualmente estas reglas restringen el cambio de estado más que el propio estado en sí mismo.
Un ejemplo de regla: No se puede crear una asignación a un proyecto para un proyecto que ya está cerrado.

Múltiple entidad compleja

Las reglas de múltiple entidad compleja requieren la ejecución de la regla en cada objeto involucrado en la regla. Por lo general, estas son las reglas que restringen el estado del sistema. Un ejemplo de esta regla sería, la fecha de comienzo de asignación de un proyecto debe estar entre la fecha de inicio del proyecto y la fecha de finalización.
En este ejemplo se debe tener una regla en el objeto correspondiente a la asignación del proyecto, para validar la regla cuando se introducen nuevas tareas, o cuando la fecha de comienzo o de fin del objeto de asignación de proyecto ha cambiado. Pero también debe ser

ejecutada en el objeto correspondiente al proyecto cada vez que cambia la fecha de comienzo o finalización del proyecto.

Reglas de evento de cambios

Una regla de evento de cambios define una acción automática a ser provocada por un cambio en el estado del sistema. La acción disparada automáticamente es usualmente una manipulación de datos. Sin embargo, hay acciones que no envuelven un cambio de datos, por ejemplo enviar un mail o imprimir un reporte.

Estas reglas se pueden agrupar en las siguientes sub categorías:

Tipo	Subtipo	Tipo de Regla
Reglas de cambio de eventos	No Aplica	Defecto
		Cambio Historia
		Otra derivación
		Eliminación en cascada
		Otro cambio de eventos

Reglas por defecto

Una regla por defecto define un valor que se le dará a un atributo cada vez que se crea una nueva instancia y no se da un valor para el atributo. Estos valores predeterminados se aplican solo cuando se crea un nuevo registro y no se vuelven a aplicar cuando se cambian los registros existentes.

Reglas de cambio historia

Estas reglas son ejecutadas para llevar un histórico de todas las instancias. Por ejemplo: Para todas las clases, se debe registrar la fecha de creación, el usuario que ha creado la instancia, así como la fecha de actualización y el usuario que ha realizado la última actualización.

Otra derivación

Estas reglas se utilizan para definir el valor que se le dará a un atributo cada vez que se crea una instancia o actualiza. Este valor anula cualquier valor que pudiera haber tenido asignado el atributo. Por ejemplo: Cuando se crea o actualiza una línea de una orden, el cálculo del total es Cantidad*Precio actual del producto.

Reglas de eliminación en cascada

Una regla de eliminación en cascada es una regla que especifica que en caso de relaciones 1..n, cuando se elimina al padre, los hijos serán automáticamente eliminados. Un ejemplo de esta

regla sería: Cuando se elimina una orden, las líneas de la orden serán automáticamente eliminadas.

Otras reglas correspondientes a cambios de eventos

Son reglas complejas que involucran múltiples entidades pero que estas no estás relacionadas entre sí. Un ejemplo de esta regla: Cuando el sueldo de un empleado o la comisión cambia, los datos correspondientes a la información a publicar en la revista de la empresa deben ser modificados.

Se muestra a continuación un esquema de la clasificación general:

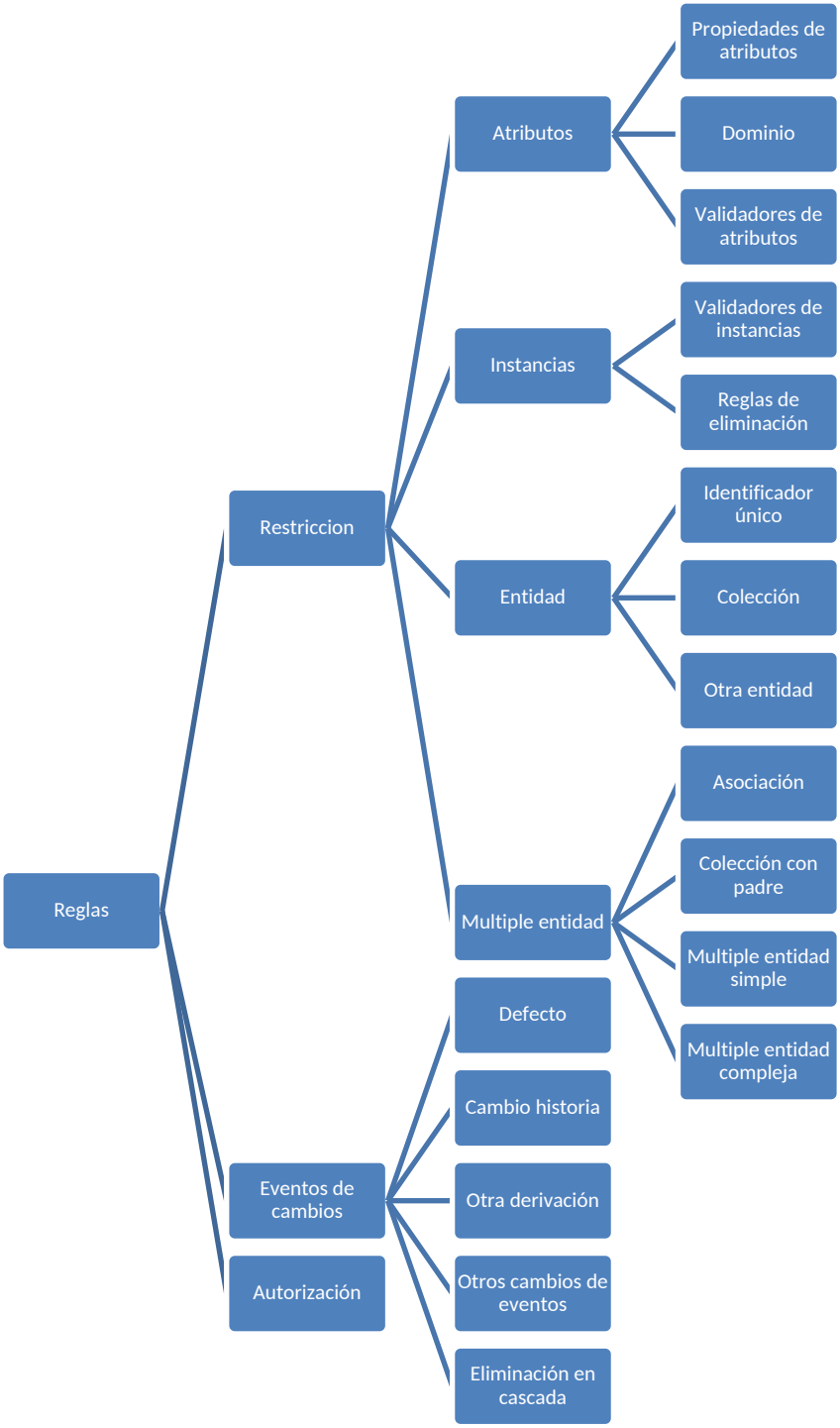

Figura 59: Clasificación Oracle.

Primatek

Primatek [44] es una empresa de consultoría enfocada a los procesos de negocio, basados en la clasificación de Ronald G. Ross, Barbara von Halle, Terry Moriarty y Versata se inspiraron para crear su propia clasificación. El objetivo de su clasificación es identificar que componente implementaría las reglas en una arquitectura SOA.

La primera observación que realizan es que la mayoría de las clasificaciones mencionadas y que hay en el Mercado se pueden mapear a la clasificación del Business Rule Group. La clasificación definida es:

Categoría	Descripción	Componentes que la ejecutan
Reglas para la transformación de datos	Transforman los datos de un formato a otro	Base de datos, herramientas de integración, herramientas de ETL (Extract, transform and load) y XSLT
Reglas de integridad referencial	Son reglas que representan y controlan las relaciones entre entidades. Son asociaciones de base de datos, multiplicidad, restricciones, triggers, etc.	Base de datos
Reglas para la validación de datos	Reglas que mantienen consistente los datos en el sistema. Por ejemplo atributos obligatorios, tipos válidos, mínimos, máximos, etc.	Base de datos y páginas web
Reglas de seguridad	Son aquellas que controlan el acceso a cierta funcionalidad o datos, se basan en roles.	Servicios de autenticación
Reglas de presentación	Reglas que permitan que contenido dinámico pueda ser presentado a los usuarios. Definen que cosas deben aparecer y en qué orden.	Páginas web, Managers de contenidos
Reglas de workflow o procesos de negocio	Reglas que se relacionan al proceso de negocio.	BPM, motores de workflow
Reglas para tomar decisiones	Son reglas para definir el aspecto dinámico de la organización.	Motor de reglas
Reglas para un motor de valorización	Reglas que representan el modelo de valorización que se aplicara en una transacción.	Motor de valorización

Tabla 9: Clasificación de Primatek.

Cuando definimos las reglas de negocio hablo de la distinción que hace Ronald Ross entre reglas de negocio y reglas de sistema, en Primatek se guían en esa clasificación marcando cuales de sus reglas son de sistema y cuáles de negocio.

Categoría	**Negocio vs Sistema**
Reglas para la transformación de datos	Sistema
Reglas de integridad referencial	Sistema
Reglas para la validación de datos	Sistema
Reglas de seguridad	Sistema
Reglas de presentación	Sistema
Reglas de workflow o procesos de negocio	Negocio
Reglas para tomar decisiones	Negocio
Reglas para un motor de valorización	Negocio

Las primeras cinco no serían reglas de negocio según esa clasificación, pero presentan justificaciones de porque hay que considerarlas. Las primeras tres son todas relacionadas a los datos, por lo que incluso se podría definir una categoría padre que agrupe las tres, y es el negocio el que determina como los datos se relacionan entre sí, la multiplicidad, los tipos de datos, etc. También es el negocio el que decide que roles tienen acceso a qué funcionalidades y que cosas deben ser mostradas al usuario, por lo que para ellos estos tipos de reglas son de negocio y es importante incluirlas.

Además hacen una comparación con la clasificación de Ronald Ross presentada antes en este documento, con esa equivalencia muestran que sus categorías de reglas pueden compararse con los tipos de reglas esperados en esa otra clasificación.

Categoría	Rechazos	Disparador/ Cálculo	Disparador /Derivación	Disparador /Habilitación	Disparador /Copia	Disparador /Ejecución
Reglas para la transformación de datos					X	
Reglas de integridad referencial	X					X
Reglas para la validación de datos	X					
Reglas de seguridad	X					
Reglas de presentación				X	X	X
Reglas de workflow o procesos de negocio				X		X
Reglas para tomar decisiones	X	X	X	X	X	X
Reglas para un motor de valorización		X				

Tabla 10: Comparación entre la clasificación de Ronald Ross y Primatek.

Se muestra en la figura 60 un esquema de la clasificación general.

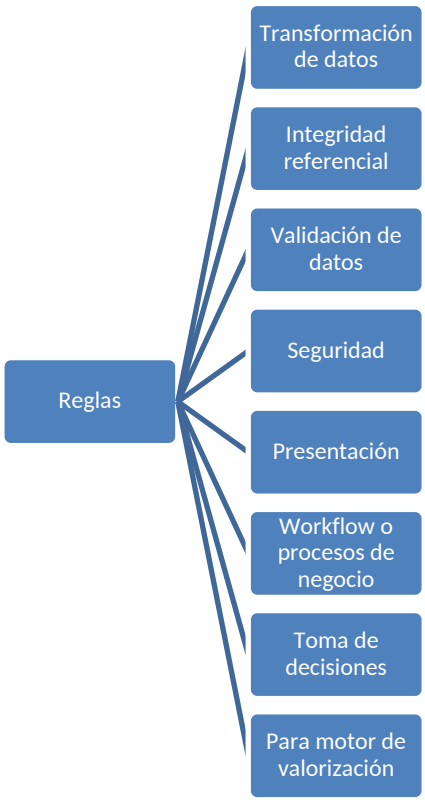

Figura 60: Clasificación Primatek.

ANEXO E: Herramientas de software para la gestión de reglas

Además del BRMS de Jboss, estudiamos otras herramientas para distinguir que funcionalidades brindan y como se integran con otras herramientas de desarrollo.

MICROSOFT

El BizTalk Server es una plataforma de integración de procesos de negocio que incluye al Business Rules Framework (BRF)[37] como una librería de .Net que contiene una serie de módulos, componentes y herramientas. Los módulos principales se componen del compositor (Business Rule Composer) de reglas de negocio con políticas de construcción, del Rule Engine Deployment Wizard, que es un asistente para deployar las políticas creadas en el compositor de reglas y del motor de reglas en sí mismo que es el que ejecuta las políticas creadas. Las políticas serían las reglas de negocio utilizadas para remplazar la codificación de las políticas de negocio que cambian constantemente y que pueden estar en procesos de negocio complejos. El BRF ayuda a incorporar llamadas a esas reglas y permite que los analistas del negocio las actualicen fácilmente. El motor de reglas es un motor de inferencias que puede asociar a las reglas a cualquier objeto de negocio (componentes de .net), documentos XML, o tablas de la base de datos. Las reglas serán utilizadas para determinar la ruta de ejecución de un proceso de negocio, basándose en la determinación de los resultados de la ejecución de la regla. El compositor de reglas permite crear las reglas agregando predicados y hechos y definiendo acciones, esto se puede hacer en la interfaz gráfica que brinde el compositor. Se puede agregar operadores como AND, OR y NOT para generar condiciones y comparaciones complejas. El compositor no solo permite crear, sino también testear, publicar y deployar múltiples versiones de las reglas de negocio y vocabularios lo cual facilita el mantenimiento y manejo de las mismas. El Business Rules Composer puede ser usado no solo por desarrolladores sino también por administradores y analistas del negocio ya sea para desarrollarlas o aplicarlas. Se define un vocabulario para las reglas de negocio, que son nombres definidos por los usuarios para los términos y hechos definidos en las condiciones y acciones, esto hace que sea más fácil de leer y entender, y pueda ser compartido por más usuarios. El Rule Engine Deployment Wizard ofrece exportar reglas o vocabularios de SQL a XML, importar reglas o vocabularios de XML a SQL, y también deployar las reglas dejándolas disponibles para actualizarlas y usarlas en el motor de reglas.

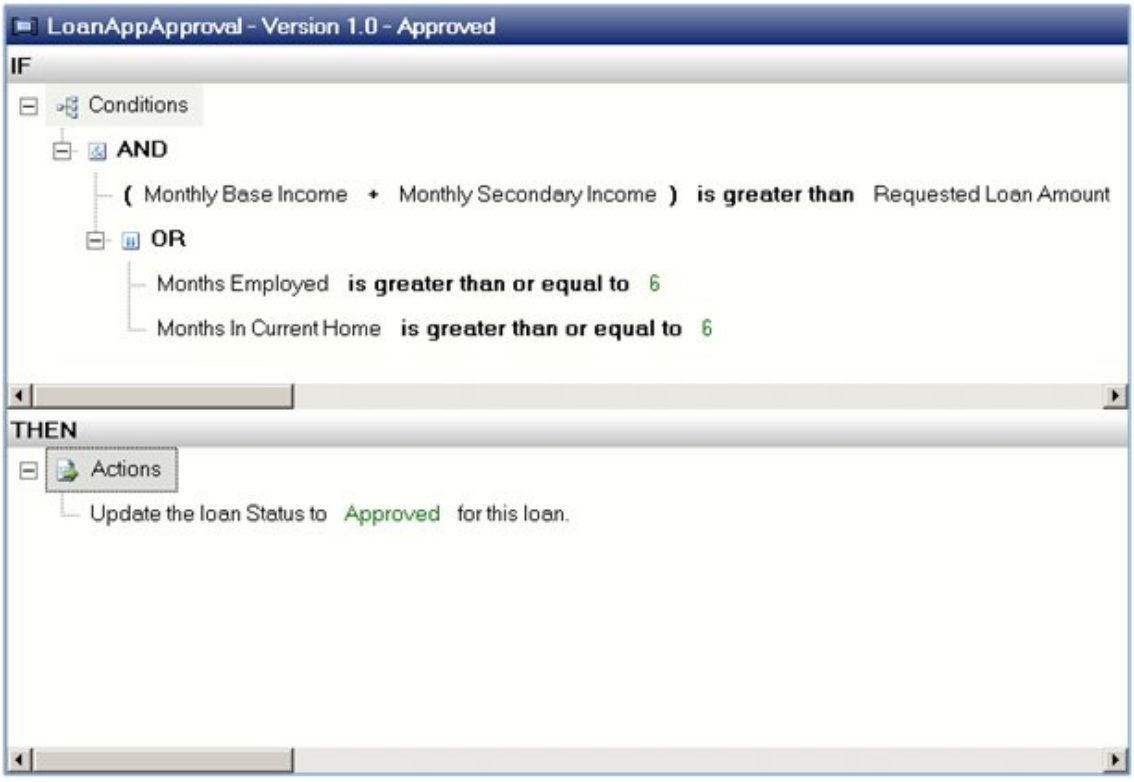

Figura 61: Ejemplo de definición de regla BizTalk Business Rule Composer.

IN RULE

Es una solución de Microsoft para el manejo, la ejecución y la auditoria de reglas de negocio, para el desarrollo en .net. Se integra con las tecnologías Microsoft como Azure, BizTalk Server, SharePoint 2010, Silverlight, Windows Workflow Fundation, etc. La auditoría esta en las manos de los expertos del negocio, los cambios en las reglas no necesitan ningún tipo de programación. En la imagen 62 se puede ver todo lo que ofrece InRule [27].

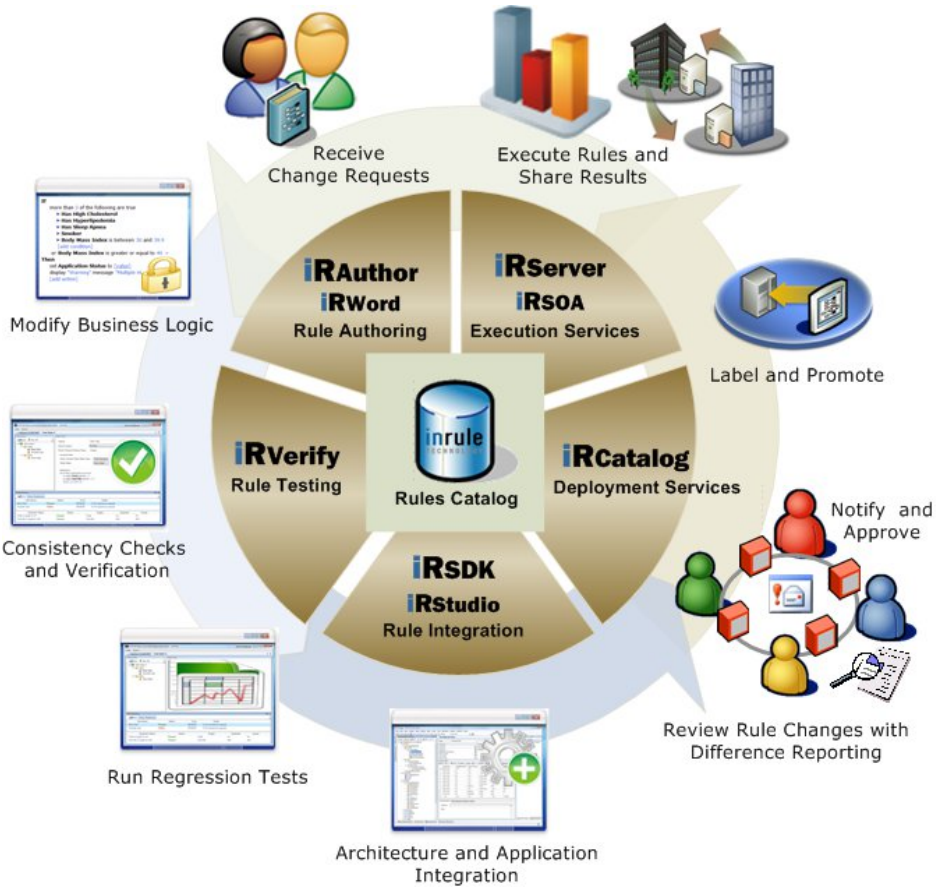

Figura 62: Herramientas que ofrece InRule.

Los componentes se pueden agrupar en diferentes categorías.

La primera categoría que veremos es la de Autoría que incluye tres componentes:

***irAuthor*:** Es un entorno para la definición y el mantenimiento de reglas de negocio. Permite definir vocabulario específico y semántica específica para que las reglas sean más simples de entender y compartir en las diferentes áreas del negocio. Las reglas de negocio pueden ser expresadas "Business Language Authoring", tablas de decisión y templates de reglas.

***irVerify*:** Entorno de testeo integrado para ejecutar y debuggear aplicaciones con reglas de negocio. Es posible realizar testeos sin la necesidad de ejecutar las pruebas en una aplicación completa o de un ambiente de testeo separado. Permite realizar la ejecución de la regla mostrando el paso a paso, se pueden utilizar filtros para observar solo la información relevante así como mostrar la entrada y la salida de la misma. Es posible guardar la información del testeo para futuros testeos de regresión.

***irWord*:** Permite la creación de reglas ejecutables y cálculos dentro de Microsoft Word. Esto permite una transición más razonable entre los requisitos y la lógica de ejecución. Provee las mismas funcionalidades que irAuthor para la definición de reglas. Se pretende que con esto se aumente la eficiencia y eficacia a la hora de definir las reglas al poder hacerlo en un documento Word.

La segunda categoría es la de Almacenamiento y Manejo que incluye un componente:

irCatalog: Un servicio para el repositorio de que reglas provee variedad en las capacidad del manejo de las reglas. Es posible que una vez almacenadas en el repositorio estas puedan ser utilizadas sin la necesidad de recompilar el código.

La tercera categoría incluye a los componentes que permiten la integración con otros sistemas:

irSDK: Contiene un SDK provee a los desarrolladores la capacidad de integrar inRule en sus aplicaciones de negocio. Se da la posibilidad de elegir entre diferentes enfoques para integrar inRule de la manera más conveniente. irAuthor es concebido como un framework extensible que permite crear o remover reglas y para realizar el manejo de las reglas de negocio sin demasiada extensión de código.

irStudio: Permite a los desarrolladores .NET dar permisos a otros usuario para que estos puedan crear, editar y manejar el conjunto de reglas.

Se permite la integración e interoperabilidad con diferentes tecnologías Microsoft, por ejemplo:

- Azure
- BizTalk Server
- SharePoint 2010
- Silverlight
- Windows Workflow Foundation

Por ultimo está el componente de ejecución con dos componentes:

irServer: es un motor de reglas de alta performance, es el sistema nervioso central de inRule. Provee servicios adicionales como manejo de estados, selección de ejecución (secuencial, optimizada, etc.), monitorea la performance, accede a la base de datos, puede dar alertas y enviar mails.

irSoa: facilita el acceso al motor de reglas de inRule mediante servicios.

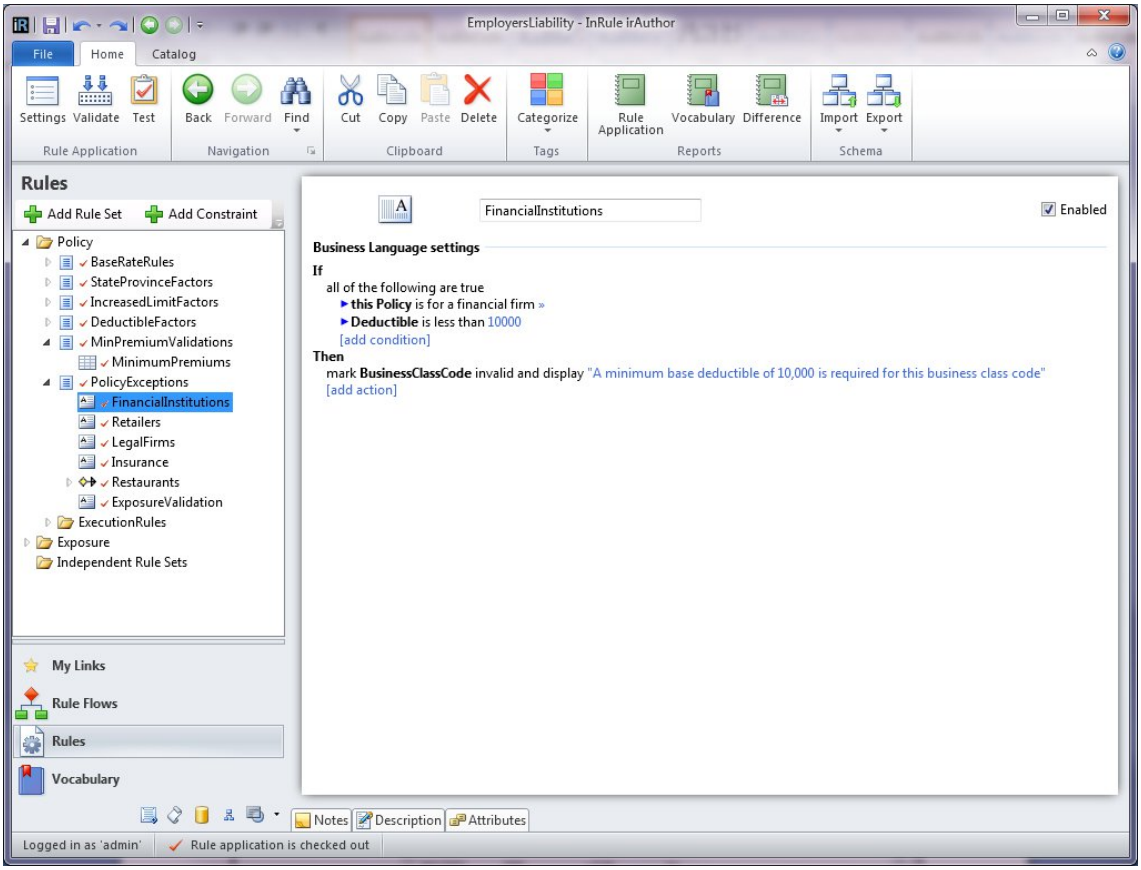

Figura 63: Imagen del editor de reglas de InRule.

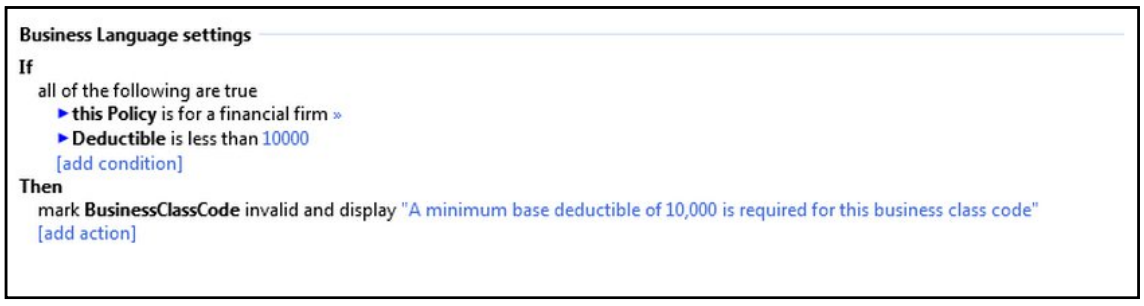

Figura 64: Regla de negocio expresada en InRule.

Open Rule

OpenRule [30] es un BRMS open source que le da el control a los analistas de negocio minimizando el software y las curva de aprendizaje ya que el diseño se hace directamente en Excel. Integra las reglas de negocios con aprendizaje automático y técnicas de optimización, soporta cualquier aplicación web basada en reglas de negocio. Esta integración se logra combinando Excel, Eclipse y otras herramientas open source. OpenRules provee tablas de decisiones para representar la estructura de las decisiones de la organización. Un usuario puede modificar las tablas existentes para hacer un diseño concreto sobre cierta área del negocio, provee mecanismos para generar patrones permitiendo que se escriba la lógica de las reglas una vez y se pueda re usar en varios lugares.

La mayoría de los motores de reglas ofrecen alguna herramienta del estilo de Excel para administrar las reglas, y la mayoría de los analistas de negocio usan herramientas como Excel y Word para presentar las reglas de negocio. Luego de seleccionar algún producto deben hacer el mapeo de las reglas, en cambio OpenRules permite crear, modificar y ejecutar las reglas directamente desde Excel, o algún software del estilo como OpenOffice o GoogleDocs. Incluso permite generar formularios web desde Excel, generando layout de formularios web complejos en tablas de Excel, esos formularios se traducen automáticamente a páginas HTML, esto sin la necesidad de aprender el lenguaje o aprender a usar herramientas para crearlas. Se logra minimizar la necesidad de desarrolladores de software pero si se los necesita para colaborar y lograr la integración entre las decisiones de los analistas del negocio con la infraestructura del software. Hoy en día el IDE de Eclipse es la herramienta por defecto utilizada para el desarrollo del software, y OpenRules tiene un plugin para permitir debuggear, versionar las reglas, hacer deploy de los Web Services, etc, además de lograr un fácil acceso a todas las herramientas de Java.

OpenRules cubre todas las fases necesarias en el manejo de Reglas de Negocio:

- Descubrimiento de reglas
- Cosecha de reglas
- Automatización de reglas
- Análisis y testing de reglas
- Integración de reglas
- Ejecución de reglas
- Mantenimiento de reglas

A continuación presentamos algunos ejemplos de cómo se definen las los términos, reglas y tablas de decisiones en tablas de Excel. En la siguiente tabla de decisión se definen decisiones a partir del glosario:

Glossary glossary		
Variable Name	**Business Concept**	**Attribute**
Gender	Customer	Gender
Marital Status		maritalStatus
Greeting	Response	Greeting
Salutation		Salutation
Current Hour		Hour

En la primera columna de la siguiente tabla se define el nombre de la decisión y en la segunda columna se define la tabla de decisión que implementa la decisión, en este ejemplo se decide como saludar dependiendo de la hora actual.

//DT DefineGreeting	
If	Then
Current Hour	**Greeting**
0-11	Good Morning
11-17	Good Afternoon
17-22	Good Evening
22-24	Good Night

La tabla define una palabra para el saludo dependiendo del género y estado civil:

DT DefineSalutation					
Condition		Condition		Conclusion	
Gender		**Marital Status**		**Salutation**	
Is	Male			Is	Mr.
Is	Female	Is	Married	Is	Mrs.
Is	Female	Is	Single	Is	Ms.

ILOG IBM

ILOG JRules [32] es un BRMS que brinda un control automatizado sobre las decisiones de negocio a los analistas, arquitectos y desarrolladores. El BRMS se usa para definir, instalar, ejecutar, monitorear y mantener la lógica de las decisiones de la empresa. Las reglas de negocio formalizan las políticas de negocio en sentencias del estilo IF-THEN, una política de negocio puede estar formada por varias reglas de negocio, de esta forma las reglas quedan entendibles para los usuarios y pueden ser ejecutadas por el motor de reglas. Con esta forma las reglas son empaquetadas en conjuntos de reglas y llamadas desde el código de una aplicación como una entidad, de esta forma los cambios en las decisiones de la organización no implican cambios en el código de la aplicación, las reglas las pueden escribir los desarrolladores utilizando herramientas de desarrollo o un analista de negocio. A continuación se presenta el proceso que proponen para incorporar las reglas a la aplicación:

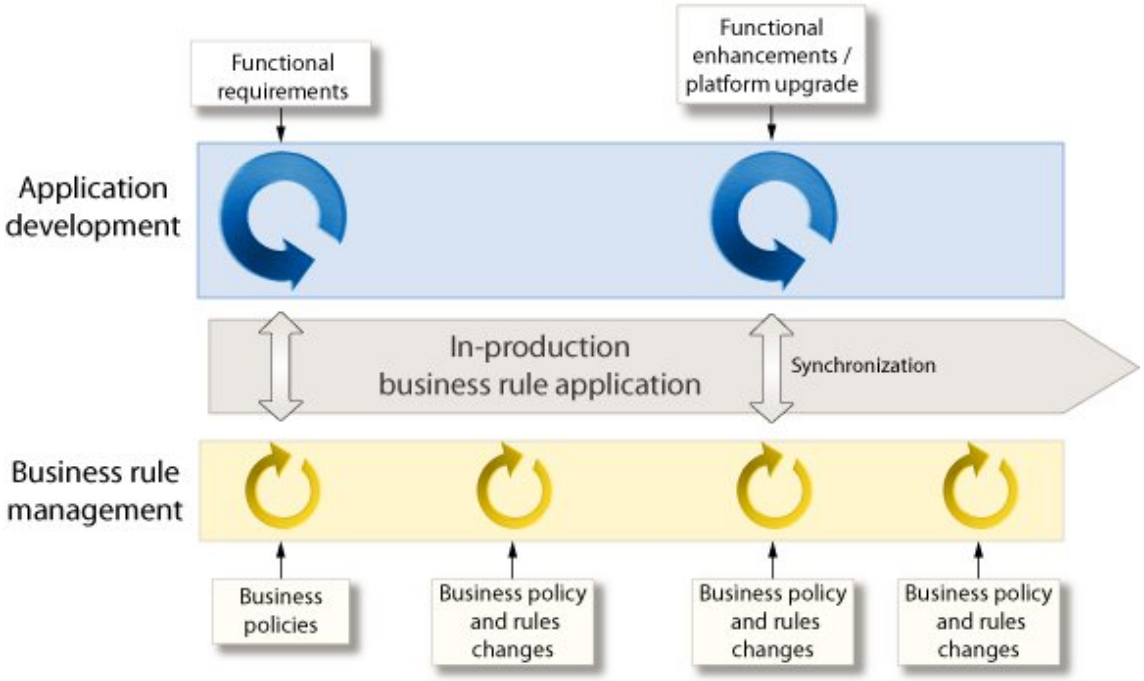

Figura 65: Proceso que propone IBM para incorporar reglas.

Con esta separación las decisiones de las políticas de negocio y la arquitectura de la aplicación pueden ser manejadas asincrónicamente. JRules tiene un ambiente para diseñar, desarrollar, instalar las aplicaciones con reglas de negocio. La separación entre los desarrolladores y los analistas de negocio permite que cada uno utilice herramientas que se corresponden a sus habilidades y conocimientos. Los desarrolladores de software utilizaran ambientes de desarrollo para Java y se enfocaran más que nada en el desarrollo del sistema, y los analistas de negocio se enfocan más en testear y manejar las reglas por lo que usaran herramientas para auditarlas. Ambas herramientas deben tener acceso a un ambiente de ejecución de reglas para poder testearlas, validarlas y ponerlas en producción.

JRules divide la implementación en tres áreas y provee módulos que apuntan a los roles de los usuarios específicos de cada área, las áreas serían:

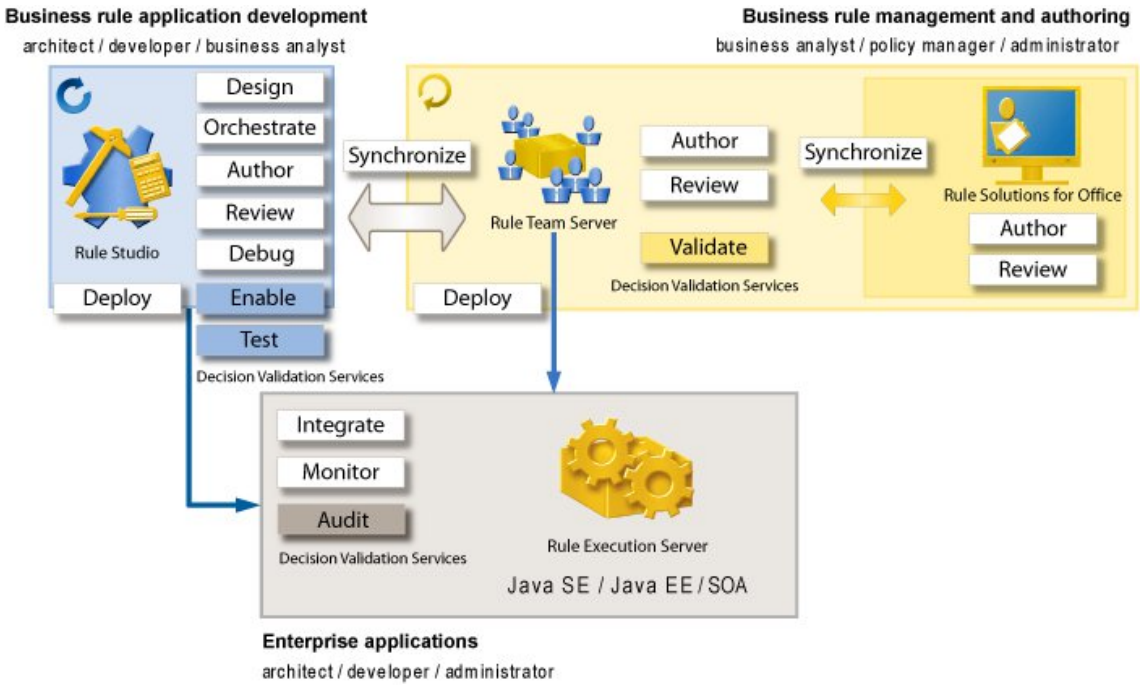

Figura 66: Áreas, roles y herramientas definidos por IBM.

En el área de desarrollo de la aplicación de reglas de negocio, los desarrolladores de software trabajan en el Eclipse diseñando una aplicación Java y un proyecto para el manejo de reglas. Usan servicios de validación de decisiones, también pueden testear las reglas con escenarios reales o ficticios, y solucionar problemas en el Rule Team Server. El vocabulario lo definen los desarrolladores con los analistas según el modelo de objetos del negocio, luego los desarrolladores implementan el modelo de objetos del negocio en implementaciones XML o Java, y si el modelo ya existe puede ser importado en JRules, el Rule Studio también proporciona editores para el modelo.

El área de los usuarios del negocio se encarga de manejar y hacer la auditoria de las reglas de negocio en el Rule Team Server, ahí pueden escribir las reglas y hacer un mantenimiento de las mismas, tanto durante el desarrollo del sistema o ya estando en producción. Los usuarios pueden hacer simulaciones para el testeo, y los managers de las políticas de negocio a través de los RuleDocs que son documentos de Microsoft Office realizan auditorias de las reglas de negocio, estos documentos son publicados en el Rule Team Server y así mantener esos documentos al día con los cambios en el negocio.

Los administradores tienen acceso al Rule Execution Server para monitorear las reglas deployadas en el servidor y manejar los servicios de decisiones, realizan la auditoria del sistema.

ORACLE

Oracle Business Rules (OBR) [33] integra sin problemas procesos de BPEL (Oracle Business Process Execution Language) en una arquitectura SOA (Service-Oriented Architecture) para permitir decisiones complejas estructuradas en documentos XML. La lógica de la aplicación es modelada visualmente en el Oracle JDeveloper permitiendo que sea diseñada, modificada y entendida por los analistas del negocio además de los desarrolladores de software. Todos los datos utilizados para tomar decisiones son representados explícitamente en documentos XML. Se considera que una regla de negocio tiene una parte IF y otra THEN, la parte con IF controla uno o más aspectos del negocio, si pasa el control entonces una o más acciones serán ejecutadas en la parte THEN.

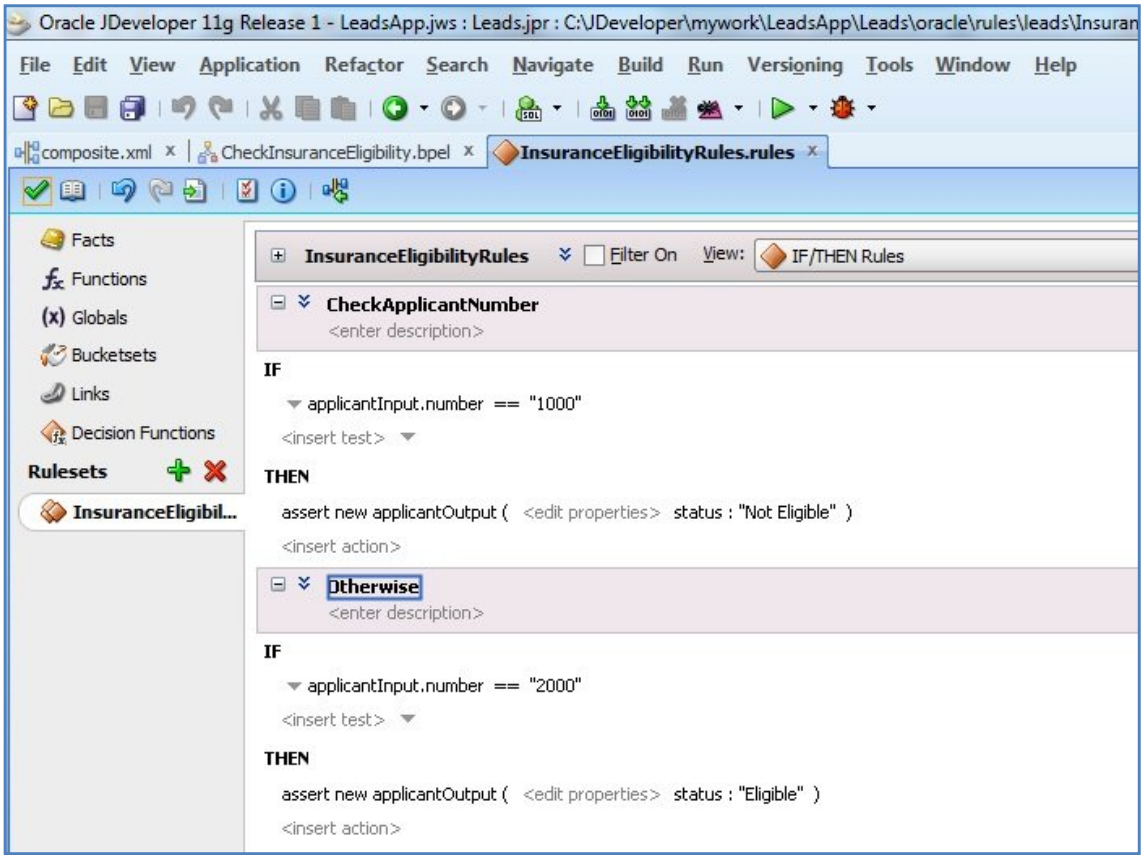

Figura 67: Editor de reglas de Oracle Business Rules.

OBR reconoce patrones para definir el tipo de términos usados en las reglas. Para analizar las combinaciones de los posibles valores de los términos de negocio se usan tablas de decisión. El OBR Designer analiza la tabla de decisiones en busca de reglas en conflicto o reglas que deberían estar y no están. Se pueden definir los términos del negocio en un archivo XML y luego importarlo en el OBR Designer como se muestra en la figura 68.

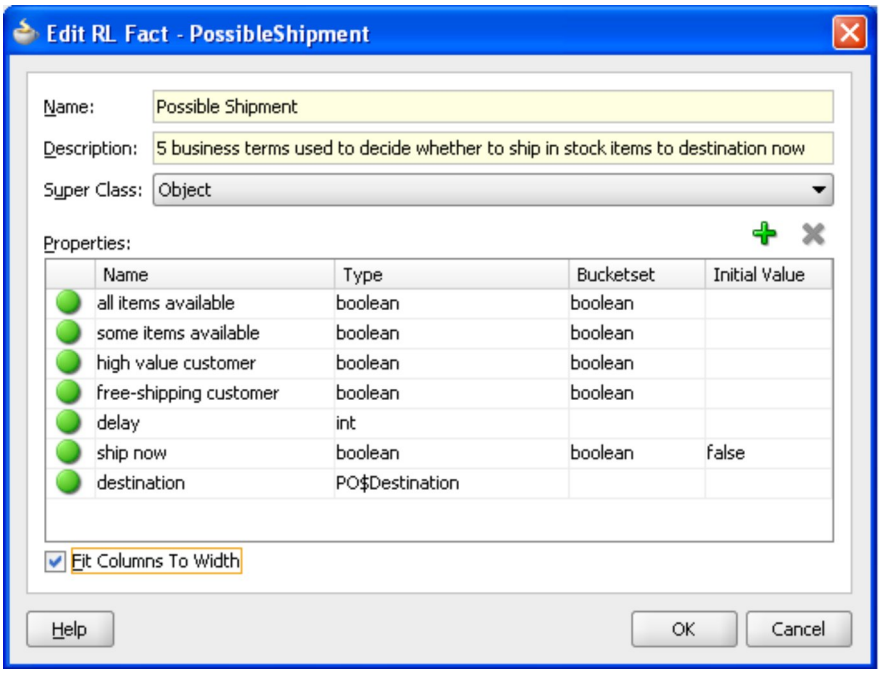

Figura 68: Definición de los términos del negocio.

Generalmente el proceso de negocio usando reglas y BPEL empieza aceptando un archivo XML como input, luego se invocan servicios y se obtiene un documento XML como output. El proceso de negocio en sí mismo es un servicio, y ese servicio puede invocar a otros servicios, incluso invocar servicios externos al sistema, las reglas de negocio son encapsuladas en BPEL como servicios con documentos XML que contienen los términos y definiciones de negocio. Oracle está trabajando en el modelado enfocado al lenguaje natural controlado para la auditoria de las reglas.

ANEXO F: Implementación de la gramática.

A continuación presentamos la gramática completa del Editor de reglas:

```
grammar org.xtext.example.mydsl.MyDsl with org.eclipse.xtext.common.Terminals

generate myDsl "http://www.xtext.org/example/mydsl/MyDsl"

Regla:
	regla += (TieneQueTener | TieneProhibido | HaDeSerEntendidoComo |
HaDeSerEjecutadoCuando)+
;


TieneQueTener:
	"Nombre:" nombreRegla=QualifiedName

	"Regla (Validacion): Toda" entidad=Entidad "tiene que tener"
condEntPpal=(CondicionEntidadPpal) andorExter+=AndOrNuevaEntidad* si=(CondicionSi)?
"fin."
;


/* Condiciones unicamente de la entidad principal */
CondicionEntidadPpal:
		"( (" ( atributoEntidad=CondicionAtributoEntidad | ( "la propiedad "
atributo=Atributo))
			(
				(comparacion=ComparacionInterna? ")"
andor+=(AndOrInterno)*) | comparacion=ComparacionExterna ")"
andorext+=(AndOrInternoNuevaEnt)*
			)
	")"
;

AndOrInterno:
	cond=("y (" | "o ("(atributoEntidad=CondicionAtributoEntidad|("la propiedad "
atributo=Atributo)) comparacion=ComparacionInterna? ")"
;

AndOrInternoNuevaEnt:

	cond=("y ( la propiedad del concepto anterior "|"o ( la propiedad del concepto
anterior ") atributo=Atributo comparacion=ComparacionInterna? ")"
;
 /* ********************** */
```

```
/* Condiciones de una entidad fija con una entidad nueva */
AndOrNuevaEntidad:
        "y ( ( la propiedad " atributo=Atributo comparacion=ComparacionExterna ")"
andor+=(AndOrEntidadNuevaEntidad)* andorext+=(AndOrInternoNuevaEnt)* ")"
;

AndOrEntidadNuevaEntidad:
        cond=("y ( la propiedad del concepto principal "|"o ( la propiedad del concepto
principal ") atributo=Atributo simbolo=Simbolo comparacion=AtributoAtributo ")"
;
/* ********************** */

CondicionSi:
        "si" "( ( " atributo=Atributo comparacion=ComparacionInterna? ")"
andor+=(AndOrCondicionSi)* ")"
;

AndOrCondicionSi:
        cond=("y ( la propiedad "|"o ( la propiedad ") atributo=Atributo
comparacion=ComparacionInterna? ")"
;

ComparacionInterna
        simbolo=Simbolo
comparaciones=(AtributoAtributo|AtributoValor|OperacionAtributo);

ComparacionExterna:
        simbolo=Simbolo comparaciones=(AtributoExterno|OperacionAtributoExterno)
;

/* ***************************** */



TieneProhibido:
        "Nombre:" nombreRegla=QualifiedName

        "Regla (Validacion - Negacion): Toda" entidad=Entidad "tiene prohibido tener"
condEntPpal=(CondicionEntidadPpal) andorExter+=AndOrNuevaEntidad* si=(CondicionSi)?
"fin."
;



/* ***************************** */

HaDeSerEntendidoComo:
        "Nombre:" nombreRegla=QualifiedName
```

```
        "Regla (Derivacion): La propiedad de" entidad=Entidad "llamado" atributo=Atributo
"ha de ser entendido como" valor=Valor ("si" condicion=CondicionCuando)? "fin."
;

/* ***************************** */



HaDeSerEjecutadoCuando:
        "Nombre:" nombreRegla=QualifiedName
        "Regla (Ejecucion): El metodo de" entidad=Entidad "llamado" metodo=Metodo "ha
de ser ejecutado cuando" condicion=CondicionCuando "fin."
;
CondicionCuando:
        condicionCuando=CondicionEntidadPpal andorExter+=AndOrNuevaEntidad*
;



// ******** Variantes

AtributoAtributo:
        "que la propiedad" atributo=Atributo;

AtributoExterno:
        "la propiedad del concepto" entidad=Entidad "llamado" atributo=Atributo
;

AtributoValor:
        "al valor" valor=ValidValue
;

CondicionAtributoEntidad: "la propiedad del concepto" entidad=Entidad "llamado"
atributo=Atributo complemento=CondicionAtributoEntidadComplemento
;

CondicionAtributoEntidadComplemento:"tal que la propiedad" atributo=Atributo
;

OperacionAtributoExterno:
        "a la suma/multiplicacion de la propiedad del concepto" entidad=Entidad "llamado"
atributo=Atributo operaciones+=(OperacionComplemento)+
;

OperacionAtributo:
        "a la suma/multiplicacion de la propiedad llamado" atributo=Atributo
operaciones+=(OperacionComplemento)+
;
```

```
OperacionComplemento:
        op=("+" | "*") atributo=(Atributo | Valor)
;


//******* Terminales

Entidad:
   nombre=QualifiedName
;

Atributo:
   nombre=QualifiedName
;

Metodo:
   nombre=QualifiedName
;

Valor:
   "(valor)" nombre=ValidValue
;

QualifiedName:
  ValidID (=>'.' ValidID)*;

ValidID:
        ID;

ValidValue:
        INT|STRING
;


Simbolo:
        "igual a" | "distinto" | "mayor o igual (>=)" | "mayor (>)" | "menor (<)" | "menor o
igual (<=)"
;
```

ANEXO G: Instalación ambiente de desarrollo.

Este anexo tratará de cómo instalar el ambiente para que un usuario técnico pueda desarrollar sobre el Plugin para la Edición de Reglas de Negocio, sobre el Plugin de Testing Unitario así como también sobre el Plugin para Exportar Reglas.

Los requerimientos del sistema necesarios para desarrollar son los siguientes:

- Sistemas operativos: Windows 7, 8, XP, Linux.
- Instalación de Java 1.6 o superior.
- Microsoft Excel 2010 o superior.
- Eclipse Juno (http://www.eclipse.org/juno/).
- Servidor de aplicaciones JBoss versión 7.1.1 (http://www.jboss.org/jbossas/downloads/)
- Guvnor versión 5.4 (http://www.jboss.org/drools/downloads)
- Plugin xText (http://download.eclipse.org/modeling/tmf/xtext/updates/composite/releases/).

Estos son los pasos para la configuración del ambiente:

1) Guvnor viene en formato “war” por lo que solo colocándolo en la carpeta “standalone/deployments” del JBoss queda instalado, y una vez levantado el servidor quedará disponible para su uso.
2) El Eclipse no necesita instalación, es solo una carpeta con un ejecutable llamado Eclipse.exe, pero una vez abierto el Eclipse se debe agregar al mismo el servidor JBoss indicado.
3) Incorporar el plugin del xText al Eclipse.

Estos tres pasos sirven para configurar Eclipse y JBoss con Guvnor. Ahora vamos a explicar cómo agregar los proyectos para poder desarrollar.

Junto con los entregables de este proyecto se encuentra un archivo .zip llamado Editor. Este archivo se debe descomprimir en el directorio deseado, se recomienda que sea en la raíz del disco C. El mismo contiene varios archivos, pero para esta parte solo necesitamos los proyectos, estos se encuentran en la carpeta proyectos y son los siguientes cinco:

- org.xtext.example.mydsl
- org.xtext.example.mydsl.tests
- org.xtext.example.mydsl.ui
- TestearReglasPlugin
- ExportReglasPlugin

Todos estos proyectos incluyen dentro de una carpeta “lib” todos los jars necesarios para el desarrollo.

Finalmente el desarrollador deberá incluir estos proyectos en el workspace que considere adecuado y levantar estos proyectos desde Eclipse para comenzar a desarrollar.

ANEXO H: Instalación y configuración del editor de reglas

En este anexo estaremos hablando acerca de cómo instalar el Editor de Reglas de Negocio para que un usuario funcional pueda hacer uso de él.

Los requerimientos del sistema necesarios para instalar el Editor son los siguientes:

- Sistemas operativos: Windows 7, 8, XP, Linux.
- Instalación de Java 1.6 o superior.
- Microsoft Excel 2010 o superior.

Junto con los entregables de este proyecto se encuentra un archivo .zip llamado Editor. Este archivo se debe descomprimir en el directorio deseado, se recomienda que sea en la raíz del disco C.

Una vez descomprimido el archivo nos encontramos con una carpeta llamada Editor que contiene:

- Ejecutable llamada Editor.exe, basta con realizar doble clic sobre el ejecutable para comenzar a usar la herramienta.
- Carpeta "properties" con archivo de configuración config.properties: En este archivo de configuración se deberán setear ciertos valores para que el Editor pueda funcionar correctamente, los mismos son:
    - paqueteGuvnor: Paquete de Guvnor al que se suben los drl escritos por el usuario.
    - pathGuvnor: Url donde se encuentra deployado Guvnor.
    - nombreModelo: Nombre del modelo a descargar, este modelo reside en Guvnor en el paquete especificado en la entrada "paqueteGuvnor".
    - userGuvnor: Usuario de Guvnor.
    - passGuvnor: Password de Guvnor.
    - nombreExcelModelo: Nombre del Excel que se generará automáticamente para el testeo unitario.
- Varios archivos de configuración que no son de interés para el usuario funcional.

Recordar que en el momento que se abra el Editor se crearán dos carpetas más.

- Carpeta "modelo": Se descarga automáticamente el modelo de Guvnor asociado al paquete especificado en el archivo de configuración.
- Carpeta "excelModelo": Se genera automáticamente un Excel con las características especificadas en la sección 5.3.3 para realizar el teste unitario de las reglas.

Una vez realizados los pasos anteriores es momento de abrir el Editor. Previamente se debe crear una carpeta de workspace, se recomienda que sea en la raíz del disco C, donde se residirá el proyecto que contiene las reglas de negocio. Esta carpeta de workspace la seleccionamos cuando abrimos el Editor como muestra la figura 69.

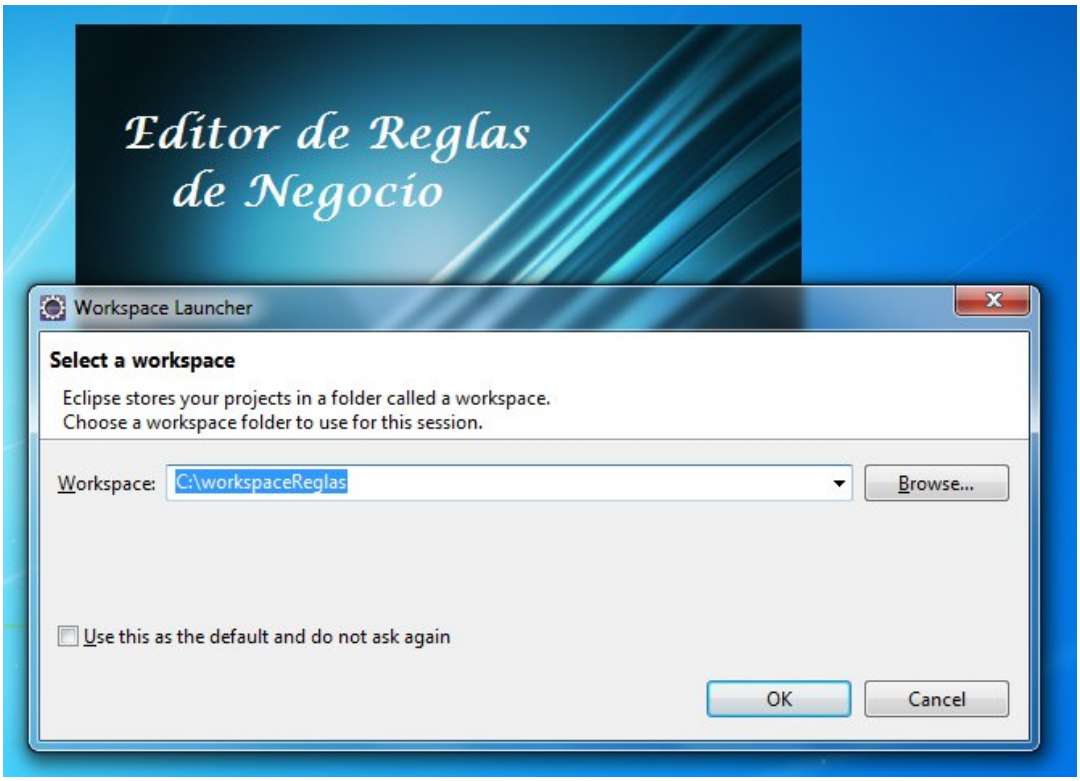

Figura 69: Selección de workspace.

La primera vez que abrimos el Editor no habrá un lugar donde guardar las reglas, hay que tener un proyecto donde puedan guardarse.

La recomendación que hacemos en este trabajo es que los drl generados residan en un repositorio SVN. Conforme a esto, una vez que se comienza a trabajar con el Editor debemos descargarnos del repositorio un proyecto ya creado donde pasaran a residir las reglas. De todas formas, puede pasar que sea la primera vez que se usa el Editor y por ende el primer proyecto que se crea para editar reglas, o sea, que éste será el primer proyecto a subir al repositorio.

Estos son los pasos para crear y subir un proyecto al SVN.

Paso 1) Creamos el proyecto.

Para esto vamos a FileàNew àJava Project.

Se desplegará en siguiente cuadro:

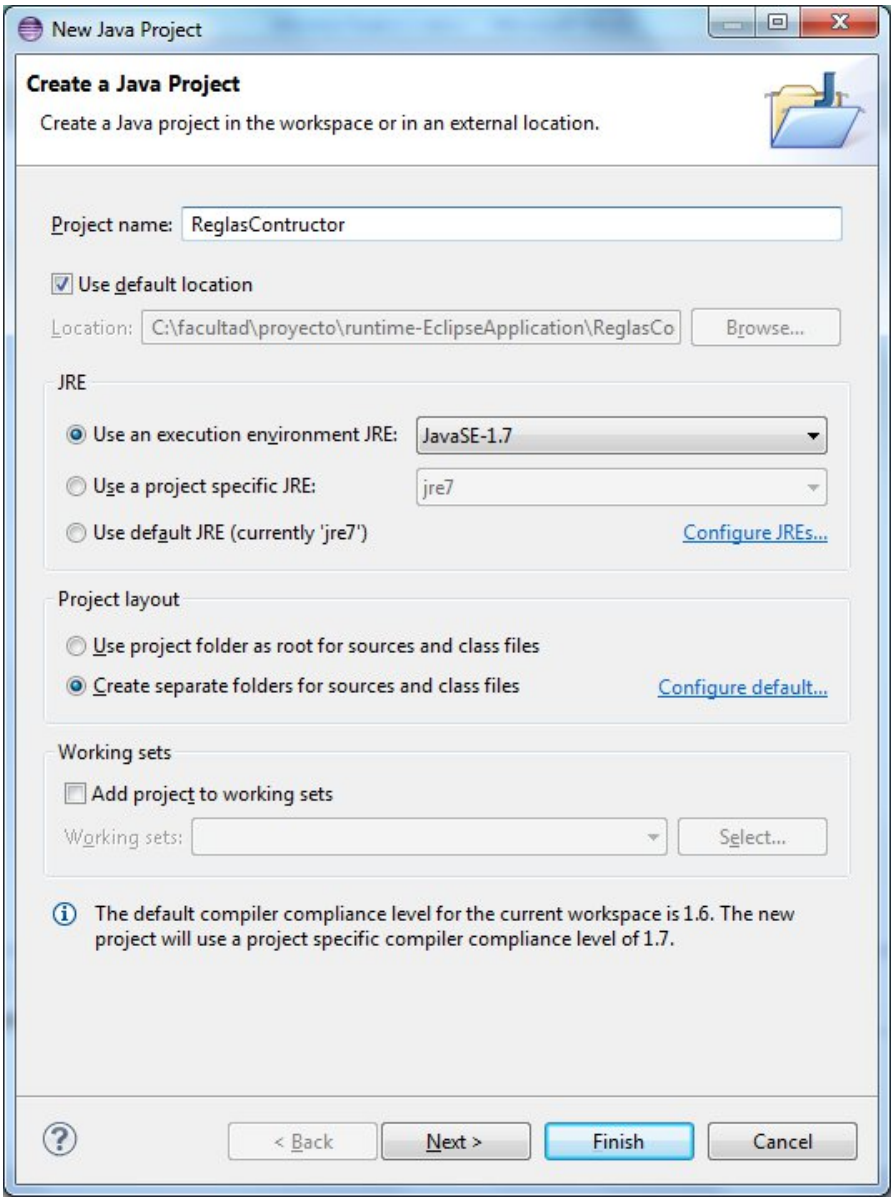

Figura 70: Cuadro dialogo creación proyecto.

Ponemos un nombre para el proyecto que se quiere a crear y se termina oprimiendo el botón "Finish". El proyecto creado se verá como en la figura 71.

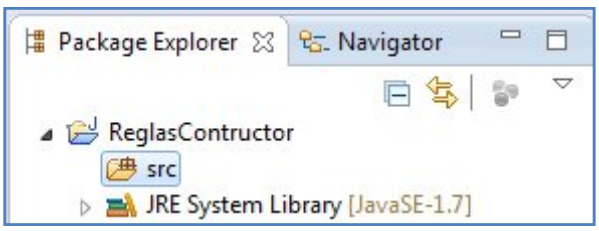

Figura 71: Proyecto ReglasConstructor.

Paso 2) Creación paquete src-gen.

Se debe crear un paquete al mismo nivel del paquete src llamado src-gen. Para eso será necesario hacer clic derecho encima del proyecto y elegir lo siguiente: NewàSource Folder.

Se abrirá el siguiente cuadro de dialogo en el que se escribe el nombre del paquete “src-gen” y se oprime el botón “Finish”.

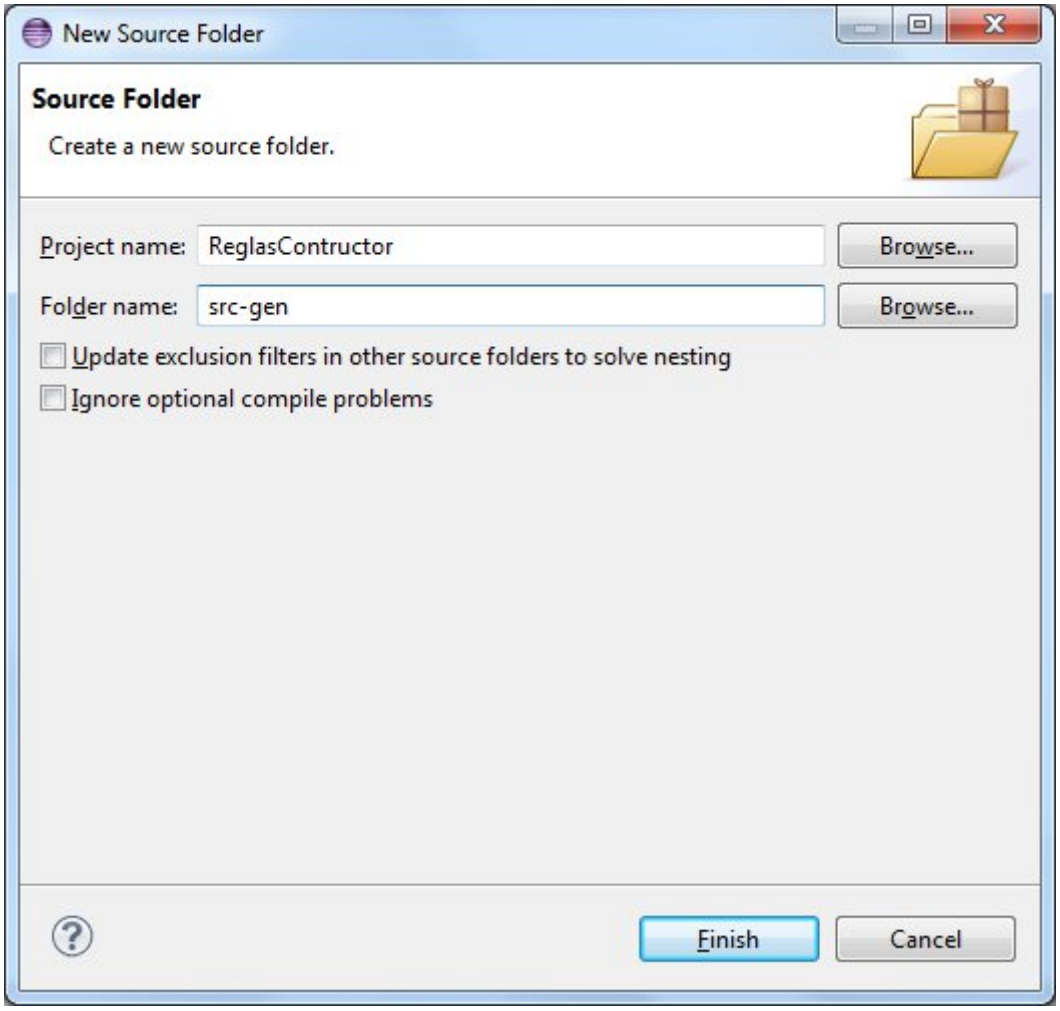

Figura 72: Creación paquete src-gen.

En el paquete src se crearán los dsl, estos serán archivos con extensión “mydsl”.
En el paquete src-gen se general los archivos drl correspondientes a los dsl. En la imagen 73 se puede ver cómo queda la estructura.

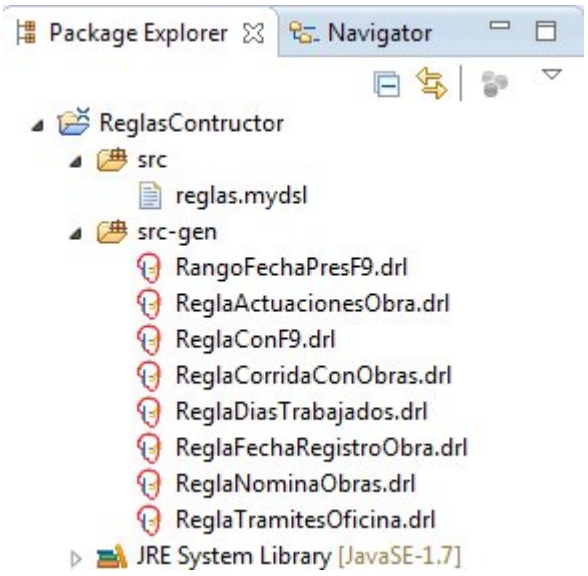

Figura 73: Estructura proyecto.

Paso 3) Subir el proyecto al repositorio.

Para subir el proyecto al repositorio se debe realizar clic derecho encima del proyecto y luego se eligen las siguientes opciones: Team à Share Project.

Se despliega un cuadro de dialogo con una grilla, de donde se debe seleccionar la opción SVN. Luego hacer clic en “Next” con lo que se desplegará el siguiente cuadro:

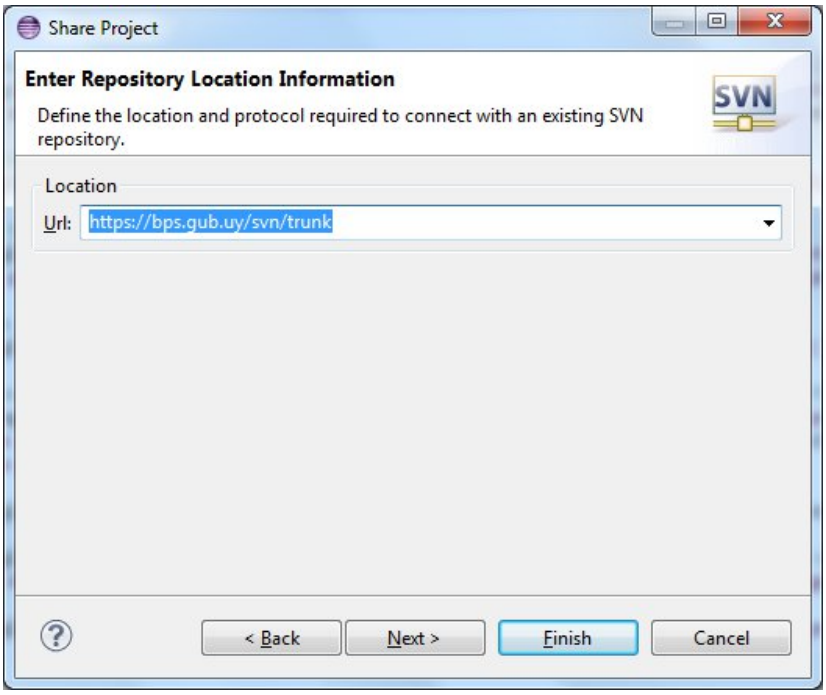

Figura 74: Cuadro de dialogo repositorio svn.

En “Url” se debe indicar ls url del repositorio SVN donde residirán las reglas. Finalmente se hace clic en “Finish” y el proyecto ya ha sido subido.

También existe la posibilidad de que no se quieran mantener las reglas en un repositorio, en este caso no será necesario subir el proyecto al mismo y se mantendrán todas las reglas localmente. Esto significa realizar solamente hasta el segundo paso.

El caso más común es el que trata de un nuevo usuario del Editor que desea descargar un proyecto del SVN, para esto debe realizar los siguientes pasos:

Paso 1) Desde el menú se debe hacer: Window à Show View à SVN Repositories. Se desplegará una lista con los proyectos que contiene cada repositorio.

Paso 2) Seleccionar el proyecto de reglas que se quiere descargar, hacer clic derecho sobre el mismo y elegir la opción Checkout.

Con estos dos pasos el proyecto estará descargado y pronto para trabajar en él.

Una vez realizados los pasos en este Anexo, ya será posible comenzar a utilizar el Editor de Reglas de Negocio.

ANEXO I: Estados de ciclo de vida a nivel organizacional

Explicaremos más en detalle de que consta cada una de las etapas presentadas en el "Ciclo de vida de las reglas de negocio a nivel organizacional" en la sección 2.20.

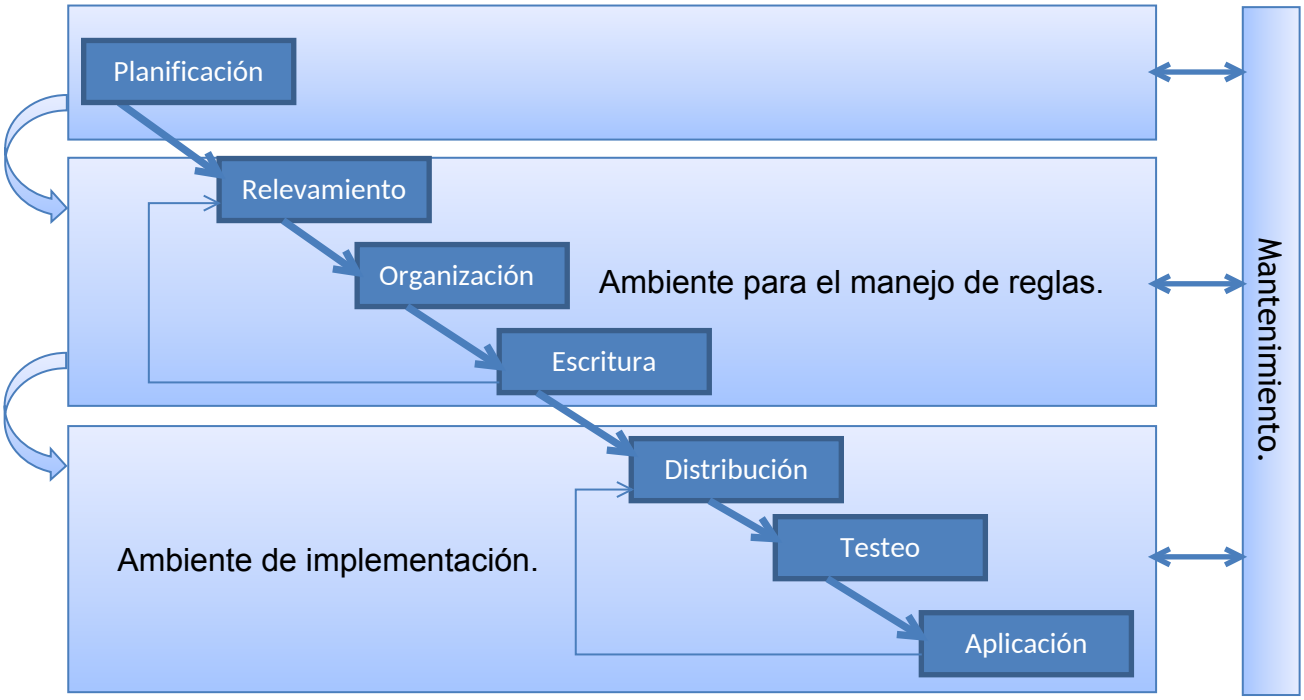

Figura 75: Etapas del proceso de incorporar reglas de negocio a la organización.

Planificación

Lo primero que se necesita es desarrollar una estrategia y un plan para el manejo de proyectos de reglas de negocio. Los actores involucrados en esta etapa son Analistas de negocio y Especialistas técnicos. Se debe realizar un estudio de como las diferentes partes del universo de las reglas de negocio impactan en los modelos lógicos, los recursos y otros artefactos incluidos en la organización, se debe tener en cuenta la visión estratégica de la sección de TI incluyendo los planes de integración a nivel empresarial.
La salida de esta etapa contiene un plan para la puesta en producción en alto nivel de las reglas por cada segmento del negocio, las relaciones entre estos segmentos, áreas de prioridad, el secuenciamiento de estas puestas en producción y un cronograma tentativo.
Los efectos de este primer paso incluyen la verificación con iniciativas similares, buscando maximizar la oportunidad de reúso de reglas en alguna etapa posterior y un mapa estratégico para la puesta en producción de las reglas de acuerdo a las necesidades del negocio.

Relevamiento

En esta etapa se hace el relevamiento de las reglas de negocio en las diferentes secciones de la organización. Los actores que participan en este estado son los especialistas técnicos, analistas de negocio, expertos en el dominio y empleados con muchos años en la organización. Los

recursos que se utilizan son por ejemplo manuales de usuarios, código de sistemas legados, contratos de negocio, notas, e-mails, manuales de procedimiento.
La salida que se obtiene en este estado es una lista exhaustiva con las potenciales reglas correspondientes a las diferentes secciones del negocio. Los procedimientos para la identificación son por ejemplo la aplicación de minería de datos en sistemas legados, lectura de manuales de usuarios así como de contratos de negocio y legislación correspondiente, por otro lado también se incluyen entrevistas con los empleados que cuentan con muchos años en la organización.
El resultado sería la exposición y la identificación formal de las potenciales reglas de negocio. Los efectos indirectos constan de una significativa mejora en la transparencia de los procesos de negocio y la oportunidad de desarrollo de código más eficiente.
En este paso es importante destacar la necesidad de permanecer enfocados en el manejo de las reglas de negocio. Debido al volumen de los procesos de reingeniería, la redimensión del código de los programas y otras oportunidades de eficiencia que son expuestas es probable que se tienda a provocar un desvío en el foco de atención.

Organización

Luego de realizar el relevamiento de las reglas de negocio, se debe realizar una verificación de que el listado obtenido representa una organización inicial de las reglas. En este estado los actores son los Analistas de negocio y los Especialistas técnicos. La entrada son las potenciales reglas de negocio identificadas en la etapa anterior y la salida es la validación de que ese es el conjunto de reglas con el que se va a trabajar, se debe incluir un plan preliminar con el *dónde* y *cómo* las reglas serán implementadas y actualizadas. Este paso suele ser uno de los que más tiempo requiere ya que necesita una extensiva intervención manual y un análisis profundo.
Los procedimientos claves incluyen la eliminación de reglas que se encuentran fuera del alcance, redundantes o no prioritarias. Se debe dejar en claro de dónde fue extraída la regla, así como realizar una clasificación de las mismas y una preparación de la regla en cuanto a que metadata debe contener. Adicionalmente se puede incluir un mapeo de *dónde* la regla será implementada, y *cómo* ésta será instalada.

Escritura

La idea principal de este estado es la transformación de datos implícitos en conocimiento explicito mediante la especificación formal de las reglas de negocio.
Los actores en este estado son los Analistas de Negocio, Personal de la gerencia que autorice decisiones que afecten las reglas de negocio y expertos que asistan con la interpretación de contratos y legislación. La entrada es la organización del estado anterior y la salida representa las reglas de negocio completamente creadas en un repositorio de reglas y expresadas en un vocabulario del negocio.
El repositorio de las reglas de negocio es manejado por los representantes del negocio en la organización, es presentado en un entorno de usuario amigable que utiliza términos estándares de la terminología del negocio, este debería ser manejado de forma centralizada. Establecer el repositorio permite a los Analistas de negocio focalizarse en la creación y administración de las reglas así como a los especialistas técnicos les permite focalizarse en la implementación.
Los efectos en este estado incluyen una mejora en el entendimiento y la comunicación de las reglas y un desarrollo más alto en cuanto a la calidad de las mismas. Habilita un desplazamiento del control de las reglas de negocio desde los Especialistas técnicos hacia los Analistas de Negocio, clarifica el manejo de las reglas de negocio y los entornos de

implementación y da una mayor flexibilidad a la hora de elegir una implementación o una solución técnica.

Distribución

Este paso consta de compartir y distribuir las reglas formales que se encuentran en el repositorio hacía el entorno de implementación que se haya seleccionado. La entrada incluye las reglas de negocio desde el repositorio y la información de los pasos que incluyen evaluaciones preliminares acerca de dónde las reglas necesitan ser implementadas y cómo ocurrirá su instalación.

Una correcta interoperabilidad es importante para mantener la estabilidad del repositorio de reglas y permitir el control de las actualizaciones de las reglas por los Analistas de negocio.

La salida incluye las decisiones detalladas con respecto a cómo la automatización entre el repositorio de reglas y los puntos de distribución se llevarán a cabo en una determinada versión.

Los actores claves son los Proveedores de BRMS, los especialistas técnicos de la organización y el personal técnico de la gerencia. En este momento es necesario hacer la elección del BRMS que se va a utilizar, las soluciones básicas que incluyen los siguientes tres elementos, un motor de reglas, manejo para la gestión de la configuración y una interfaz de acceso a usuario de negocio. La solución elegida depende del número de variables incluidas en los volúmenes de transacción, la frecuencia con que las reglas cambian, la cantidad de las reglas.

Testeo

El testeo sirve para asegurarse de que la interoperabilidad entre el repositorio y el entorno de implementación sea el adecuado, así como también asegurarse que el entorno seleccionado está la altura de lo que la organización necesita. La entrada en este paso incluye las reglas de negocio desde el repositorio por medio del entorno de implementación elegido y la salida incluye el resultado de los test. Los actores son los expertos en el dominio, los Analistas del negocio, especialistas técnicos y los proveedores de BRMS. El testing se enfocará en el testeo unitario, en el testeo de integración que incluye la interoperabilidad y conectividad desde el repositorio de reglas hacia las aplicaciones y por último en el testeo de aceptación. El testeo debe estar enfocado en la lógica de las reglas y abarcar la ambigüedad de las mismas, la precisión y su completitud.

Aplicación

Este paso pone las reglas de negocio en pleno funcionamiento, formalmente implementadas y las coloca operacionalmente activas. La entrada de este estado incluye el resultado y el feedback del testeo y la salida incluye las reglas de negocio completamente en producción. Los actores incluyen al personal de la gerencia, los Analistas de negocio y especialistas técnicos. Las técnicas tradicionales a nivel organización para desplegar nuevos sistemas son útiles en este paso, se puede pasar a producción en paralelo, por etapas o manejando pilotos. Se deben definir un plan a corto plazo referente a las medidas a tomar en caso de problemas técnicos o errores lógicos. Este paso debe arrojar como resultado un plan acerca de cómo desplegar nuevas reglas de negocio, así como la mejora de la capacidad de los Analistas de negocio para responder rápidamente ante una actualización en las reglas para que estos

puedan implementarlas. Esto último ayuda a la organización a la mejora de su flexibilidad y capacidad de reacción frente a los cambios.

Mantenimiento

El mantenimiento del entorno incluye evaluaciones rutinarias, realineaciones, repriorización y selección de dónde la próxima automatización de reglas de negocio debe ocurrir, considerando las grandes influencias del contexto. El mantenimiento estará enfocado en técnicas para la captura sistemática de cambios o la aparición de nuevas reglas, rexaminando la organización en cada uno de los segmentos de negocio, sistemas y procesos que necesiten acceder a las reglas y cómo se establecerá la comunicación con estos. Los administradores del repositorio de reglas se deben concentrar en efectivizar el reúso de las reglas y la precisión de las mismas, equilibrar la volatilidad de las mismas y la rotación de las cargas de trabajo y la actividad central del repositorio de reglas. En el mantenimiento se debe tener conocimiento acerca de nuevas soluciones de interoperabilidad entre el repositorio y los sitios de distribución, habilitando rápidos despliegues manejando la performance de los sistemas al mismo tiempo que permite a los usuarios de negocio ser propietarios legítimos de las reglas de negocio.

ANEXO J: Estudio del área de negocio.

Las obras en fiscalización presentan determinadas características, a la hora de realizar un cierre dichas características son tenidas en cuenta ya que dada una determinada combinación de las mismas se puede afirmar que una obra está en condiciones para ser cerrada. Existen tres formas de realizar un cierre de una obra desde el aplicativo Constructor, estos son:

- Cierre de obras automático masivo.
- Cierre de obras automático sin deuda.
- Cierre de obras automático con deuda.

Cierre de obras automático masivo.

Vamos a profundizar en cada uno de estas distintas formas de cerrar obras automáticamente.

Como mencionábamos anteriormente, surgió de parte de BPS la necesidad de automatizar el cierre de obras, teniendo en cuenta esto se buscó una manera de poder realizar un cierre masivo (cierre simultaneo de varias obras) de forma automática bajo determinadas condiciones en la obra. Hoy en día, el Constructor para realizar ese procesamiento masivo tiene un proceso que está dividido en dos grandes partes, por un lado lo que es llamado "*Carga del universo*" y por otro el "*Cierre de Obras*" propiamente dicho. Presentaremos un estudio de negocio exponiendo de qué constan estas dos etapas comenzando por la que se ejecuta primero, la "*Carga del universo*".

La carga del universo es un proceso previo al cierre de las obras, al ejecutarse recaba datos de diferentes sistemas a fin de obtener información que posibilitará evaluar individualmente cada obra para saber si esta puede ser cerrada, así como datos que se obtienen a nivel informativo para mostrar en diversos reportes. Estos datos que son recabados son almacenados en la base de datos del Constructor.

Los servicios externos a Fiscalización (internos al BPS) que se debieron consultar para reunir la información de las obras fueron los siguientes:

- Registro de Obras
- Registro de Empresas
- Recaudación Nominada
- Fiscalización
- Cuenta Empresa

La carga recibe determinados parámetros, a saber:

- Fecha desde
- Fecha hasta
- Número de agencia
- Tipo de obra (pública o privada)
- Con F9/Sin F9

En caso de que la opción ingresada sea "*Con F9*" se realizará la carga de las obras que hayan presentado el formulario F9 (formulario de solicitud de cierre de obra) y cuya fecha de presentación esté comprendida en el rango de fechas "fecha desde" y "fecha hasta". A su vez esas obras deben cumplir la condición de que hayan sido inscriptas en la Agencia ingresada como parámetro.

En caso de optar por la opción "*Sin F9*" (sin presentación de formulario F9), entra en juego el formulario F1 que es presentado en el momento que se inicia la gestión para comenzar a registrar una obra ante el BPS, dicho formulario contiene una estimación de la fecha de finalización de la obra. Se cargaran en este caso las obras que cumplan las siguientes condiciones:

- Las empresas correspondientes a las obras que no hayan presentado el formulario F9.
- La fecha de finalización estimada presentada en el formulario F1 se encuentra entre las fechas "fecha desde" y "fecha hasta" ingresadas como parámetro en la carga.
- La fecha de finalización estimada presentada en el formulario F1 más 6 meses, es posterior al día de la ejecución del cierre automático. Esta cantidad de meses puede variar entre una ejecución y otra, o sea que puede llegar a redefinirse.

Como se mencionaba anteriormente, se obtiene información de diferentes negocios. La siguiente tabla muestra cuáles son los datos que se extraen de cada uno de ellos a la hora de realizar la carga del universo.

Registro de obras	Recaudación nominada	Registro de empresas	Cuenta empresa
Empresa titular	Contribuyente Titular	Número interno de contribuyente	Saldo
Obra	Empresa Titular		
Padrón	Aportación Titular		
Unidad	Mes de cargo desde		
Fecha desde	Mes de cargo hasta		
Fecha hasta	Cantidad de nóminas presentadas		
Forma de realización	Cantidad de días trabajados		
Fecha de inicio	Mes de presentación de la última nómina		
Fecha de fin estimada			
Fecha de presentación del F9			
Departamento			
Características			

Figura 76: Extracción de datos de diferentes negocios.

Algunas aclaraciones sobre datos que aparecen en la tabla:

- La aportación de la titular es el rubro de la empresa en cuestión, en nuestro caso siempre el valor cuatro que es el correspondiente a construcción.
- El rango de meses de cargo corresponden a los números *yyyymm* del rango del periodo desde y el periodo hasta.

El área de *Recaudación Nóminada* recoge la información que identifica los servicios, remuneraciones y datos de la relación laboral de los trabajadores con la empresa durante el transcurso de cada mes de actividad. Cuando no existe información el *Recaudación Nóminada* para la empresa-obra-período de la obra, se descartará la misma como candidata a ser cerrada y pasa a estar en un estado que indica que "No tiene datos en recaudación", cabe destacar que de todas formas la obra es registrada en el Constructor.

Posteriormente se consulta al sistema *Guia-Gdt* de Fiscalización para saber si existen actuaciones asociadas, la existencia de algún tipo de actuación puede ser un impedimento para que la obra pueda ser cerrada, en caso de existir, se registran los datos de las mismas. Una actuación es una entidad que refleja una acción inspectora, interna o externa (esto se puede ver reflejado en una deuda) sobre una empresa o parte de ella. *Guia-Gdt* es quién se encarga de manejar la gestión de las actuaciones.

Cuenta Empresa es el área de negocio que se encarga de que cada empresa tenga una cuenta única en el BPS, en donde queden registrados todos los movimientos en aportes de forma centralizada. Teniendo en cuenta los siguientes datos de la obra los cuales se recabaron previamente:

- Contribuyente
- Empresa
- Número de obra
- Fecha desde
- Fecha hasta

Ahora se obtiene la información desglosada para la empresa en cuestión acerca de sus movimientos, es posible obtener de aquí el saldo en cuenta que tiene la misma, dato que posteriormente se utilizará como una de las condiciones a ser tenidas en cuenta para determinar si la obra debe ser cerrada o no.

Hasta aquí se presentó el proceso de carga del universo, pasamos a describir el "*Cierre de Obras*" propiamente dicho.

Una vez realizada la carga, se procede a realizar un procesamiento a cada una de las obras que fueron cargadas para determinar si corresponde o no ser cerrarse.

Este procesamiento implica evaluar determinadas condiciones pre-definidas por el usuario, estas suelen variar a corto plazo, y es necesario que el usuario del sistema pueda definirlas cuando lo crea conveniente sin la necesidad de modificar el sistema y sin la necesidad de la intervención del equipo técnico. Hoy en día las condiciones son semi-variables, a esto nos referimos con que pueden deducirse de los datos de una tabla almacenada en base, entonces pese a no tener la flexibilidad que se requiere se pueden variar los datos de la tabla y con eso se logra cambiar en cierta forma la condición. Algo no menor es que para modificar la tabla es necesaria la intervención de técnicos y que éste método no hace las condiciones lo suficientemente flexibles.

Las condiciones que se deben evaluar para determinar si una obra puede ser cerrada son las siguientes:

- La fecha desde registrada de la obra debe ser mayor o igual setiembre que 2006.
- La cantidad de días trabajados de la obra debe ser menor o igual a 150.
- Si se está evaluando una obra de una empresa privada entonces esta tiene que tener presentado ante BPS más del 70% de sus nóminas. En caso de que la obra sea pública no se evalúa esta condición.
- El saldo en cuenta de empresa debe ser menor o igual a 3 unidades reajustables, tomando el valor de la unidad reajustable del último mes de la ejecución del cierre.
- Si se está evaluando una obra de una empresa privada entonces la duración de la obra debe ser menor o igual 12 meses. En caso de que la obra sea pública no se evalúa esta condición.
- La agencia de la obra debe ser igual a 1, que es el correspondiente a Montevideo (en el cierre de este proyecto, se está extendiendo para más agencias).
- Si existe una actuación asociada a la empresa correspondiente a la obra, esta no es cerrada.
- Si la obra tiene asociado algún trámite en las oficinas 226, 229 y 2004 la obra no puede ser cerrada.
- El código del departamento debe ser igual a 1, que es el código correspondiente a Montevideo (en el cierre de este proyecto, se está extendiendo para más departamentos).

Todas las obras que pasen los filtros enunciados anteriormente serán cerradas.

Cierre de obras automático sin deuda.

Luego de tener disponible un cierre masivo de obras, se comenzó a trabajar en una funcionalidad que permitiera a los usuarios realizar el cierre de obras automático pero esta vez a demanda y no de forma masiva, esto implica que el usuario debe indicar al aplicativo cuál es la obra a ser cerrada con su correspondiente empresa.

Para evaluar si la obra puede ser cerrada o no, se buscan los mismos datos de los mismos negocios que se enumeraron para el cierre masivo en la carga del universo.

Estos son los parámetros de entrada que se deben recibir para realizar un cierre a demanda sin deuda:

- Oficina
- N° SESP
- Empresa
- Obra

Se deben realizar las siguientes validaciones:

- Formato del número de SESP (número de trámite con el que queda registrado el cierre).
- Existencia del número de SESP.
- Disponibilidad de número de SESP.
- Empresa válida.

- Obra válida.
- Relación entre la empresa y la obra válida.

Con estos datos se sugiere una fecha desde y una fecha hasta que se utilizará como rango en el cual se va a efectuar el cierre. La obra puede tener varios periodos de actividad registrados en el BPS, la fecha desde que se calcula es la menor fecha de todos los periodos activos de la obra, la fecha hasta es la mayor fecha de todos los periodos activos de la obra. En caso de que el último periodo no esté cerrado, es decir, que la empresa aún no haya presentado el formulario F9 de solicitud de cierre, este valor es determinado por el usuario del sistema. Luego de calculados estos valores el usuario puede cambiar ese rango de fechas pero se debe corroborar que el periodo nuevo definido este comprendido dentro del periodo calculado.

Los cierres pueden ser definitivos o parciales, esto refiere a si el cierre o culminación de la obra es definitivo o si es momentáneo.

Para que la obra pueda ser cerrada debe pasar por una serie de filtros, estos son:

- La fecha desde debe ser mayor a setiembre de 2006.
- Se verifica que no existan actuaciones en el sistema *Guia-Gdt*.
- Se verifica que existan movimientos en *Cuenta Empresa* para el último mes correspondiente a la fecha hasta.
- Se debe verificar que el saldo en *Cuenta Empresa* sea menor a 3 unidades reajustables.

En caso de que pase por cada uno de estos filtros la obra será cerrada.

Cierre de obras automático con deuda.

Esta funcionalidad fue desarrollada una vez puesto en producción el cierre automático a demanda sin deuda.

A grandes rasgos este tipo de cierre es similar al anterior en el sentido que permite realizar cierres a demanda. Una diferencia con el *sin deuda* es que es posible que la obra tenga un saldo en *Cuenta Empresa* mayor a 3 unidades reajustables.

Además, se puede realizar el cierre para una empresa contratista en particular. Las obras siempre tienen una empresa asociada cumpliendo el rol de *titular*, pero además la obra puede tener asociadas empresas contratistas(o contratadas). Por ejemplo, una empresa se puede encargar de realizar los trabajos de pintura de una casa y otra distinta puede realizar los trabajos de sanitaria. Este cierre permite especificar de forma opcional el número de empresa contratista, número de obra y el número de empresa titular, quedando registrado el cierre para este conjunto de datos.

Dados los siguientes datos:

- Oficina
- Número de Sesp
- Empresa Titular
- Número de Obra

- Artículo 7° (Se puede definir con o sin artículo 7°)
- Empresa Contratista. (Solo se ingresa este datos cuando se optó por artículo 7°)

La oficina es la oficina 226, correspondiente a dónde se realiza la gestión para los cierres de obras. Se deben realizar las mismas validaciones sobre estos parámetros que en el cierre sin deuda. Además se debe validar la existencia de la empresa contratista y que la terna empresa titular-número de obra-empresa contratista esté relacionada.

De la misma manera que en el cierre automático sin deuda, con estos datos se calcula un rango de fechas, estas varían si se ha optado por usar el artículo 7° o no. Esto refiere a la posibilidad de realizar un cierre para una contratista particular.

En caso de que no se haya optado por esa opción las fechas que se sugieren se calculan del mismo modo que en el caso de cierre automático sin deuda.

En caso de que el usuario haya elegido la opción con artículo 7°, o sea, desea realizar el cierre para una empresa contratista en particular, varía el cálculo del periodo obtenido ya que en este entran en juego los datos de la contratista. La fecha desde obtenida es la máxima entre la menor de las fechas de los periodos activos de la obra y la fecha que está registrada en la relación entre la empresa titular, la empresa contratista y la obra. Para la fecha hasta se usa un criterio similar, pero sugiere la fecha mínima entre la mayor de las fechas de los periodos activos de la obra y la fecha que está registrada en la relación empresa titular, empresa contratista y la obra.

En el caso de las fechas hasta, es posible que los valores a los que se les va a calcular el mínimo sean valores nulos, en ese caso no se obtiene ninguna fecha hasta.

Se deben definir además de lo presentado anteriormente, los siguientes datos:

- Tipo de cierre (total o parcial).
- Recargos Condicionados. Se debe definir si incluye recargos condicionados o no, una vez obtenida la deuda a pagar la misma puede estar condicionada a determina forma de pago con el fin de obtener o no beneficios en los recargos. En caso de elegir con recargos condicionados se debe ingresar la fecha de F9.
- Fecha F9, fecha de presentación del formulario F9, es un campo calculado.
- Fecha de recargo personal, representa la fecha a la que se van a calcular las deudas correspondientes.

La fecha de F9 está sujeta a la presentación del formulario F9, en caso de que éste no haya sido presentado se deja este campo en blanco.

La fecha de recargo es un campo calculado y que depende de si se quiere realizar el cierre con recargos condicionados. El modo de calcularlo es el siguiente:

Si se opta por realizar un cierre no condicionado, la fecha de recargo es la fecha correspondiente al último día del mes anterior al día de la ejecución.
Si se opta por realizar un cierre condicionado, la fecha es la menor entre la fecha que mencionábamos anteriormente y la fecha correspondiente al último día del tercer mes posterior a la fecha de presentación del F9.

Los filtros que deben pasar las obras para que pueda ser cerrada son los mismos que para el cierre a demanda con deuda excepto el que hace referencia al saldo en la cuenta, donde ahora se debe verificar que este sea mayor a 3 unidades reajustables.

Esto ha sido un breve estudio del área de negocio.

ANEXO K: Especificación de la gramática.

Como se mencionó en la sección 5.3.2 de las palabras claves de RuleSpeak que definen cada tipo de regla:

- tiene que
- tiene que … si …
- ha de ser calculado como…
- ha de ser entendido como aquel…que/cuyo…
- tiene que ser ejecutado…cuando…
- tiene/está prohibido…
- tiene/está prohibido…si…
- tiene/está autorizado… sólo si…
- no tiene por qué…

Implementamos las siguientes en el Editor:

- tiene que
- tiene que … si …
- ha de ser entendido como …
- tiene que ser ejecutado…cuando…
- tiene/está prohibido…
- tiene/está prohibido…si…

La sentencia "tiene que" fue explicada en la sección 5.3.3, en este anexo veremos el resto.

Tiene que… si…

Toda entidad **tiene que** tener el atributo <|>|<=|>=|==|!= que el atributo de entidad **si** el atributo de entidad es <|>|<=|>=|==|!= que el atributo de entidad.

Al igual que en la regla "tiene que", la entidad es alguna de las entidades descargadas del modelo de Guvnor y atributo algún atributo de esa entidad, los símbolos de comparación marcan las diferentes opciones a elegir.

Ejemplo: Toda obra **tiene que** tener la fecha presentación F9 <= fecha desde si fecha presentación F1 <= fecha hasta.

Las variantes para esta regla son las mismas que las de "tiene que:

Variante 1: Toda entidad **tiene que** tener el atributo **si** el atributo de entidad es <|>|<=|>=|==|!= que el atributo de entidad.
En la que se marca que el atributo no puede ser nulo, por ejemplo: Toda obra **tiene que** tener fecha presentación F9 **si** fecha presentación F1 <= fecha hasta.

Variante 2: Toda entidad **tiene que** tener el atributo <|>|<=|>=|==|!= que atributo **si** el atributo de entidad es <|>|<=|>=|==|!= que el valor. En donde la condición se evalúa con un valor ingresado en la regla que puede ser un texto, un número, una fecha, etc. Por ejemplo: Toda obra **tiene que** tener fecha presentación F9 <= fecha desde **si** el id de obra >= 2000.

Variante 3: Toda entidad **tiene que** tener (el atributo <|>|<=|>=|==|!= que el atributo de entidad) (**y** (atributo es <|>|<=|>=|==|!= que valor|atributo))* **si** atributo es <|>|<=|>=|==|!= que valor|atributo (**y** (atributo es <|>|<=|>=|==|!= que valor|atributo))*, en la que se pueden agregar otras condiciones de atributos y entidades con el agregado "y" la cantidad de veces que se necesite, y lo mismo agregando "o" entre las condiciones.

Variante 4: Toda entidad **tiene que** tener el atributo <|>|<=|>=|==|!= que la suma/multiplicación de atributo (+/* atributo)⁺ **si** atributo es <|>|<=|>=|==|!= que valor|atributo, pudiendo operar con otros atributos o valores.

Estas variantes se pueden conjugar unas con otras formando más variantes.

Tiene prohibido…

Indica que algo no está permitido de forma incondicional.

Toda entidad **tiene prohibido** tener el atributo <|>|<=|>=|==|!= que el atributo de entidad.

Tiene prohibido…si….

Toda entidad **tiene prohibido** tener el atributo <|>|<=|>=|==|!= que el atributo de entidad **si** el atributo de entidad es <|>|<=|>=|==|!= que el atributo de entidad.

Las variantes para estas reglas son las mismas que para la regla "tiene que".

Ha de ser entendido como… si…

Indica que algo ha de ser derivado de cierta forma. Algunas variantes serían:

Variante 1: El atributo de la entidad **ha de ser entendido como** valor**.** En este caso se asigna el valor al atributo de la entidad, podría ser un valor constante o el valor de un atributo.

Variante 2: El atributo de la entidad **ha de ser entendido como** valor **si** el atributo de la entidad <|>|<=|>=|==|!= valor|atributo. Se asigna el valor al atributo de la entidad, podría ser un valor constante o el valor de un atributo como en la primera variante, pero esta asignación se realiza solo si se cumplen las condiciones siguientes.

Variante 3: El atributo de la entidad **ha de ser entendido como** valor **si** el atributo de entidad es <|>|<=|>=|==|!= que la suma/multiplicación de atributo (+/* atributo|valor)⁺.

Ejemplo: El tipoPersona **ha de ser entendido como** "Adulto" **si** la edad de la persona >= 18.

...tiene que ser ejecutado cuando...

Indica que es obligatorio que sea ejecutado en un momento determinado. Algunas variantes son:

Variante 1: Nombre método de entidad **tiene que ser ejecutado cuando** atributo booleano.

Variante 2: Nombre método de entidad **tiene que ser ejecutado cuando** atributo <|>|<=|>=|==|!= valor.

Variante 3: Nombre método de entidad **tiene que ser ejecutado cuando** atributo <|>|<=|>=|==|!= que atributo.

Variante 4: Nombre método de entidad **tiene que ser ejecutado cuando** se cumple que la suma/multiplicación de atributo (+/* atributo)$^{+}$ es <|>|<=|>=|==|!= valor.

...tiene/está autorizado... sólo si...

Una regla del negocio indicando que algo está permitido condicionalmente, indica que algo está permitido solo bajo ciertas condiciones usando “si”.

Toda entidad **tiene/está autorizado** atributo booleano **solo si** atributo es <|>|<=|>=|==|!= que valor.

Por ejemplo: Todo cliente **está autorizado** realizarPedido **sólo si** cuenta de crédito == “true”.

Existen otras variantes como en los casos anteriores.

...no tiene por qué...

Una sentencia de permiso que indica que algo no es obligatorio, de forma incondicional. En entre algunas de las variantes están:

Variante 1: El atributo de entidad <|>|<=|>=|==|!= que valor|atributo **no tiene por qué cumplirse si** atributo ==|!= booleano.

Variante 2: El atributo de entidad <|>|<=|>=|==|!= que valor|atributo **no tiene por qué cumplirse si** atributo es <|>|<=|>=|==|!= que valor.

Variante 3: El atributo de entidad <|>|<=|>=|==|!= que valor|atributo **no tiene por qué cumplirse si** atributo es <|>|<=|>=|==|!= que la suma/multiplicación de atributo (+/* atributo|valor)$^{+}$.

Ejemplo: La duración de obra <= 12 **no tiene por qué** cumplirse si el tipo de obra == “Pública”.

ANEXO L: Manual de usuario Testing Unitario.

En este manual estaremos describiendo el significado de la primera hoja del Excel, así como la forma en que debe llenarse la segunda hoja del mismo por parte del usuario funcional.

La siguiente imagen representa un ejemplo de primera hoja de Excel, la misma representa el modelo de datos con el que se está trabajando. Servirá al usuario para conocer al detalle el modelo y especificará el orden en que deben escribirse los datos en la segunda hoja del Excel.

Objeto	DataCofObra				
Atributo	idObra	Simple	Integer		
Atributo	agencia	Simple	Integer		
Atributo	cantidadDiasTrabajados	Simple	Integer		
Atributo	departamento	Simple	Integer		
Atributo	fechaDesdeObra	Simple	Date		
Atributo	fechaHastaObra	Simple	Date		
Atributo	fechaPresF9	Simple	Date		
Atributo	fechaInicioF1	Simple	Date		
Atributo	numeroEmpresa	Simple	String		
Atributo	numeroObra	Simple	String		
Atributo	saldoCuenta	Simple	Integer		
Atributo	padron	Simple	String		
Atributo	unidad	Simple	String		
Atributo	duracionObra	Simple	Integer		
Atributo	dataObraPadronDepartamento	Entidad	DataObraPadronDepartamento		
Atributo	dataObraActuaciones	ColeccionEn	DataObraActuacion		
Objeto	DataObraPadronDepartamento				
Atributo	codDepartamento	Simple	Integer		
Atributo	nombreDepartamento	Simple	String		
Objeto	DataObraActuacion				
Atributo	idObraAct	Simple	Integer		
Atributo	estado	Simple	String		
Atributo	numeroActuacion	Simple	Integer		
Atributo	tipoEstado	Simple	String		

Figura 77: Modelo de datos del Excel.

Esta primera hoja se genera de forma automática al abrir el Editor y debe leerse de la siguiente forma:

Si la primer entrada correspondiente a la primer columna de una fila contiene la palabra "*Objeto*" significa que se va a comenzar a describir una nueva entidad, en la segunda columna correspondiente a esa fila se encontrara el nombre de dicha entidad. En las sucesivas filas las entradas correspondientes a la primera columna contendrán la palabra "*Atributo*".

Esto seguirá así hasta que se enumeren todos los atributos de la entidad que se ha comenzado a describir. Cuando una fila corresponde a un atributo la entrada de la misma correspondiente a la segunda columna contiene el nombre del atributo de la entidad que se está describiendo.

La entrada correspondiente a la tercer columna contiene el tipo de atributo, existe tres tipos, los atributos "*Simple*" son atributos cuyo tipo de datos es simple, con esto nos referimos a: *Integer*, *String* y *Date*, los atributos "*Entidad*" cuyo tipo de datos se corresponde con una entidad, esto sería una asociación en un modelo de datos y finalmente el tercer tipo que es *ColeccionEntidad* que indica que el atributo es una colección.

La entrada correspondiente a la cuarta columna de un atributo contiene el tipo de datos propiamente dicho del atributo, por ejemplo, si el atributo es de tipo "*Simple*" puede contener Integer, si el atributo es de tipo "*Entidad*" puede contener por ejemplo el tipo *DataObraPadronDepartamento* y finalmente si el tipo del atributo es "*ColeccionEntidad*" entonces la cuarta columna podría ser por ejemplo *DataObraActuacion*.

Una vez que se han agregado para una entidad todos los atributos se comienza a describir la siguiente entidad, en el ejemplo que mostrábamos, primero se describe todo lo referente a la entidad DataCofObra y luego se continúa con la descripción de la entidad DataObraPadronDepartamento a partir de la fila 17.

La primera hoja será utilizada como referencia para escribir los datos del universo de la segunda hoja.

Mostramos una captura de una pequeña instancia de un objeto, que se encuentra en la segunda hoja del Excel, para explicar cómo se debe llenar esta hoja.

ent	DataCorrida	1		02/03/2012			01/01/2012	31/12/2012		1	2-2	
ref	DataCofObra	1	100	130			5	01/04/2012	20/12/2012	04/05/2012	10/05/2012	22/12/2012
ref	DataObraActuacion	1	homologado	1	integral		1					
ref	DataObraActuacion	2	homologado	2	integral		1					
ref	DataObraSesp	1		VD5	226	1						
ref	DataObraSesp	2		GR7	227	1						

Figura 78: Ejemplo de instanciación de un modelo de datos.

Cada vez que se comienza a describir una nueva instancia de una entidad se debe comenzar escribiéndola en una nueva fila y la entrada de la primera columna debe contener la palabra *ent*.

El orden en que deben aparecen los datos de los atributos para cada entidad se debe corresponder con el orden en que aparecen descritos en la primera hoja. Por ejemplo la entidad DataCorrida en la primera hoja está representada de la siguiente forma:

Objeto	DataCorrida			
Atributo	idCorrida	Simple	Integer	
Atributo	estado	Simple	Integer	
Atributo	fechaUltA	Simple	Date	
Atributo	numeroSe	Simple	String	
Atributo	oficina	Simple	Integer	
Atributo	fechaDesc	Simple	Date	
Atributo	fechaHast	Simple	Date	
Atributo	usuarioUl	Simple	String	
Atributo	conF9	Simple	Integer	
Atributo	dataCofOb	Coleccion	DataCofObra	

Figura 79: Definición de una Entidad.

Entonces podemos ver el mapeo:

idCorrida = 1;
estado = null;
fechaUltAct = 02/03/2012

Se debe continuar esta lectura de forma sucesiva.

Por otro lado, si el dato que se está ingresando se corresponde con un tipo de atributo *Simple* entonces el tipo de datos de la celda debe ser acorde al tipo del atributo, por ejemplo, si el atributo en el modelo es un *String* entonces el tipo de datos de la celda debe ser *Texto*.
Hemos explicado que hacer en caso de que el tipo de atributo sea *Simple*, pues bien, en caso de que el tipo sea *Entidad* esto significa que existe una asociación, en este caso lo que se hace es escribir en la celda el número de fila donde comienza a describirse la entidad asociada, la única diferencia de la descripción de esta entidad es que la entrada correspondiente a la primer columna deberá decir *ref* y no *ent* como en el caso de la anterior.
Nos queda describir que sucede en caso de que el tipo del atributo sea *ColeccionEntidad*, en este caso lo que se hace es escribir el rango de filas donde se encuentran las instancias de la Colección en el formato *instanciaInicial-instanciaFinal*, por ejemplo, *2-5*.

En la sección 5.5.3 explicamos cual es el resultado que se produce cuando una regla es ejecutada correctamente, describimos en los siguientes puntos cuales son los mensajes de error que se pueden obtener y porqué suceden.

- Mensaje: "Ha ocurrido un error mientras se identificaba la regla a testear."

Este mensaje indica que el Plugin no ha podido detectar la regla sobre la cual se está realizando clic derecho.

- Mensaje: "Ha ocurrido un error mientras se respalda la regla para ser ejecutada.";

Indica que se dio un error mientras se estaba copiando la regla a un directorio local del Editor, donde será modificada para su procesamiento, o se dio un error mientras se realizaba dicha modificación.

- Mensaje: "Ha ocurrido un error mientras se intenta levantar los datos del Excel, verifique que el mismo existe.";

Es posible que se haya eliminado el Excel generado en el momento de abrir el Editor de Reglas de Negocio erróneamente, en ese caso se desplegará dicho mensaje.

- Mensaje: "Ha ocurrido un error mientras se intenta levantar los datos del Excel, verifique que exista una hoja con datos para su modelo.";

Cuando un usuario intenta ejecutar un test sobre una regla pero aún no se han cargado datos en el Excel, entonces aparece el mensaje anterior.

- Mensaje: Ha ocurrido un error mientras se intenta levantar los datos del Excel, revisar la fila: i, columna j.

Este error se obtiene cuando el usuario ingresa un dato en el Excel y la celda tiene un tipo de datos asociado distinto al tipo de datos del modelo.

- Mensaje: "Ha ocurrido un error mientras se ejecutaba la regla.";

En este caso se ha producido un error en Drools mientras se procesa la regla.

- Mensaje: "Ha ocurrido un error mientras se escribe el resultado, asegúrese de que el Excel esté cerrado al momento de la ejecución de las pruebas.";

Muchas veces sucede que no se cierra el Excel a la hora de ejecutar la prueba por lo tanto no se puede escribir el resultado de la misma, una vez cerrada la planilla puede volver a ejecutarse el caso.