Universidad Austral de Chile

Facultad de Ciencias de la Ingeniería

Escuela de Ingeniería Civil en Informática

DESARROLLO E IMPLEMENTACIÓN DE UN PROTOTIPO DE SISTEMA DE ADMINISTRACIÓN DE REGLAS DE NEGOCIO PARA TELEFÓNICA DEL SUR.

Tesis para optar al título de:
Ingeniero Civil en Informática

Profesor Patrocinante:
Sr. Rodrigo Ariel Soriano Oyarzún
Ingeniero Civil en Informática

Profesor Co-Patrocinante:
Sr. Juan Pablo Salazar Fernández
Ingeniero Civil en Informática
Magíster en Administración de Empresas.

Profesor Informante:
Sra. Gladys Myriam Mansilla Gómez.
Ingeniero Matemático,
Analista de Sistemas,
Magíster en Estadística
D.E.A Teoría de la Señal y Comunicaciones

CRISTIAN ROLANDO ALIANTE ZAMBRANO
EDGARDO ALEJANDRO IBÁÑEZ ORTIZ

VALDIVIA – CHILE
2008

Valdivia, 18 de Noviembre de 2008

De: Rodrigo Soriano
Patrocinante

A: Juan Pablo Salazar Fernández
Director
Escuela de Ingeniería Civil en Informática

Ref: Calificación proyecto de título

De mi consideración:

Habiendo revisado el trabajo de titulación **"Desarrollo e Implementación de un Prototipo de Sistema de Administración de Reglas de Negocio para Telefónica del Sur",** presentado por los alumnos sres Cristian Rolando Aliante Zambrano y Edgardo Alejandro Ibáñez Ortiz, mi evaluación del mismo es la siguiente:

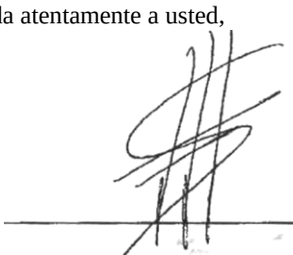
Nota: 6,9 (seis coma nueve)

Fundamento de la nota:

Me parece un proyecto muy innovador y desarrollado correctamente.

Aspecto	**Evaluación**
Cumplimiento de objetivos	7,0
Satisfacción de alguna necesidad	7,0
Aplicación del método científico	7,0
Interpretación de los datos y obtención de conclusiones	6,5
Originalidad	7,0
Aplicación de criterios de análisis y diseño	6,5
Perspectivas del trabajo	7,0
Coherencia y rigurosidad lógica	7,0
Precisión del lenguaje técnico	6,8

Sin otro particular, saluda atentamente a usted,



Rodrigo Soriano O
Subgerente de Informática
Telefónica del Sur S.A.

Valdivia, 05 de noviembre de 2008

De: Juan Pablo Salazar Fernández
Académico
Instituto de Informática

Ref: Calificación proyecto de título

De mi consideración:

Habiendo revisado el trabajo de titulación "**Desarrollo e Implementación de un Prototipo de Sistema de Administración de Reglas de Negocio para Telefónica del Sur**", presentado por los estudiantes sres. Cristian Rolando Aliante Zambrano y Edgardo Alejandro Ibafiez Ortiz, mi evaluación del mismo es la siguiente:

Nota: 6,9 (seis, coma nueve).

Fundamento de la nota:

El presente trabajo de titulación se planteó como objetivo el desarrollo de un sistema de administración de reglas de negocio que considere control y gestión centralizada y que permita optimizar los tiempos de mantención y desarrollo de los sistemas informáticos de la empresa Telefónica del Sur, sin afectar de manera apreciable el tiempo de respuesta de éstos.

Este objetivo fue logrado a cabalidad, ya que se hizo una revisión acuciosa de las tecnologías que permitían abordar el proyecto y de la problemática de la empresa, se seleccionó e implemento un prototipo que utilizara reglas de negocio reales, se hizo una propuesta de rediseño del flujo de trabajo y se validó el prototipo con profesionales de la empresa.

Aspecto	**Evaluación**
Cumplimiento de objetivos	7,0
Satisfacción de alguna necesidad	7,0
Aplicación de metodologías adecuadas	7,0
Interpretación de los datos y obtención de conclusiones	7,0
Originalidad	7,0
Aplicación de criterios de análisis y diseño	7,0
Perspectivas del trabajo	7,0
Coherencia y rigurosidad lógica	7,0
Precisión del lenguaje técnico	6,5

Sin otro particular, saluda atentamente a usted,

Juan Pablo Salazar Fernández
Académico
Instituto de Informática

Valdivia, 12 de diciembre del 2008

De: Gladys Mansilla Gómez
Informante

A: Juan Pablo Salazar Fernández
Director

Escuela de Ingeniería Civil en Informática

Ref: Calificación proyecto de título

De mi consideración:

Habiendo revisado el trabajo de titulación **"DESARROLLO E IMPLEMENTACIÓN DE UN PROTOTIPO DE SISTEMA DE ADMINISTRACIÓN DE REGLAS DE NEGOCIO PARA TELEFÓNICA DEL SUR",** presentado por los alumnos srs. **CRISTIAN ROLANDO ALIANTE ZAMBRANO**
EDGARDO ALEJANDRO IBÁÑEZ ORTIZ

, mi evaluación del mismo es la siguiente:

Nota: 6.9 (seis coma nueve).

Fundamento de la nota:

Es un trabajo muy útil para las empresas hoy en día.

Aspecto	**Evaluación**
Cumplimiento de objetivos	7
Satisfacción de alguna necesidad	7
Aplicación del método científico	7
Interpretación de los datos y obtención de conclusiones	7
Originalidad	7
Aplicación de criterios de análisis y diseño	7
Perspectivas del trabajo	7
Coherencia y rigurosidad lógica	7
Precisión del lenguaje técnico	6.3

Sin otro particular, saluda atentamente a usted,

Gladys Mansilla Gomez
Docente
Instituto de informática

Agradecimientos

La llegada a este momento de mi vida se la debo sin duda a mi familia; Luis, Mirna y Angélica, a quienes agradezco todo el amor, esfuerzo y apoyo que me han entregado, sin el cual no hubiese podido llegar a ser la persona que soy.

A mis tíos y abuelos, quienes siempre me han apoyado de manera incondicional en este largo recorrido.

A Ana, quien ha sido mi compañera durante gran parte del periodo de estudios y la nueva vida laboral que estamos comenzando juntos.

Edgardo.

Al llegar al final de mi etapa universitaria, la que termina con este trabajo, quiero agradecer a Dios quien ha llenado mi vida espiritualmente a lo largo de estos años, a mis padres, Carlos y Nancy, quienes han sido pilares fundamentales en mi formación y todo lo que soy se lo debo a ellos. A mis hermanos quienes siempre han confiado en mí. Gracias Familia por su amor incondicional.

A mi hija Laurita, quien ha bendecido y llenado mi vida de felicidad y me da fuerzas para el día a día. Y a todos quienes han puesto un granito en esta etapa de mi vida…Muchas gracias.

Cristian.

Agradecemos a nuestros profesores, que nos traspasaron sus conocimientos, experiencias y valores profesionales que utilizamos día a día.

A quienes patrocinaron este proyecto de Tesis, la guía y apoyo de Rodrigo, Juan Pablo, y Héctor, a nuestros colegas de Telefónica del Sur, con quienes compartimos gratos momentos y más de alguna rencilla futbolera.

A nuestros compañeros y amigos de largas jornadas de biblioteca, laboratorios, café y pan con mortadela, quienes siguen estando ahí cuando hemos necesitado de su ayuda.

INDICE

INDICE DE TABLAS

INDICE DE FIGURAS

SINTESIS

Las reglas de negocio son fundamentales para las organizaciones actuales, ya que en ellas está contenida la lógica de negocio de la empresa, la cual debe adaptarse a los cambios de mercado y a las estrategias corporativas con gran facilidad, de tal forma de mantener la competitividad y las oportunidades de negocio.

En este proyecto de tesis se investigó algunas de las más populares alternativas existentes en el mercado para el manejo de reglas de negocio, tomando en cuenta las necesidades de la empresa en relación a costos, características y usabilidad de la herramienta a utilizar, además de interfaces de comunicación que permitan la interconexion entre los sistemas de la compañía y las reglas de negocio.

Una vez elegido el producto adecuado se procedió a su implementación, con el fin de separar las reglas de negocio del sistema de tasación de llamadas locales, otorgando un mayor control de las reglas de negocio, permitiendo crearlas y modificarlas sin necesidad de bajar los sistemas de producción, además de dar la posibilidad de que el personal del área de facturación pueda realizar estas operaciones, aliviando la carga de trabajo del personal del área de informática.

El prototipo fue desarrollado sobre la plataforma BRMS ("*Business Rules Management System*.") de JBoss, que por medio de servicios Web es alimentada desde la base de datos con la información necesaria para ejecutar las reglas de tasación de llamadas locales.

La evaluación de este proyecto de tesis se llevo a cabo mediante la comparación de los resultados obtenidos por las reglas de negocio implementadas versus las reglas que actualmente están en los sistemas, además de los tiempos de respuesta obtenidos.

ABSTRACT

The business rules are essential for today organizations because the business logic of the company is contained in them. The company must adapt to changing marketplace and corporate strategies in an easy way, so as to maintain competitiveness and business opportunities.

In this thesis project was investigated some of the most popular alternatives in the marketplace for managing business rules, taking account the needs of the company in relation to costs, and usability features of the tool to use, also the communication interface between the company systems and the business rules engine.

Once chosen the right product, we proceeded to its implementation, in order to separate the business rules for the local calls billing system, giving to the company greater control of the business rules, allowing it to create and modify them without stopping production systems, in addition to the possibility that the billing area can perform these operations, making easier the workload of the information technology area.

The prototype was developed on the BRMS (“Business Rules Management System") platform of JBoss. It is fed through Web services from the database with information needed to execute the business rules of local calls billing.

The project evaluation is based in the comparation between the results of business rules engine and the actually billing system. Also is important the response time of the solution implemented, hence, this parameter will be considered in the final analysis

CAPITULO I. INTRODUCCION

1.1 Descripción del Capitulo

El presente capitulo tiene por objetivo dar una pequeña introducción a lo que se conoce como “Reglas de Negocio” y lo que soluciona, dar a conocer los objetivos generales y específicos a cumplir en este proyecto y proporcionar el marco teórico, el cual nos brinda el conocimiento necesario como base para empezar a desarrollar el proyecto de tesis..

1.2 Antecedentes generales

Las Reglas de Negocio se encuentran siempre presentes en una organización, bien de manera explícita (una política de sueldos, el horario laboral, el descuento a aplicar en función de las condiciones de la venta, etc.) o de manera implícita no expresada (el trato cortés con los clientes, la responsabilidad del supervisor sobre sus supervisados, etc.) siempre implicando la participación directa o indirecta de personas.

Sin embargo, el término Reglas de Negocio queda reservado únicamente para aquellas reglas que revisten carácter explícito y que pueden ser y son, expresadas de manera entendible, registradas, localizables y modificables.

En la mayor parte de los sistemas de hoy en día las reglas de negocio están inmersas en los códigos de los programas, lo que implica varias desventajas:

- Falta de claridad en las reglas del negocio, ya que las mismas están “embebidas” entre varios módulos y códigos entendibles sólo por los programadores.
- Cada vez que se cambia una regla es necesario recompilar el código de la aplicación
- Suele ser dificultoso localizar el lugar exacto donde una regla de negocio se encuentra
- Las reglas no pueden ser gestionadas por la gente del negocio.

- Imposibilidad de compartir las reglas de negocio con otras empresas, ya sean proveedores o clientes.

Estas condiciones complican el quehacer de la empresa, reduciendo su capacidad de reacción ante cambios del mercado y haciendo ineficientes sus procesos. Para solucionar este problema, surge la opción de utilizar sistemas de administración de reglas de negocio, los cuales proveen las herramientas necesarias para manejar eficientemente la cambiante lógica del negocio.

1.3 Objetivos Generales y Específicos

1.3.1 Objetivos Generales.

Desarrollar e implementar un sistema de administración de reglas de negocio, que considere control y gestión centralizada, optimizando los tiempos de mantención y desarrollo de los sistemas de Telefónica del Sur.

1.3.2 Objetivos Específicos.

1. Comprender y describir el estado del arte de la administración de reglas de negocio y las distintas herramientas que permiten su gestión.
2. Definir los requisitos y tecnologías a utilizar para administrar las reglas del sistema de tasación.
3. Diseñar e implementar una solución que permita administrar las reglas de negocio del sistema de tasación.
4. Diseñar e implementar interfaces comunicación desde y hacia los sistemas de Telefónica del Sur que se adapten a nuevos requerimientos o formas de comunicación, utilizando protocolos abiertos.
5. Evaluar la solución mediante métricas de conectividad, tiempo de respuesta y proporción de reglas administradas en un ámbito determinado;

1.4 Marco Teórico

1.4.1 Sistema basado en reglas

Para escoger de manera informada el producto que manejará las reglas de negocio, primero es necesario conocer los conceptos que hay detrás de los sistemas basados en reglas, de que manera funcionan y cuales son sus tipos y caracteristicas que cada uno tiene.

1.4.1.1 Definición de reglas de negocio.

Reglas de Negocio son los elementos individuales que permiten ser definidos, delimitados y expresados de forma inequívoca y que en su conjunto componen el marco estructural, la política, la estrategia y la operativa de una empresa u organización.

Las Reglas de Negocio deben definirse y mantenerse de manera independiente de los modelos y los procesos de la empresa, ya que su misión es definir de manera granular las políticas y modos de operar.

Las reglas de negocio se pueden clasificar según su función en:

- Operativas: llamadas así porque su ejecución implica la vinculación personal y que por tanto pueden ser violadas. Un ejemplo de Regla Operativa es: Los obreros deben llevar casco. Evidentemente, si un obrero no se pone el casco, está violando esta regla.
- Estructurales: son las que definen condiciones y tratamientos y que, al no depender de la acción de las personas, no pueden ser violadas. Un ejemplo de regla estructural es: Se considera cliente preferencial al que haya comprado más de $ 200.000 en el último año.

Por su naturaleza, las Reglas de Negocio pueden ser: Textuales (interpretables) y Mecánicas (automatizables). Las reglas Textuales muestran su contenido mediante expresiones de texto y normalmente aceptan algo de interpretación. Por el contrario, las reglas Mecánicas se expresan mediante fórmulas, tablas o expresiones matemáticas y por tanto pueden automatizarse. [TRI]

Existen organizaciones dedicadas al estudio y establecimiento de estándares para la unificación de las Reglas de Negocio. En ese sentido, publicaciones como el SBVR (“*Semantics of Business Vocabulary & Business Rules*”) [BRG] del OMG (“*Object Management Group*”) [OMG] han hecho importantes esfuerzos en favor de lograr una estandarización de esta área.

1.4.1.2 Motores de Reglas

El objetivo del motor de reglas de negocio es proporcionar una infraestructura desacoplada del código fuente, formando un repositorio al cual el sistema pide servicios de decisión.

En dicho repositorio, residen las reglas del sistema, y que son invocadas por medio de un protocolo de aplicaciones. Además, se puede construir un sistema para que el usuario pueda definir sus reglas y cambiarlas, esto permite que los desarrolladores no sean los principales responsables en cambiar reglas ya hechas y, por tanto, ya no tienen la "responsabilidad" de liberar el sistema.

En general, un motor de reglas consta de las siguientes partes:

- Memoria de trabajo (donde se almacenan los hechos)
- Base de reglas
- Motor de inferencia
- Interfaz de administración de reglas

1.4.1.3 Sistemas de Producción

Un sistema de producción proporciona una estructura que facilita la descripción y la ejecución de un proceso de búsqueda [RIC01]. En general, un sistema de producción se compone de cuatro partes principales:

- Un conjunto de reglas donde el lado izquierdo determina si la regla es o no aplicable y un lado derecho donde se describe la operación a realizar cuando la regla se cumple.

- Una base de conocimientos que gatilla la ejecución del set de reglas, esta base de conocimientos puede ser permanente o puede permanecer de manera temporal, sólo para la operación de decisión actual.
- Una estrategia de control que especifica el orden en el que las reglas son procesadas, y la forma de resolver los conflictos que pueden aparecer cuando varias reglas coinciden simultáneamente.
- Un mecanismo que se encarga de ir aplicando las reglas.

Existen muchos sistemas enfocados a distintas tareas que se enmarcan dentro de los sistemas de producción, y según la definición del apartado anterior los motores de reglas de negocio encajan perfectamente en esta categoría.

1.4.1.4 El Motor de Inferencia

Una regla es una proposición lógica que relaciona dos o más objetos e incluye dos partes, la condición y la conclusión. Cada una de estas partes consiste en una expresión lógica con una o más afirmaciones objeto-valor conectadas mediante los operadores lógicos "*and, or y not*". Una regla se escribe normalmente como "Si condición, entonces conclusión".

El motor de inferencia es el corazón de un sistema de producción, es el que alimentado por la base de conocimientos, construye dinámicamente el razonamiento, decidiendo qué reglas se activan y en qué orden. [CHA88].

En un motor de reglas hay dos tipos de elementos: los hechos, que son el conocimiento que se tiene de un problema específico, y el set de reglas, que representan la lógica a seguir para un determinado estado de los hechos. El motor de inferencia usa ambos para obtener nuevas conclusiones o hechos. Por ejemplo, si la condición de una regla es cierta, entonces la conclusión de la regla debe ser también cierta. De esta forma la cantidad de hechos se incrementa debido a la incorporación de las nuevas conclusiones. Por ello, tanto los hechos iniciales o datos de partida como las conclusiones derivadas de ellos forman parte de los hechos o datos de que se dispone en un instante dado.

Un motor de inferencia posee un ciclo base mediante el cual se toman las decisiones, este ciclo posee cuatro fases:

- Fase de selección: En esta fase se escoge un conjunto de reglas por sobre las demás para ser analizadas, esta restricción permite una economía de tiempo en las fases siguientes.
- Fase de filtrado: En esta fase el motor de inferencia compara la condición de las reglas con el conjunto de hechos disponibles y selecciona el conjunto de reglas aplicables.
- Fase de resolución: En esta fase se decide qué regla será utilizada en base a alguna estrategia que puede ser simple, como la regla menos utilizada o la menos compleja, o alguna otra estrategia que seleccione la que tenga mayor relación con el contexto en que se está trabajando, por nombrar algunas.
- Fase de ejecución: En esta fase se utiliza la regla elegida en la fase anterior y se ejecuta su conclusión. Esta acción generalmente agrega nuevos hechos, pero también podría llamar a un procedimiento externo, o solicitar interacción con el usuario.

Para obtener conclusiones, los motores utilizan diferentes tipos de reglas y estrategias de inferencia y control. En los apartados siguientes se discutirán las reglas de inferencia:

- Modus Ponens
- Modus Tollens

Y las estrategias de inferencia:

- Encadenamiento de reglas
- Encadenamiento de reglas orientado a un objetivo

1.4.1.5 Modus Ponens y Modus Tollens

El "*Modus Ponens"* se utiliza para obtener conclusiones simples, en donde, si se afirma la condición (antecedente) de la regla entonces se afirma la conclusión (consecuente), formando parte del conocimiento. A modo de ejemplo, si tenemos la regla, "Si A es cierto, entonces B es cierto" y que se sabe además que "A es cierto". Entonces la regla *"Modus Ponens"* concluye que "B es cierto".

La regla de inferencia *"Modus Tollens"* se utiliza también para obtener conclusiones simples, en este caso se examina la conclusión y si es falsa, se concluye que la condición también es falsa. Por ejemplo, si tenemos la regla, "Si A es cierto, entonces B es cierto" pero se sabe que "B es falso." Entonces, utilizando la regla *"Modus Ponens"* no se puede obtener ninguna conclusión pero la regla *"Modus Tollens"* concluye que "A es falso". [CHA88]

1.4.1.6 Encadenamiento de Reglas

Es utilizado para obtener conclusiones compuestas. Esta estrategia de inferencia puede utilizarse cuando las condiciones de ciertas reglas coinciden con las conclusiones de otras. Cuando se encadenan las reglas, los hechos pueden utilizarse para dar lugar a nuevos hechos. Esto se repite sucesivamente hasta que no pueden obtenerse más conclusiones. El tiempo que consume este proceso hasta su terminación depende, por una parte, de los hechos conocidos, y, por otra, de las reglas que se activan.

Este algoritmo puede ser implementado de dos formas [GIA01]

- Encadenamiento hacia adelante: es el razonamiento desde los hechos hacia las conclusiones que resulten de ellos. Por ejemplo, si se ve que está lloviendo antes de salir a trabajar (el hecho), entonces se debe llevar un paraguas (conclusión). Técnicamente esta estrategia busca las reglas cuya condición sea verificada por los hechos de su base de hechos; si existe esa regla entonces se aplica. De la misma forma buscará otra regla, la aplicará y así sucesivamente hasta que el hecho buscado haya sido deducido o hasta que no hayan más reglas, se deben contar con hechos iniciales para empezar con el razonamiento.
- Encadenamiento hacia atrás: es el razonamiento en reversa, desde una hipótesis habrá de comprobar una posible conclusión, a los hechos que la sustentan. Por ejemplo, si no hemos mirado hacia afuera y alguien entra a la casa con paraguas y los zapatos mojados, la hipótesis sería que está lloviendo, y si para apoyar la hipótesis le consultamos a la persona si de verdad está lloviendo y la respuesta es sí, entonces la hipótesis es verdadera y se convierte en un hecho. La hipótesis a demostrar es utilizando los hechos y las reglas de la base de conocimientos. Si el hecho a demostrar ya forma parte de la base de conocimiento el proceso termina, si no busca en dicha

base una regla que concluya dicho objetivo, pasando a ser los sub-objetivos a demostrar las condiciones de las reglas y así recursivamente.

1.4.1.7 Encadenamiento de Reglas Orientado a un Objetivo

El algoritmo de encadenamiento de reglas orientado a un objetivo requiere que el usuario seleccione, en primer lugar, una variable o nodo objetivo; entonces el algoritmo navega a través de las reglas en búsqueda de una conclusión para el nodo objetivo. Si no se obtiene ninguna conclusión con la información existente, entonces el algoritmo fuerza a preguntar al usuario en busca de nueva información sobre los elementos que son relevantes para obtener información sobre el objetivo.

Las estrategias de encadenamiento de reglas se utilizan en problemas en los que algunos hechos (por ejemplo, síntomas) se dan por conocidos y se buscan algunas conclusiones (por ejemplo, enfermedades). Por el contrario, las estrategias de encadenamiento de reglas orientadas a un objetivo se utilizan en problemas en los que se dan algunos objetivos (enfermedades) y se buscan los hechos (síntomas) para que estas sean posibles.

1.4.1.8 Control de Coherencia

El control de coherencia tiene por función la de prevenir la entrada de información incoherente en la base de conocimiento. Si la base de conocimiento contiene información inconsistente (por ejemplo, reglas y/o hechos), es muy probable que el sistema no se comporte como se esperaba y obtenga conclusiones absurdas.

El objetivo del control de la coherencia consiste en:

- Ayudar al usuario a no dar hechos inconsistentes, por ejemplo, dándole al usuario las restricciones que debe satisfacer la información demandada.
- Evitar que entre en la base de conocimiento cualquier tipo de hechos inconsistente o contradictorio, generado a partir de la misma ejecución de las reglas.

El control de la coherencia debe hacerse controlando la coherencia de las reglas y la de los hechos.

1.4.1.9 Coherencia de Reglas

Un conjunto de reglas es coherente si existe, al menos, un conjunto de valores de todos los objetos que produzcan conclusiones no contradictorias. Por lo tanto, un conjunto coherente de reglas no tiene por qué producir conclusiones no contradictorias para todos los posibles conjuntos de valores de los objetos.

Un conjunto de reglas puede ser coherente, aun si algunos conjuntos de valores puedan producir conclusiones inconsistentes. Estos conjuntos de valores se llaman valores no factibles.

1.4.1.10 Coherencia de hechos

Los datos suministrados por los usuarios deben ser también consistentes entre sí y con el conjunto de reglas de la base de datos. Por ello, el sistema no debe aceptar hechos que contradigan el conjunto de reglas y/o el conjunto de hechos existente en cada instante del proceso. El sistema debe también comprobar si existe o no una solución factible e informar al usuario en consecuencia.

1.4.1.11 Algoritmo rete

El Algoritmo Rete es ampliamente reconocido como el algoritmo más eficiente para la implementación de sistemas de producción. El algoritmo fue originalmente desarrollado en la Universidad Carnegie Mellon por Charles L. Forgy en el año1979, evolucionado paulatinamente hasta la fecha. [GIA]

Este algoritmo se basa en la comparación de valores (denominados “hechos”) mediante patrones de comparación (denominadas “reglas”) en un sistema de producción (llamado “motor de reglas”). En este caso, el sistema de producción está formado por una o más condiciones y un conjunto de acciones, las cuales deben ser ejecutadas cuando las condiciones son verdaderas. Las características más importantes son.

- Reduce o elimina ciertos tipos de redundancia a través del uso compartido de nodos de la red.
- Eficiencia en la eliminación de elementos de memoria cuando los hechos son anulados de la memoria temporal.
- Rete es el único algoritmo para sistemas cuya eficiencia no depende del número de reglas.

1.4.1.12 Ventajas y desventajas del uso de un motor de reglas de negocio.

Como en todas las cosas, el uso de un motor de reglas tiene ventajas y desventajas, dependiendo de la situación en que se utilice, ya que no todos los problemas tienen una solución adecuada con el uso de esta tecnología. A continuación se enumeran algunas.

Ventajas:

- Usan lenguajes declarativos, lo que permite modelar situaciones complejas con un código menos extenso y más entendible que su contraparte procedural.
- El conocimiento se concentra en un solo lugar y ya no es sólo propiedad de la persona encargada, sino que está a disposición de cualquier persona.
- Tienen una buena escalabilidad ya que la curva de tiempo de respuesta tiene una tendencia logarítmica a medida que el número de reglas crece.
- Gran parte de los productos disponibles en el mercado poseen herramientas como lenguajes específicos de dominio, especificación gráfica de flujo de reglas y otras que permiten lograr el entendimiento del código por gente que no necesariamente posee los conocimientos técnicos para hacerlo.

Desventajas:

- En aplicaciones donde las reglas son de carácter técnico y no tienen mayor variación en el tiempo, no es justificable el uso de un motor de reglas.
- Cuando la cantidad de reglas es baja, el uso de un motor de reglas es menos eficiente que el uso de un lenguaje procedural.
- El uso de un motor de reglas necesita de máquinas con alto poder de procesamiento.

- Es una tecnología relativamente nueva, por lo que existen muchas empresas que aparecen y desaparecen rápidamente en el mercado, dejando productos sin soporte.
- Existe poco conocimiento de la tecnología, por lo que la mano de obra se torna escasa y por consiguiente de un costo mayor.

Es importante considerar los aspectos mencionados antes de embarcarse en la implementación de un motor de reglas, ya que frecuentemente se cae en el vicio de utilizarlo para resolver todos los problemas existentes, donde, en muchos casos, es más eficiente una solución alternativa.

1.4.1.13 Potencialidades y uso actual de un motor de reglas.

1.4.1.13.1 Reglas de negocio y BPM[1]

Los sistemas de administración de procesos de negocio o BPMS[2] han tenido una gran aceptación a la hora de automatizar procesos, estos sistmas permiten tener una mejor visibilidad y control sobre la forma en que se realizan los procesos clave en la organización. Los sistemas BPM se encargan de orquestar un conjunto de servicios de manera de llevar a cabo un proceso, en esta coordinación se toman distintas decisiones, situación en la cual se ha optado por utilizar motores de reglas que permitan automatizar esta acción.

El uso de ambas tecnologías proporciona ventajas como la estandarización y centralización de las decisiones además de la capacidad de adaptación de la organización a los cambios del mercado.

Una de las áreas donde más se utiliza esta combinación es en el sector financiero, en donde aportan flexibilidad y manejo de múltiples variables en la concesión de créditos y gestión del riesgo de los clientes, tema clave en la rentabilidad de las empresas de este sector.

1.4.1.13.2 Reglas de negocio e Inteligencia de Negocios

[1] BPM: Bussines Process Managment, Administración de Procesos de Negocio

[2] BPMS: Bussines Process Managment System, Sistema de Administración de Procesos de Negocio

Los sistemas de reglas de negocio e inteligencia de negocio a menudo se utilizan a distintos niveles dentro de la organización. Por un lado, las reglas de negocio asociadas a los procesos de la organización y por otro lado la inteligencia de negocios orientada a la comprensión de los datos transaccionales. Sin embargo, actualmente se ha dado una combinación de ambas logrando beneficios que no se tenían considerados hasta el momento.

> Al trabajar unidas ambas tecnologías se pueden agregar nuevas capacidades a la inteligencia de negocios tradicional. Así, utilizando inteligencia de negocios basada en un BRMS[3], podrían modificarse las políticas para cambiar la asociación de determinadas reglas y contarse con mejores elementos para analizar el negocio. Por ejemplo, si la aprobación de una solicitud está asociada al valor del dólar, al cambiarse esa regla y su asociación, podría irse a la base de datos histórica de las transacciones de un período determinado y conocer cuánto influyó esa variable en la demanda real y las aprobaciones que se cursaron. [CIE]

Un BRMS aporta una capacidad muy importante que no posee tradicionalmente la inteligencia de negocios. Cuando el motor de reglas toma una decisión, éste conoce claramente en qué se basó para tomarla, lo que añade importante información al historial de las decisiones tomadas en la empresa, información que favorece el análisis de futuras decisiones.

Esta integración permite a las empresas ver cómo son en realidad tomadas las decisiones y no cómo ellos creen que son tomadas.

1.4.2 Descripción de estándares y protocolos de comunicación.

El sistema de administración de reglas deberá ser capaz de interactuar con sistemas implementados en distintas tecnologías, por esto se describirán algunas de las más importantes con el fin de tener un mayor conocimiento a la hora de decidir cual se ocupará.

1.4.2.1 Sockets

[3] Bussines Rules Managment System, Sistema de administración de reglas de negocio

Los *"sockets"* son sistemas de comunicación entre procesos de diferentes computadores de una red, en los que se puede emitir o recibir información.

La comunicación entre procesos se basa en la filosofía cliente-servidor; es decir, en esta comunicación un proceso actuará como servidor creando un *"socket"* cuyo nombre conocerá el proceso cliente, facilitando la comunicación entre ambos.

Una característica de los *"sockets"* es que permiten una comunicación bidireccional, la cual los diferencia de los *"pipes*", o canales de comunicación unidireccional entre procesos de una misma máquina. El mecanismo de comunicación vía *"sockets"* tiene los siguientes pasos: [DDC]

- El proceso servidor crea un *"socket"* con nombre y espera la conexión.
- El proceso cliente crea un *"socket"* sin nombre.
- El proceso cliente realiza una petición de conexión al *"socket"* servidor.
- El cliente realiza la conexión a través de su *"socket"* mientras el proceso servidor mantiene el *"socket"* servidor original con nombre.

Los procesos se comunican solamente entre los sockets del mismo tipo, los cuales se detallan en la Tabla 1.

Nombre	**Característica**	**Tipo socket**	**Protocolo**
Socket de datagrama	Bidireccional, sin duplicados	SOCK_STREAM	TCP
Socket de datagrama	Bidireccional	SOCK_DGRAM	UDP
Socket de paquete secuencial	Bidireccional y secuencial	SOCK_SEQPACKET	No hay protocolo implementado
Raw socket	Permite el acceso a protocolos de más bajo nivel, como el IP.	SOCK_RAW	-

Tabla 1: Listado de tipos de socket

1.4.2.2 RMI

RMI (“*Java Remote Method Invocation*”) permite a los programadores comunicar aplicaciones basadas en tecnologías Java de manera más fácil, permitiendo interactuar con objetos que se ejecutan en máquinas remotas como si fueran objetos locales. Si se requiere establecer comunicación con otras tecnologías se debe usar CORBA o SOAP en lugar de RMI. [SM]

Al estar específicamente diseñado para Java, RMI proporciona la invocación por referencia de objetos (cosa que no hace SOAP), "recolección de basura" distribuida y pasaje de tipos arbitrarios (funcionalidad no provista por CORBA).

Por medio de RMI, un programa Java puede exportar un objeto. A partir de esa operación este objeto está disponible en la red, esperando conexiones en un puerto TCP. Un cliente puede entonces conectarse e invocar métodos. La invocación consiste en la conversión de los parámetros, lo que se realiza mediante la funcionalidad de serialización que provee Java, luego se sigue con la ejecución del método en el servidor. Mientras esto sucede, el que realiza la llamada se queda esperando por una respuesta. Una vez que termina la ejecución, el valor de retorno es serializado y enviado al cliente. El código cliente recibe este valor como si la invocación hubiera sido local.

La arquitectura RMI puede observarse en la Figura 1, como un modelo de cuatro capas: [PEC]

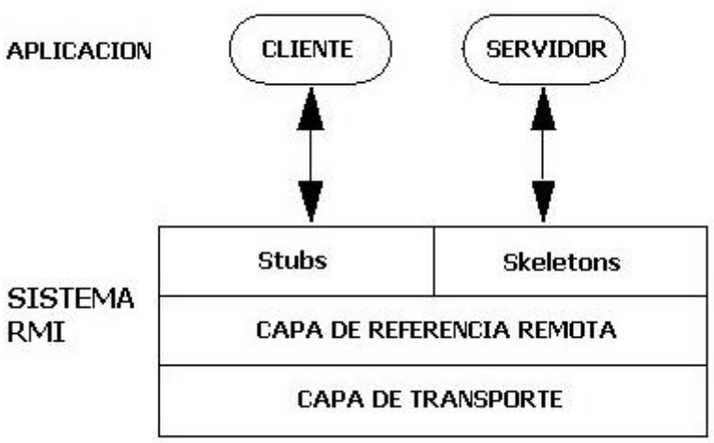


Figura 1: Arquitectura RMI

Primera capa: La primera capa es la de aplicación y se corresponde con la implementación real de las aplicaciones cliente y servidor. Aquí tienen lugar las llamadas a alto nivel para acceder y exportar objetos remotos.
Segunda capa: La segunda capa es la capa "p*roxy"*, o capa stub-skeleton. Esta capa es la que interactúa directamente con la capa de aplicación. Todas las llamadas a objetos remotos y acciones junto con sus parámetros y retorno de objetos tienen lugar en esta capa.
Tercera capa: La tercera capa es la de referencia remota, y es responsable del manejo de la parte semántica de las invocaciones remotas.
Cuarta Capa: La cuarta capa es la de transporte. Es la responsable de realizar las conexiones necesarias y manejo del transporte de los datos de una máquina a otra. El protocolo de transporte subyacente para RMI es JRMP ("*Java Remote Method Protocol"*), que solamente es "comprendido" por programas Java.

1.4.2.3 CORBA

CORBA (Common Object Request Broker Architecture), es una arquitectura estándar para sistemas de objetos distribuidos. CORBA está constituido por tres componentes:

- Un conjunto de interfaces de invocación.
- "*Object request broker"* (ORB), que es una capa de software, también llamado "middleware", que permite a los objetos realizar llamadas a métodos situados en máquinas remotas, a través de una red.
- Conjunto de adaptadores de objetos ("*Objects adapters*").

CORBA llega así, con una infraestructura "*framework"* para construir aplicaciones orientadas a objetos, de esta manera las interfaces definen los servicios que prestan los objetos, luego el ORB se encarga de la localización e invocación de los métodos sobre los objetos y el "*object adapter"* es aquel que une la implementación del objeto con el ORB.

CORBA está fundamentado en dos modelos: Un modelo de objetos, el cual agrega todas las características de la teoría orientada a objetos como son los tipos de datos, abstracción, herencia y polimorfismo y además se fundamenta en un modelo de referencia o arquitectura conocida como *OMA* ("*Object Management Architecture"*) propuesta por la OMG.[OMG]

El modelo de referencia de OMA está compuesto por:

- Object Request Broker - ORB: Representa el medio o bus de objetos a través del cual se comunican todos los objetos participantes en el sistema, provee la infraestructura que permite a objetos comunicarse, independiente de la plataforma específica y técnicas que se usen para implementar el objeto.
- Objetos de Servicio - CORBAServices: conjunto de objetos genéricos, que se usan como soporte para tareas comunes por programas distribuidos.
- Objetos de Dominio – CORBADomain: conjunto de objetos que son comunes y estándares dentro de un dominio o mercado de aplicación.
- Facilidades Comunes - CORBAFacilities: conjunto de objetos orientados hacia las aplicaciones de usuario final como Administración de datos, aplicaciones, interfaces de usuario.
- Objetos de Aplicación: objetos de aplicación desarrollados por los programadores.

En resumen, CORBA es una tecnología que oculta la programación a bajo nivel de aplicaciones distribuidas, de tal forma que los programadores no se tienen que ocupar en trabajar con "*sockets*", flujos de datos, paquetes, sesiones etc. Además, brinda una tecnología orientada a objetos, las funciones y los datos se agrupan en objetos, estos objetos pueden estar en diferentes máquinas, pero el programador accederá a ellos a través de funciones normales dentro de su programa.

1.4.2.4 Servicios Web basados en SOAP

SOAP es un protocolo para el intercambio de mensajes sobre redes de computadores, generalmente usando HTTP, es decir, de una manera muy similar técnicamente a la invocación de páginas Web. Está basado en XML, a diferencia de DCOM y CORBA que son binarios, esto facilita la lectura por parte de los humanos, pero también los mensajes resultan más largos y, por lo tanto, considerablemente más lentos de transferir.

Algunas de las ventajas que tiene SOAP sobre otras tecnologías de invocación de métodos remotos como los expuestos anteriormente son

- Su implementación y uso es sencillo y permite que el desarrollador pueda probar los servicios que esta creando.
- Es un estándar que tiene el respaldo de la W3C y otras empresas líderes de la industria, lo que le otorga respaldo y sustentabilidad en el tiempo.
- Se basa en los mismos estándares de la Web, permitiendo utilizar sus bondades como el manejo de sesión y protocolos de encriptación y autenticación.
- Es capaz de pasar entre "*firewalls*" y "*routers*",
- Utiliza XML como formato de representación de los datos a intercambiar, lo que le da robustez y sencillez en el tratamiento de éstos.
- Es independiente de la tecnología que se utilice en el desarrollo de los clientes.
- Se puede utilizar de manera anónima o mediante autenticación.

Como se expone en una de las ventajas, SOAP utiliza las mismas operaciones que se utilizan normalmente en la Web, GET y POST, que funcionan de la misma manera que en los navegadores, además de estas se agrega la operación SOAP, la cual es similar a POST pero las solicitudes se hacen en XML y permiten la transferencia de recursos más complejos como tipos de datos creados por el usuario y arreglos.

Independiente del método utilizado para realizar la solicitud, las respuestas siempre se realizan mediante XML ya que con esto se evitan problemas ocasionados por cambios en los servicios, permitiendo validar los argumentos de las funciones, entregando solidez al protocolo.

Por esta misma razón, SOAP define un estándar llamado WSDL, que permite describir la interfaz del servicio Web. En esta interfaz de define cuáles son los parámetros que el servicio permite recibir, el tipo de estos parámetros y la respuesta que entregará. Este archivo WSDL también está definido mediante un archivo XML el cual es publicado por el servidor con el fin de establecer un contrato mediante el cual acceder al servicio.

Para la implementación de servicios Web basados en SOAP existen diversos paquetes tanto comerciales como de código abierto que facilitan la tarea de creación de servicios, estos paquetes están disponibles para gran parte de los lenguajes de alto nivel

existentes en el mercado, tales como las tecnologías .NET y JAVA por nombrar algunas. [UAL]

Aspectos sobre la seguridad

En un principio fue suficiente la seguridad entregada por el protocolo HTTP, pero actualmente existen recomendaciones de seguridad en la construcción de servicios Web basados en SOAP como la que se comenta a continuación.

WS-Security

Esta especificación propone un conjunto de extensiones estándar que pueden ser utilizadas cuando se construyen servicios Web seguros, estas extensiones otorgan integridad y confidencialidad al contenido de los mensajes.

WS-Security es flexible y está diseñada para ser usada como base para la seguridad en los servicios con una amplia variedad de modelos de seguridad incluyendo PKI[4], Kerberos[5] y SSL[6].

Esta especificación provee tres principales mecanismos: la propagación de "*tokens*" de seguridad, la integridad de mensajes, y la confidencialidad de mensajes. Estos mecanismos por sí mismos no proporcionan una completa solución de seguridad; sin embargo, pretenden ser un bloque de construcción que puede ser usado con otras extensiones y protocolos específicos de aplicación de más alto nivel para lograr una amplia variedad de modelos de seguridad y tecnologías de encriptación. Estos mecanismos se pueden utilizar independientemente o en una combinación de ellos. [OAS]

1.4.2.5 Servicios REST

A diferencia de SOAP, REST no es un estándar sino que es un tipo de arquitectura de software que está basado en el funcionamiento natural de la Web, utilizando HTML como protocolo y las acciones GET, POST, PUT y UPDATE. Como REST no es un estándar, no existe una definición formal ni una especificación dada por

[4] Public Key Infrastructure.
[5] Kerberos es un protocolo de autenticacion de red
[6] Secure Socket Layer

algún organismo de estandarización como la W3C u OASIS. Es equivalente a la arquitectura cliente servidor, existe como tal, pero no existe una especificación para ella.

REST se basa en el flujo de recursos a través de la Web, recursos que pueden ser de cualquier tipo, ya sea un archivo de texto, XML, GIF, JPG, etc. Estos recursos son accesados a través de un identificador único llamado URI (Identificador Uniforme de Recurso) en donde el servidor expone los recursos que están disponibles.

Un cliente REST puede solicitar un recurso a través de HTTP GET, acción que permite descargar el archivo expuesto como servicio, este archivo podría ser por ejemplo, un listado de clientes o una lista de precios de un producto en particular, logrando de esta forma realizar una acción similar a un servicio Web SOAP.

Las principales diferencias entre un servicio Web basado en REST y uno que utiliza SOAP se resumen en la Tabla 2. [DSI]

	REST	SOAP
Características	Las operaciones se definen en los mensajes. Una dirección única para cada instancia del proceso. Cada objeto soporta las operaciones estándares definidas. Componentes débilmente acoplados.	Las operaciones son definidas como puertos WSDL. Dirección única para todas las operaciones. Múltiple instancias del proceso comparten la misma operación. Componentes fuertemente acoplados.
Ventajas declaradas	Bajo consumo de recursos. Las instancias del proceso son creadas explícitamente. El cliente no necesita información de enrutamiento a partir de la URI inicial. Los clientes pueden tener una interfaz “listener” (escuchadora) genérica para las notificaciones. Generalmente fácil de construir y adoptar.	Fácil (generalmente) de utilizar. La depuración es posible. Las operaciones complejas pueden ser escondidas detrás de una fachada. Envolver APIs existentes es sencillo Incrementa la privacidad. Herramientas de desarrollo.
Posibles desventajas	Gran número de objetos. Manejar el espacio de nombres (URIs) puede ser engorroso. La descripción sintáctica/semántica muy informal (orientada al usuario). Pocas herramientas de desarrollo.	Los clientes necesitan saber las operaciones y su semantica antes del uso. Los clientes necesitan puertos dedicados para diferentes tipos de notificaciones Las instancias del proceso son creadas implícitamente

Tabla 2: Diferencias entre servicios REST y SOAP [DSI].

CAPITULO II: DESCRIPCION DE LA PROBLEMÁTICA

2.1 Descripción del capítulo

El objetivo de este capítulo es dar a conocer la empresa en la cual se desarrolló el proyecto, como así también describir la problemática y la solución propuesta

2.2 Descripción de la empresa y estado actual

Telefonica del Sur S.A. (telsur) es una empresa líder en servicios integrales de telecomunicaciones en el sur de chile, diferenciándose por la capacidad de innovación como lo ha demostrado con el lanzamiento de la televisión digital, el sistema WiTV, que es el primero en instalarse en Chile y Sudamérica. Con esto, Telsur demuestra su interés en innovar en materia de telecomunicaciones, habiéndolo ya demostrado el año 2006 al dar portabilidad a la telefonía local a través de su producto “súper inalámbrico”.

Este compromiso con la innovación hace que para la empresa sea fundamental contar con la logística y sistemas de información adecuados a la rápida incorporación de nuevos productos y apertura de nuevos mercados.

Por esta razón, la Gerencia de Tecnologías, a través de la Sub-Gerencia de Informática, centra sus esfuerzos en la optimización de los procesos, a través de la aplicación de las tecnologías que permitan resolver de manera ágil y rápida tanto las actividades operacionales como de gestión del día a día de la compañía.

La Figura 2 nos muestra un extracto del diagrama organizacional, enfocándose en los departamentos de la gerencia de tecnologías.

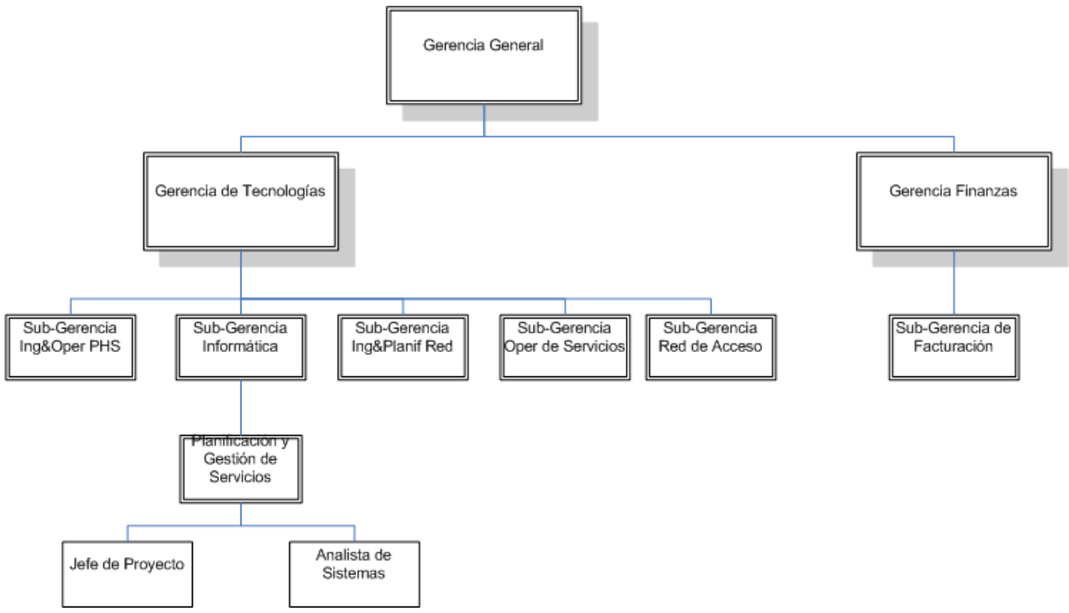


Figura 2: Diagrama Organizacional

El área encargada de los sistemas de tasación y facturación es la Sub-Gerencia de Facturación, dependiente de la Gerencia de Finanzas. Esta área es la que revisa que los procedimientos aplicados sean los correctos y define las áreas que deben ser intervenidas en los procesos cuando se realiza algún cambio de las reglas de negocio por parte del área comercial u otra. Por otra parte, la Sub-Gerencia de Informática es la que realiza los cambios decretados por la Sub-Gerencia de Facturación.

Como una forma abordar de manera más eficiente los cambios solicitados, se ha intentado parametrizar los datos más importantes de las aplicaciones mediante procedimientos almacenados o tablas que contienen la información acerca de las reglas del negocio más importantes.

Aun así, esta estrategia no es suficiente, debido a que igualmente hay reglas embebidas en el código y se hace complejo saber cuáles son las tablas y procedimientos que agrupan la lógica de negocio, y en muchos casos no existe tal parametrización.

Actualmente, sólo el personal de informática a cargo de las aplicaciones conoce el funcionamiento de éstas, y se hace muy compleja la manipulación de las reglas por parte del personal que crea las políticas de negocio, lo cual involucra un mayor tiempo de respuesta ante cualquier cambio, ya que siempre es necesario involucrar en éstos al personal de informática, el cual debe modificar el código fuente de las aplicaciones que

ya están en producción, proceso que debe contar con los cuidados necesarios para prevenir cualquier incidencia mayor producto de la modificación, no eliminando con esto la probabilidad de error.

Cuando un usuario de negocio necesita hacer modificaciones a los sistemas, este necesita seguir un procedimiento específico, el cual se detalla en la Figura 3.

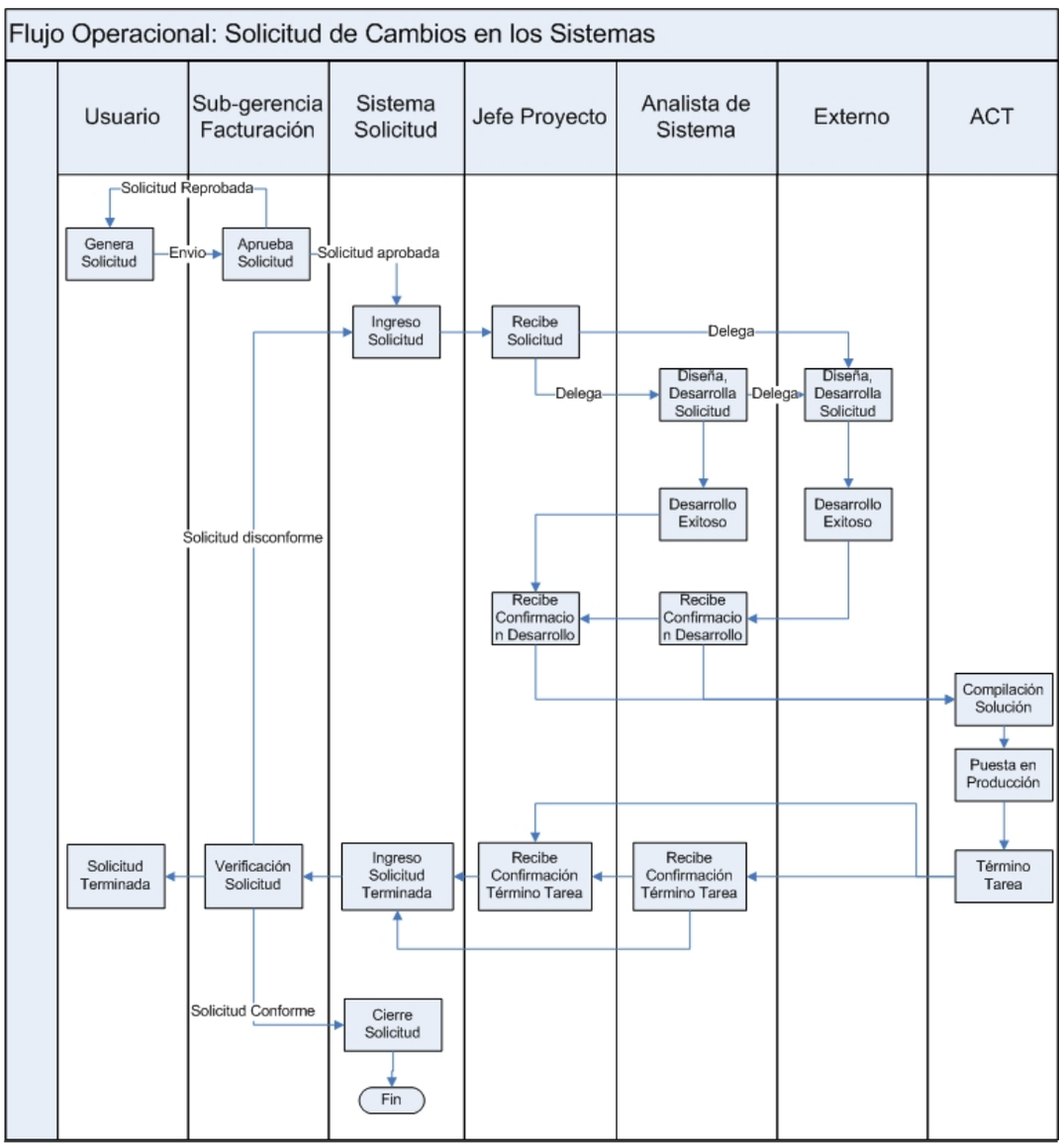


Figura 3: Flujo Operacional para cambios en el sistema de tasación

Este diagrama muestra cuáles son los estados que tienen las solicitudes y cuáles son las opciones que existen actualmente dentro del área de informática para resolverlas.

2.3 Solución Propuesta

Para dar solución a la demora en la reacción ante cambios o creación de nuevas reglas en el código fuente de los sistemas, como también involucrar al personal del negocio, se propone, como lo muestra la Figura 4, separar las reglas de negocio del código, manteniendo motores de reglas de negocio y herramientas que permitan el mantenimiento de las mismas, como son los Sistema de Gestión de Reglas de Negocio (BRMS).

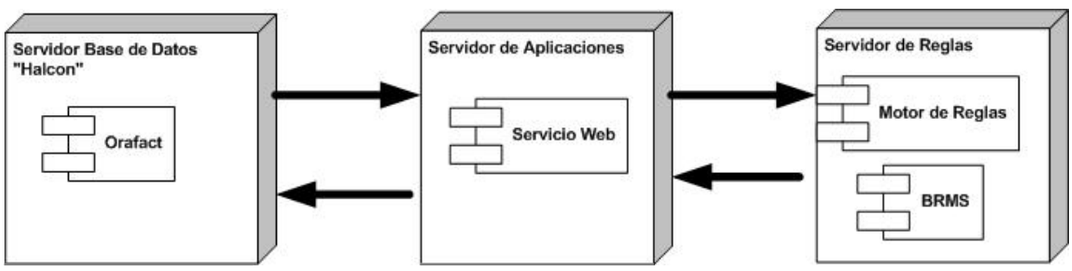


Figura 4: Arquitectura de capas Motor de reglas

Para realizar esto es necesario conocer en detalle el sistema, con el fin de obtener las reglas de negocio existentes en él. Una vez conocidas las reglas se debe proceder a escribirlas en el lenguaje del motor de reglas para que así, los sistemas de Telsur se conecten al motor de reglas y puedan ejecutarlas.

Una vez que el sistema este operativo el flujo de creación, edicion o eliminación requerido por los usuarios de negocio quedaria como lo muestra la Figura 5, donde una vez aprobada por la sub-gerencia de facturación, ésta añade o modifica las reglas de negocio en el BRMS. Si no es posible insertar o modificar una regla se debe ingresar la solicitud correspondiente para que así el departamento de informática a través del jefe de proyecto y el analista del sistema, vean la factibilidad de realizarla. Si es factible se desarrollan los módulos necesarios y se da aviso de que la solicitud puede llevarse a cabo. De no ser factible, se da aviso del motivo y se termina el proceso, pudiendo rehacerse la solicitud.

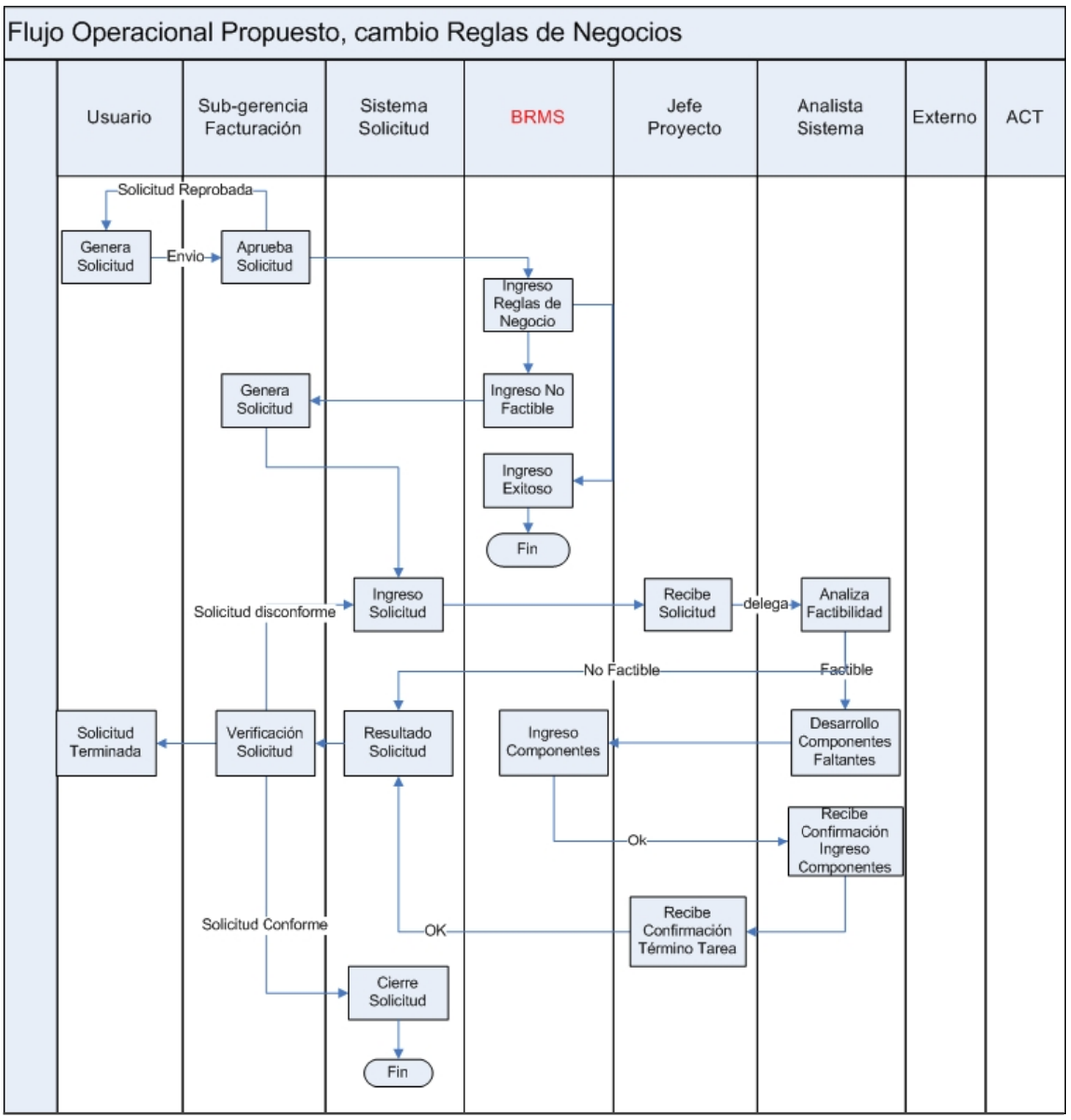


Figura 5: Flujo Operacional para cambios de reglas de negocio con Motor de Reglas

2.4 Beneficios esperados de la solución

Los beneficios son importantes, especialmente cuando se trata de hacer cambios en las reglas de negocio. Esto significa que, las personas del área de negocios pueden trabajar ellos mismos en los cambios que consideren necesarios, porque tienen el control directo de las reglas. Dando una mayor confianza en que las reglas de negocio se ejecutan de acuerdo a lo previsto por la gerencia o quien corresponda.

Además permite dar una mayor velocidad de respuesta ante el desarrollo de nuevos productos permitiendo a la empresa reaccionar con rapidez ante la competencia que intenta penetrar en el actual mercado, otorgando mayor competitividad.

Otro aspecto que mejora son los costos de mantención de los sistemas de la empresa pudiendo realizar cambios en su funcionamiento en tiempo real, por lo cual no es necesario detener los sistemas en producción al implantar una nueva característica o cambiar una existente.

CAPITULO 3: DESCRIPCION Y SELECCIÓN DE TECNOLOGIA

3.1 Descripción del Capitulo

Actualmente en el mercado existen diversas tecnologías como motores de reglas de negocio tanto Open Source como licenciadas. El propósito de este capitulo es dar a conocer estas tecnologías con el fin de elegir de acuerdo a sus características cuál es la más adecuada para el desarrollo de este proyecto.

3.2 Análisis de Motores de Reglas de Negocio

3.2.1 Oracle Business Rules

Oracle Business Rules pertenece a la plataforma SOA de Oracle. Este producto funciona de manera conjunta con el Servidor de Aplicaciones de dicha compañía [OBR].

Entorno de Desarrollo

Para facilitar el desarrollo de aplicaciones basadas en reglas, Oracle Business Rules cuenta con las siguientes herramientas:

Rule author**:** permite trabajar con reglas de negocio desde cualquier navegador. En otras palabras, es una interfaz Web con la capacidad de crear nuevas reglas o editar las ya existentes.

Rule SDK**:** Oracle Business Rules SDK es una biblioteca java que permite la administración de distintas características de las reglas de negocio. Un desarrollador puede usar esta librería para escribir las reglas del programa.

Rules Engine: Oracle Business Rules Engine (Rules Engine) es una librería java que se encarga de procesar las reglas de acuerdo a los hechos que tiene

disponibles en la memoria de trabajo, además define un lenguaje de reglas declarativo, provee un motor de inferencia y herramientas que soportan debug.

Algoritmo

En Oracle Business Rules, el sistema basado en reglas es manejado por los hechos, y con encadenamiento de reglas hacia adelante. Los hechos determinan cuáles reglas pueden ser ejecutadas. Cuando una regla es ejecutada, ésta podría agregar nuevos hechos, y estos nuevos hechos pueden conducir a la ejecución de nuevas reglas. Este proceso se repite hasta obtener una conclusión final. Este proceso se llama ciclo de inferencia

Compatibilidad y plataformas

El Rule Engine de Oracle Business Rules se puede usar de dos formas:

Jar Embebido: En esta opción, el motor de reglas va embebido en la aplicación que lo utiliza, por lo tanto se puede utilizar sólo en aplicaciones Java, las cuales a su vez son multiplataforma

Decision Service**:** Otra forma es usarlo a través de servicios Web, de esta forma es posible consumir las reglas como un servicio, por lo que el motor puede ser utilizado por aplicaciones desarrolladas en cualquier lenguaje.

En resumen, Oracle Business Rules puede trabajar directamente con java o interactuar por medio de servicios Web para estar disponible para aplicaciones escritas en cualquier lenguaje o en cualquier plataforma.

Características Generales

Alto rendimiento: Rules Engine implementa algoritmos especializados para los hechos que son definidos en el sistema.
Hilo de ejecución para una arquitectura en paralelo: Rules Engine provee un thread que puede sostener hechos mientras otro es evaluado en la red.
Agilidad: Rules Engine permite cambiar las reglas sin parar los sistemas en proceso.

3.2.2 Open Rules

Es un sistema de administración de reglas de negocio de gran escala, que está formado por un conjunto de herramientas de código abierto que permiten desarrollar aplicaciones basadas en reglas. [OPE]

Entorno de Desarrollo

El repositorio de Open Rules está basado en un conjunto jerárquico de libros y hojas de cálculo, las cuales entre sí, y con un formato claramente definido forman la base de reglas.

Para el proceso de desarrollo se integran distintas herramientas de código abierto. Las distintas funcionalidades que proporciona Open Rules para el desarrollo y las herramientas involucradas en cada una se muestran en la Figura 6:

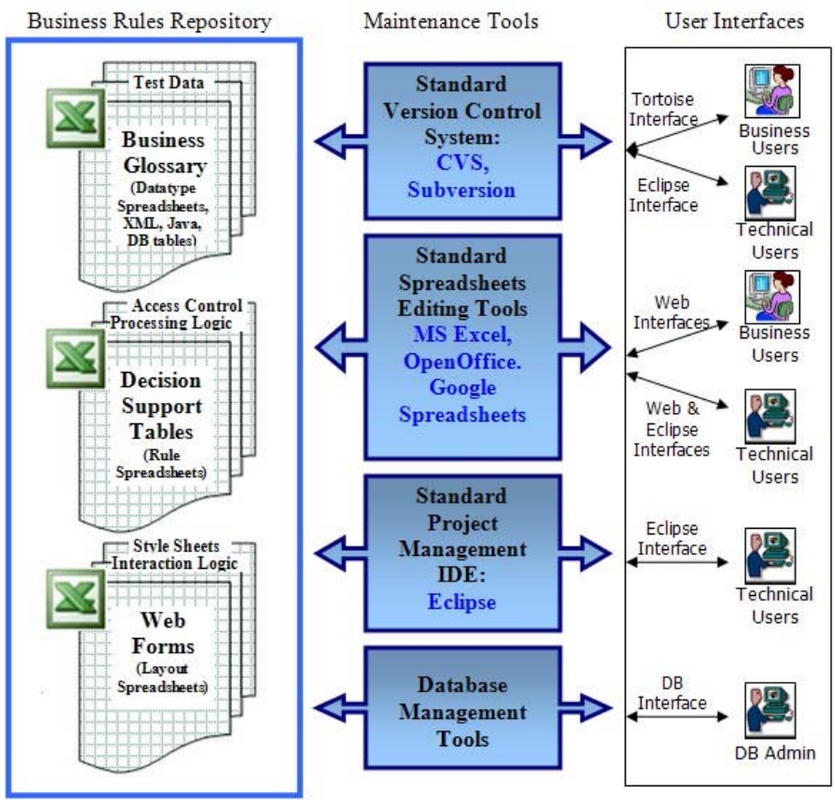


Figura 6: Funcionalidades y Herramientas Open Rules **[OPE]**

Algoritmo

La ejecución de reglas en Open Rules es muy intuitiva, no asume ningún orden implícito ni aplica algún algoritmo que especifique el orden de ejecución de las reglas, éstas simplemente se ejecutan en el orden especificado en el diseñador de reglas. Todas las reglas son ejecutadas una a una en el orden que fueron ubicadas en las tablas de reglas.

Para las tablas de decisión de forma vertical, las reglas son ejecutadas de arriba hacia abajo, mientras que para las tablas de decisión de forma horizontal, las reglas son ejecutadas de izquierda a derecha.

La lógica de ejecución es como sigue:

"Si todas las condiciones son satisfechas ENTONCES ejecutar todas las acciones".

Si al menos una condición es violada, todas las otras en la misma regla son ignoradas y no son evaluadas. La ausencia de un parámetro significa que la condición que utiliza dicho parámetro es verdadera.

Compatibilidad y plataformas

Existen tres formas de desplegar la funcionalidad del motor de reglas:

Como un método Java: De esta forma, la funcionalidad queda a disposición de cualquier aplicación que se desarrolle en Java, restringida a este lenguaje, entregando una excelente compatibilidad y velocidad.

Como una aplicación Web: Open Rules permite a los usuarios de negocio definir sus propias interfaces Web para interactuar con la base de reglas, esto se hace mediante la característica "Web Forms" a través de la cual el usuario define en la misma planilla Excel en la que se detallan las reglas, un formulario desde el cual se obtendrán los parámetros de entrada de ellas. A partir de esta especificación, el motor generará automáticamente un formulario Web JSP.

Como un servicio Web: De esta manera se consigue que el motor interactúe con aplicaciones desarrolladas en cualquier lenguaje y por lo tanto se pueda utilizar cualquier plataforma para su funcionamiento.

En resumen, se puede utilizar el motor para trabajar en aplicaciones Java de manera nativa, las cuales son independientes de la plataforma. A su vez, si se quiere interactuar con otro lenguaje, se puede utilizar la alternativa de los servicios Web, la cual si bien es un poco más pobre en cuanto a velocidad, entrega la posibilidad de conectar el motor con cualquier lenguaje.

Licenciamiento

Open Rules está disponible para cualquier persona bajo la licencia Open Source "GNU General Public License" (GPLv2).

Características Generales

Rule Validator: Provee validación de reglas y detección de errores
Rule Testing: Provee la capacidad de probar las reglas en base a datos proporcionados en planillas de cálculo.
Rule Learner y Rule Solver: Ayudan a que el motor tenga una mayor performance, descubriendo patrones regulares de uso de la base de reglas y obteniendo la manera óptima de ejecución de éstas.

3.2.3 NxBRE

NxBRE fue el primer motor de reglas de código abierto orientado a la plataforma .NET de Microsoft. NxBRE tiene dos componentes principales [AGI]:

Entorno de desarrollo

El motor de inferencia utiliza un lenguaje de reglas llamado RuleML el cual es propuesto como el estándar de especificación de reglas para diversos propósitos como

la Web semántica, reglas de negocio y reglas transformación de documentos XML entre otros.

Este lenguaje se basa en XML por lo que su creación, modificación o eliminación se puede hacer de diversas formas:

Microsoft Visio: La distribución de NxBRE incluye una plantilla para Microsoft Visio mediante la cual se puede administrar gráficamente las reglas. Con esta interfaz se permite a los usuarios de negocio interactuar con las reglas de manera más simple y comprensible.

Editores XML: Para la administración de reglas se puede utilizar desde el más básico editor de texto, como Notepad, incluido en Microsoft Windows, hasta editores especializados en XML como XML Spy o el plug-in WTP de Eclipse el cual incluye un editor muy intuitivo para este lenguaje.

Human Readable Format: Consiste en una transformación XSLT mediante la cual se transforma las bases de reglas de formato RuleML a un formato más comprensible y manejable por usuarios no especializados. Esta funcionalidad está actualmente en desarrollo y no soporta expresiones donde se hagan combinaciones complejas de and y or, pero es capaz de traducir bases de reglas sencillas.

Algoritmo

Flow Engine: utiliza XML para mantener el control del flujo de ejecución del proceso. Proporciona todas las sentencias de ejecución conocidas como if/then/else, while, foreach, etc. Todo esto en un contexto de objetos de negocio y resultados.

Inference Engine: Es un motor de inferencia de encadenamiento hacia delante que soporta conceptos como hechos, consultas, prioridad de reglas y exclusión mutua. Este alienta la separación de roles entre los expertos que diseñan las reglas de negocio y el personal de informática que une éstas con los objetos de negocio.

Compatibilidad y plataformas

NxBRE está diseñado para la plataforma .NET de Microsoft, por lo tanto, su ejecución de manera nativa está limitada a las plataformas en las que exista máquina virtual para .NET. Actualmente existe dicha compatibilidad para sistemas Windows mediante Microsoft .NET Framework y en sistemas Linux con el Proyecto MONO.

Adicionalmente está la posibilidad de exponer el motor como un Servicio Web, con lo cual las aplicaciones clientes del motor pueden ser escritos en cualquier lenguaje y correr sobre cualquier plataforma

Licenciamiento

NxBRE está licenciado bajo los términos de la LGPL (Lesser General Public License), la cual permite su uso en aplicaciones comerciales.

Características Generales

NxBRE, a diferencia de otros motores analizados, dispone de dos formas distintas de ejecución de reglas, el motor de inferencia que es capaz de obtener conclusiones a partir de hechos y reglas, y el motor de flujo que permite realizar acciones a través de sentencias de control conocidas.
Este motor de reglas está basado en JxBRE, el cual está construido para plataformas Java.

3.2.4 JBoss Rules

JBoss Rules, anteriormente conocido como Drools, sirve para implementar la lógica de evaluación y decisión junto a conjuntos de datos y hechos de gran tamaño [JBO].

Entorno de Desarrollo

El entorno de desarrollo para las reglas del negocio usado por Jboss Rules es Eclipse, al cual, para permitir la administración de las reglas, debe instalarse un “plug-in”, el que añade los elementos necesarios para la creación y modificación de las reglas del negocio.

Algoritmo

El motor de reglas de Jboss Rules basa su ejecución en el algoritmo rete, optimizado para sistemas orientados a objetos, este algoritmo basa su ejecución en el método de encadenamiento hacia delante.

Compatibilidad y plataforma

Jboss Rules permite el despliegue en múltiples plataformas, ya que es una solución Java, permitiendo crear y ejecutar las mismas reglas en entornos Java, .NET u otros. Además incorpora la posibilidad de trabajar con servicios Web, permitiendo la conexión y traspaso de información al motor de reglas desde cualquier plataforma o ambiente de desarrollo.

Licenciamiento

Jboss Rules es un motor de reglas Open Source.

Características Generales:

Jboss Rules incluye un administrador Web de reglas (BRMS), que lo hace disponible desde cualquier computador vía HTTP, permitiendo la comunicación entre los administradores del negocio y los que manejan la tecnología de la informática, dando a los primeros control en la creación de la lógica del negocio (reglas de negocio).

3.2.5 Blaze Advisor

Blaze Advisor es una solución completa y avanzada para la gestión de reglas, que abarca todo el proceso de creación, implantación y mantenimiento de las aplicaciones basadas en reglas, especialmente en aquellas de carácter transaccional y operativo.

Este software ofrece un entorno de desarrollo para crear modelos de reglas, así como para probarlas y realizar simulaciones, antes de generar automáticamente un código para su despliegue en las aplicaciones de producción. La herramienta ofrece un

entorno sofisticado de gestión completa de reglas y está enfocada en aplicaciones empresariales a gran escala o de gran valor, especialmente cuando es necesario tomar decisiones en tiempo real. Los componentes de Blaze Advisor son [FAI]:

Entorno de Desarrollo

Las reglas se definen utilizando el entorno de desarrollo integrado (IDE) de Blaze Advisor, que es un entorno completo para la creación, modificación y comprobación de reglas, además de especificación de plantillas para la creación de reglas, las cuales permite a usuarios sin conocimiento técnico la creación y modificación de las reglas

Algoritmo

Implanta y ejecuta las reglas inteligentemente, basa su ejecución en el algoritmo rete, permite la ejecución escalable de reglas e integra las base de datos y otras fuentes de información para el uso de evaluación de las reglas

Compatibilidad y Plataforma:

Un elemento diferenciador clave es el despliegue en múltiples plataformas, ya que es una solución cien por ciento Java que permite crear y ejecutar las mismas reglas en entornos Java, .NET y COBOL.

Licenciamiento

Blaze Adivisor pertenece a la corporación con fines de lucro Fair Isaac, su licencia es de tipo comercial.

Características Generales

- Gestor de desarrollo, introduce reglas nuevas en los sistemas de producción sin requerir tiempo de inactividad.
- Almacén de reglas, almacena las reglas y gestiona los cambios en ellas.

- Aplicación de mantenimiento de reglas. Permite al personal sin formación técnica revisar y actualizar las reglas sin necesidad de programación.

3.2.6 BizTalk Server

BizTalk Server (BTS) es una plataforma para la integración de procesos de negocio. Lo que se consigue con BizTalk es integrar procesos de sistemas heterogéneos u homogéneos de forma que puedan comunicarse con facilidad.

BizTalk Server provee un ambiente de trabajo (framework) utilizado en la implementación y ejecución de las reglas de negocio. Business Rule Framework (BRF) se interpreta como un conjunto de librerías que proveen la funcionalidad necesaria para la creación, evaluación y ejecución de cada una de las reglas de negocio establecidas previamente por el proceso. A su vez, el BRF se compone de [MIC]:

Entorno de Desarrollo.

El Business Rule Composer es el entorno de desarrollo de Biztalk Server, que puede ser utilizado por desarrolladores, analistas del negocio y/o administradores de los procesos de negocio, durante el proceso de creación e implementación. Esta herramienta es usada para autorizar, crear, desplegar y definir reglas de negocio, a su vez se compone de las siguientes vistas:

- Policy Explorer: se administran las políticas y reglas. Por medio de esta vista se pueden crear, modificar y eliminar políticas y reglas, además de probarlas y publicarlas en la base de datos.
- Facts Explorer: se administran los elementos “facts” utilizados por las reglas como vocabularios, esquemas XML, bases de datos y clases de .Net.
- Action Editor**:** se utiliza para ver y editar las acciones que se deben ejecutar cuando una regla es válida.

Algoritmo:

El motor de reglas de negocio (BRE[7]) implementa el mecanismo de evaluación, basado en el algoritmo Rete, para las reglas de negocio.
En su arquitectura interna, BRE está compuesto por tres componentes:

• Ruleset executor: es el responsable de evaluar y ejecutar las reglas del negocio.
• Ruleset translator: toma el objeto de la regla del negocio de entrada y lo traduce en una presentación ejecutable.
• Rule set tracking interceptor: toma la salida generada en la ejecución de la regla del negocio, y la envía a las herramientas de monitoreo.

Licenciamiento

Su licencia es de carácter comercial, perteneciente a Microsoft.

Compatibilidad y plataforma

Biz Talk Server puede ser desplegado en plataforma Windows solamente.

Características Generales

Una de las principales características del BRE es que está diseñado para que las reglas de negocio puedan ser invocadas desde procesos de integración en BTS o desde aplicaciones desarrolladas en .Net.

Mediante BizTalk podemos consumir información generada por un proceso que puede llegar de distintas formas (MAIL, BBDD, Archivo plano, FTP, SAP) decidir cómo tratar esa información y enrutarla a otro proceso mediante reglas de negocio que pueden ir cambiando dinámicamente en tiempo real. Los destinos pueden ser por ejemplo Sharepoint, una base de datos, un FTP, un archivo o construir nosotros manualmente el formato y protocolo destino.

[7] Bussiness Rules Engine

3.3 Servidor de aplicaciones

3.3.1 Jboss AS

Jboss AS es un servidor de aplicaciones J2EE de código abierto, el que puede ser utilizado en cualquier sistema operativo. Combina una arquitectura orientada a servicios con una licencia de código abierto, JBoss AS puede ser descargado, utilizado, incrustado, y distribuido sin restricciones por la licencia. Por este motivo es la plataforma más popular de "*middleware"* para desarrolladores.

Un servidor de aplicaciones java se encuentra compuesto por dos partes, un motor de Servlet y otro de EJB. Dentro del motor servlet se ejecutan exclusivamente las clásicas aplicaciones de servidor, como lo son JSP's y Servlets. Mientras que el motor EJB se ocupa de las aplicaciones desarrolladas con EJB, es por ello que Jboss AS se utiliza con un Web-Container, que puede ser cualquiera, siendo el característico Tomcat.[JBO]

Las características más importantes son:

- Cumple con los estándares.
- Confiable a nivel de empresa
- Incrustable, orientado a arquitectura de servicios.
- Flexibilidad consistente
- Servicios del "*middleware*" para cualquier objeto de Java
- Ayuda profesional
- Soporte completo para JMX
- Opción de Clustering

3.4 Análisis Comparativo y Selección

Para la elección del motor de reglas se consideró un conjunto de características deseables para la implementación final, cada una de ellas tiene un peso, lo cual hará que el motor elegido sea el que tenga la máxima sumatoria de puntaje. Este peso será multiplicado por 1 o 0 dependiendo si la característica está o no disponible. La lista de características seleccionadas y sus pesos se detallan en la Tabla 3:

Característica	Peso en %
Administración de reglas vía Web	15
Administración gráfica de reglas (Excel, Visio, Otro)	5
Compatibilidad multiplataforma	15
Disponibilidad de soporte	10
Manejo de DSL	10
Control de cambios de reglas	5
Usuarios	10
Paquete de reglas vía HTTP	5
Entorno de pruebas de reglas	5
Open Source	20

Tabla 3: Peso por característica de motores de reglas

El resumen de las características disponibles en cada motor de reglas estudiado es lo muestra la Tabla 4:

	Administración de reglas vía Web	Administración gráfica de reglas	Compatibilidad nativa multiplataforma	Disponibilidad de Soporte	Manejo de DSL	Control de versiones de reglas	Login Usuarios	Paquete de reglas vía HTTP	Entorno de pruebas de reglas	Open Source	**Puntaje**
JBoss Rules	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**O**	**X**	**95**
Oracle Business Rules	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**O**	**80**
OpenRules	**O**	**X**	**X**	**X**	**O**	**X**	**O**	**O**	**O**	**X**	**55**
NxBRE	**O**	**X**	**O**	**X**	**O**	**O**	**O**	**O**	**O**	**X**	**35**
Blaze Advisor	**X**	**X**	**X**	**X**	**O**	**O**	**X**	**O**	**X**	**O**	**60**
BizTalk Server	**O**	**X**	**X**	**X**	**X**	**X**	**X**	**O**	**X**	**O**	**60**

Tabla 4: Análisis Comparativivo por motor de reglas

Como se ve en la tabla resumen anterior, el producto que mejor cumple con las características esperadas es JBoss Rules, ya que posee gran parte de ellas y lo más importante, es de código abierto, lo que permite reducir costos, manteniendo características similares a las de sus contrapartes que son licenciadas como Oracle Business Rules o BizTalk Server de Microsoft.

Dentro de los servidores de aplicaciones anteriormente descritos, la alternativa que más se acomodó a la necesidad de soportar Jboss Rules fue Jboss, ya que todas las implementaciones operan de mejor manera bajo este servidor, según lo investigado.

3.5 Descripción de la herramienta seleccionada

JBoss Rules es un motor de reglas de negocio open source basado en estándares, rápido y altamente eficiente que facilita ver las reglas del negocio a medida que son codificadas. JBoss Rules también soporta varias formas de ingresos de reglas mediante tablas de decisión y lenguaje, facilitando la rápida modificación de las políticas comerciales para responder a las oportunidades y amenazas competitivas del mercado.

La arquitectura de Jboss Rules se muestra en la Figura 7.

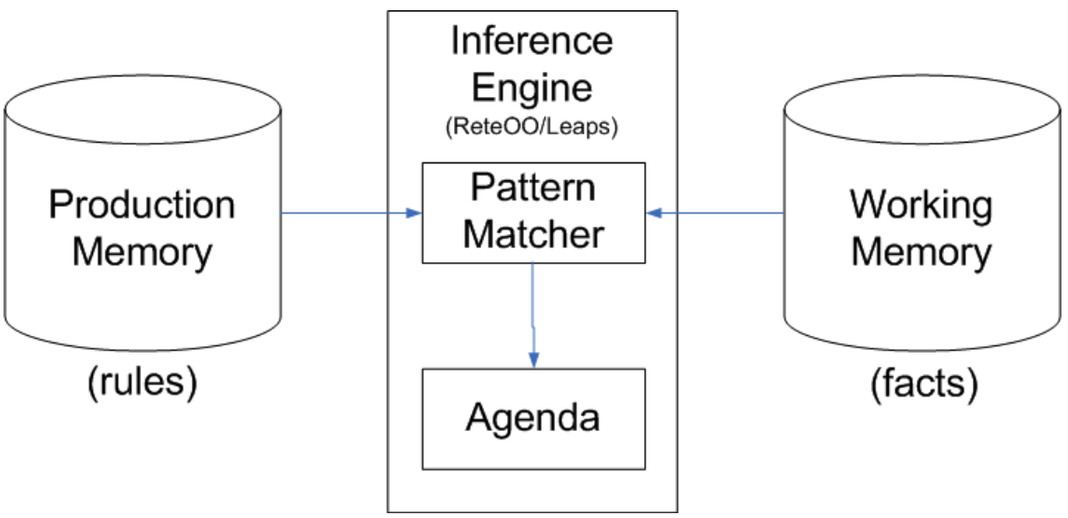


Figura 7: Arquitectura Jboss Rules [JBO]

El motor de inferencia debe sacar conclusiones desde los hechos y las reglas de producción, dando como resultado acciones. Este proceso es llamado “Patter Matching” el cual se basa en el algoritmo ReteOO.

El motor de inferencia controla el proceso completo de aplicar los hechos de las reglas a la memoria de trabajo. Las reglas son almacenadas en la memoria de producción, mientras que los hechos son insertados en la memoria de trabajo donde pueden ser modificados o retractados.

IDE: Rule Workbench

Jboss Rules permite la administración avanzada de las reglas para los desarrolladores, como se observa en la figura 8, a través del entorno de desarrollo Eclipse de Java, el cual provee funcionalidades mediante la instalación de un plug-in, disponible en la página oficial de Jboss [JBO], el que añade nuevas características al entorno de desarrollo, como lo muestra la Figura 9.

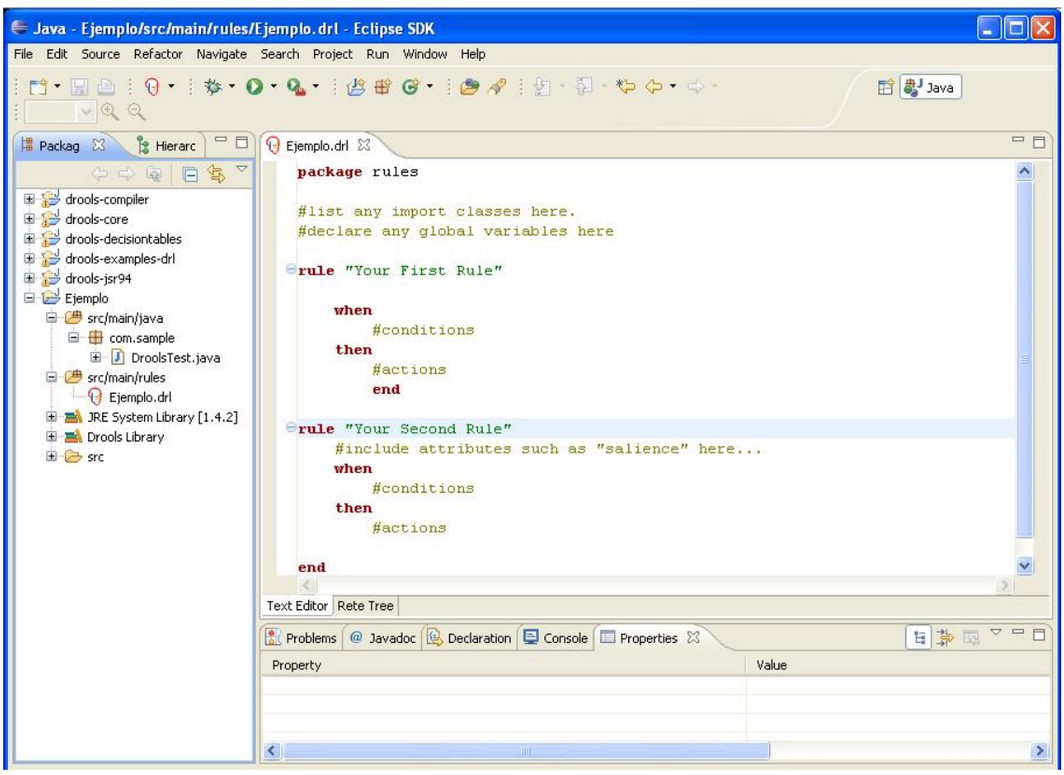


Figura 8: Jboss Rules en Eclipse

Las características que posee son:

- Editor textual y grafico de reglas.
- Un editor que conoce de la sintaxis DRL, facilitando asistencia al desarrollo.
- Wizards que ayudan a:
    - La creación rápida de nuevos proyectos
    - Creación de nuevos recursos, paquetes de reglas
    - Creación de nuevo lenguaje DSL

- Editor de DSL
- Validador de reglas

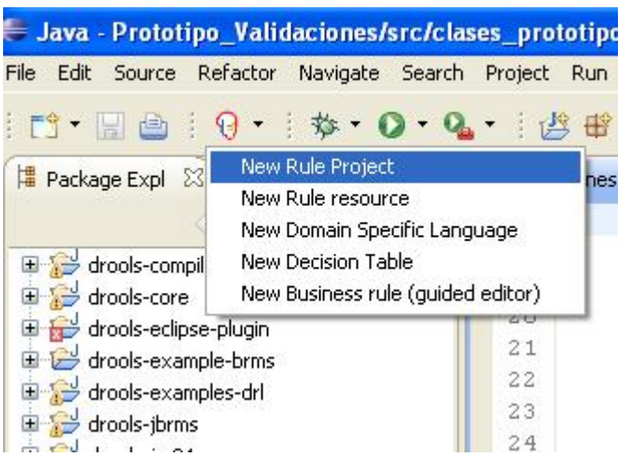


Figura 9: Plugins Eclipse de Jboss Rules

El lenguaje de reglas.

Archivo de reglas.

Un archivo de regla es típicamente un archivo con extensión .drl, que puede contener múltiples reglas, funciones, consultas, así como también declaración de recursos, como imports, atributos y variables globales que son usados por las reglas. La estructura del archivo de reglas es el siguiente:

```
package package-name
imports
globals
functions
queries
Reglas
```

El orden de los elementos que son declarados no importa, excepto el nombre del paquete, que si es declarado debe ser el primer elemento en aparecer.

Las reglas tienen la siguiente estructura:

```
rule "nombre de la regla"
     atributos
     when
          LHS (lado izquierdo, condición)
     then
          RHS (lado derecho, acción)
end
```

Como todo lenguaje de programación, el lenguaje de reglas de Jboss Rules posee palabras reservadas, las cuales no pueden ser ocupadas como nombre de variables.

Rule Flow

Jboss Rules, a través de “Rule Flow” permite definir el orden en que son evaluadas un grupo de reglas o reglas de un paquete, pudiendo ser de forma secuencial o paralela. Todo esto se lleva a cabo mediante un editor gráfico, como muestra la Figura 10, semejante a un workflow, donde se dan las condiciones para que se ejecute determinado grupo de reglas.

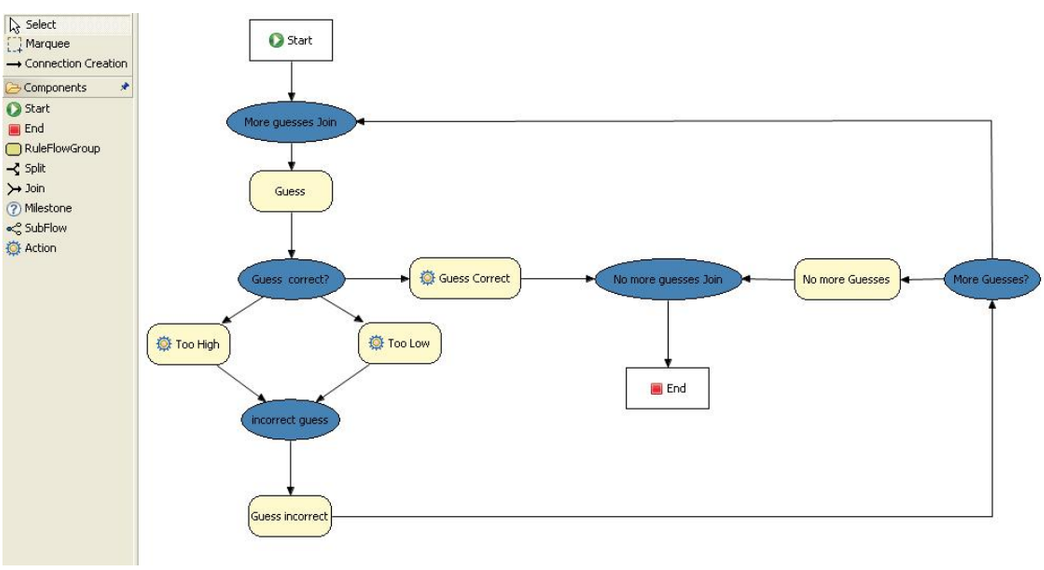


Figura 10: Rule Flow en Eclipse

BRMS (Business Rule Management System)

Este es un componente de Jboss Rules que permite administrar, almacenar, editar, y desplegar las reglas, como se aprecia en la Figura 11. El BRMS es una interfaz Web muy amigable que permite administrar las reglas por usuarios de distintos ambientes, como analistas del negocio, desarrolladores, etc.

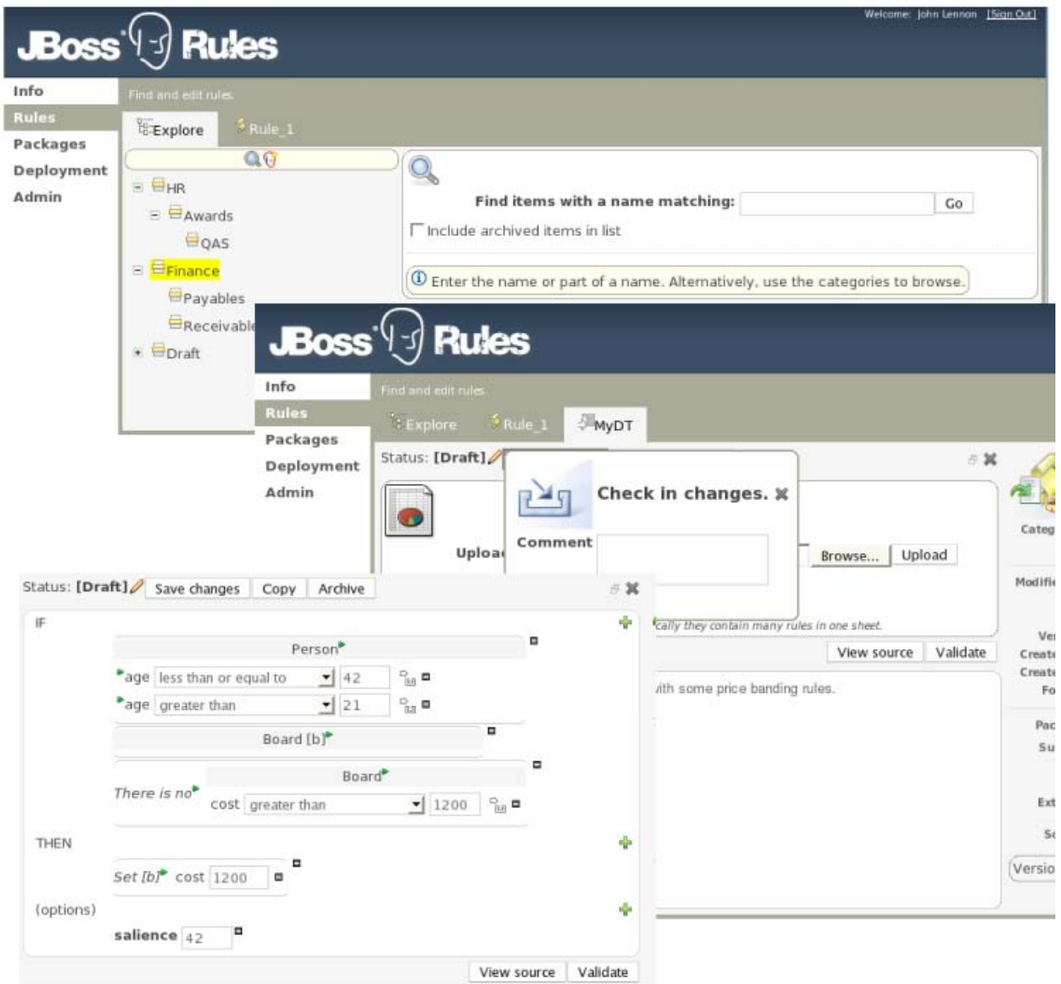


Figura 11: Business Rule Management System (BRMS)

Algunas características del BRMS.

- Múltiples editores de reglas.
- Control de versiones
- Permite exportar e importar paquetes de reglas.

- Categorización de reglas

La arquitectura del BRMS se muestra en la Figura 12:

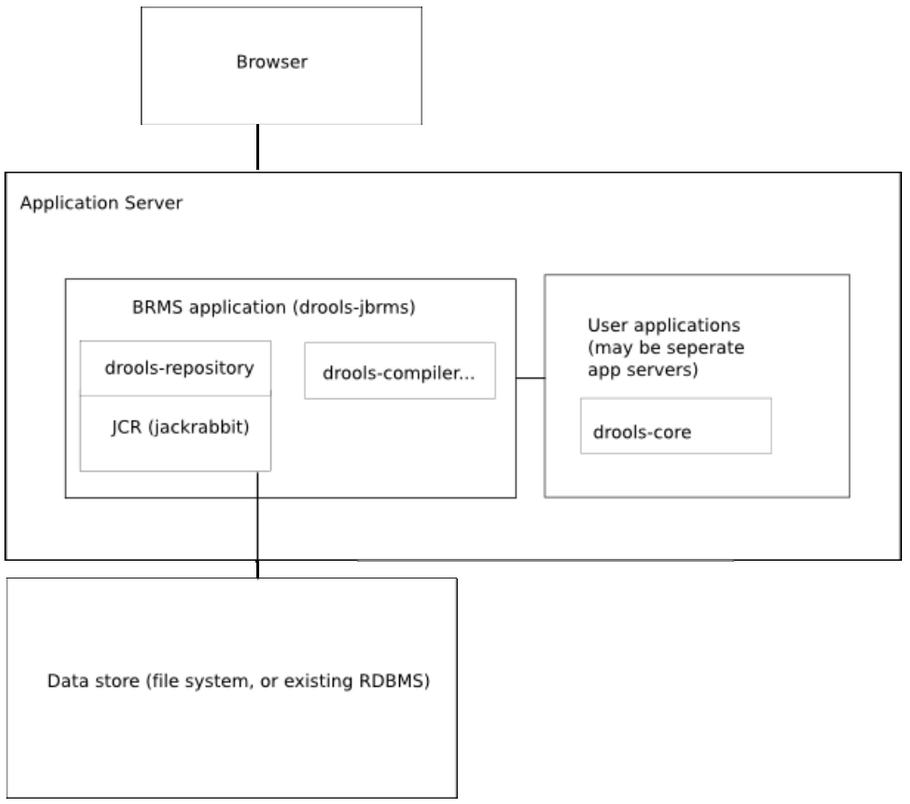


Figura 12: Arquitectura BRMS **[JBO]**

Decision Table

Jboss Rules soporta la administración de reglas en hojas de calculo (Excel, Open Office, etc.), el que mediante una plantilla permite crear, editar y administrar las reglas de negocio. Esta plantilla es invocada del entorno de desarrollo o del BRMS para evaluar las reglas que contiene.

3.6 Otras tecnologías

3.6.1 Entorno de desarrollo

La aplicación final desarrollada tiene varias capas, para las cuales es posible utilizar diferentes entornos de desarrollo, según las ventajas y desventajas que cada uno posea.

Para la construcción de la capa de comunicación (servicios Web) se pensó en la utilización de otros entornos como JDeveloper de Oracle, que posee las funcionalidades necesarias para el desarrollo de servicios Web. Sin embargo, se decidió utilizar Eclipse de manera de utilizar una única herramienta que soporte todo el desarrollo.

Eclipse

Eclipse es una plataforma de software sobre la cual se pueden desarrollar distintos tipos de aplicaciones. Esta herramienta es de código abierto y está basada en extensiones mediante las cuales la plataforma va adquiriendo distintas funcionalidades según sean las necesidades del usuario.

Una de las características que hacen que Eclipse sea uno de los IDE´s más utilizados es que éste provee las herramientas necesarias para desarrollar extensiones, por lo que cualquier usuario puede realizar las extensiones que estime convenientes, aumentando las capacidades del IDE a su gusto y necesidad.

Como se explicó anteriormente, Jboss Rules provee una extensión de Eclipse con funcionalidades avanzadas de administración de reglas que están ausentes en la interfaz Web. Estas funcionalidades facilitan aún más el desarrollo, entregando funcionalidades como la edición de reglas con intellisense[8] o la posibilidad de crear flujos de reglas basados en diagramas, lo cuales posteriormente pueden ser cargados en el BRMS.
A su vez, existen otras extensiones como WTP, la cual provee al desarrollador de editores XML y facilidades de integración con distintos servidores de aplicaciones J2EE, como Apache Tomcat, Jboss AS entre otros.

3.6.2 Balanceador de carga

Debido a la necesidad de contar con un motor de reglas que tuviese alta disponibilidad y mínimos tiempos de respuesta ante aplicaciones clientes que hacen un uso intensivo de éste, se optó por montar una arquitectura cluster para el servidor de

[8] Intellisense: Autocompletación implementada por editores para facilitar la programacion

aplicaciones. En esta arquitectura, un elemento fundamental es el balanceador de carga, el cual se encarga de distribuir eficientemente las peticiones de los clientes entre los distintos nodos del cluster del servidor de aplicaciones que contiene el motor de reglas.

Apache mod_jk

La manera más eficiente de realizar balance de carga en Jboss AS es mediante el modulo mod_jk de Apache Web Server. Este módulo está diseñado para interactuar directamente con Apache Tomcat, que finalmente es el núcleo del servidor de aplicaciones de Jboss.

Mod_jk trabaja con archivos de configuración, mediante los cuales se puede configurar diferentes propiedades del balanceo de carga, como la ubicación que tiene cada nodo dentro de la red, la prioridad de uso de cada nodo del cluster y que aplicaciones alojadas en el cluster están disponibles de manera paralela.

3.6.3 Motor de Servicios Web

Una parte importante en la implementación del motor de reglas, es establecer la forma en que éste se comunica con las aplicaciones que lo utilizarán. Para desarrollar esta capa de comunicación es necesario apoyarse en un motor de Servicios Web, el cual se encarga, entre otras cosas, de manejar los mensajes SOAP, hacer la transformación entre los datos que están en estos mensajes en XML a objetos Java y viceversa. Si bien el servidor de aplicaciones Jboss AS contiene su propio motor de Servicios Web, JBossWS, el uso de éste no cuenta con abundantes casos de éxito ni otorga las facilidades de desarrollo que entregan otros motores más populares, como Apache Axis2.

Apache Axis2/Java

Apache Axis2 es una implementación del estándar de Servicios Web basada en Java, tanto del lado del cliente como del servidor. Axis2 provee una arquitectura modular pensada en agregar fácilmente complementos que provean funcionalidades como WS-Security, WS-Addressing entre otros. Axis2 permite hacer lo siguiente:

- Enviar mensajes SOAP.

- Recibir y procesar mensajes SOAP.
- Crear un servicio Web desde una clase Java.
- Crear las clases de implementación tanto de un servicio como de un cliente a partir del WSDL.
- Enviar y recibir mensajes SOAP con archivos adjuntos.
- Crear o usar servicios Web basados en REST.
- Utilizar las recomendaciones WS-Security, WS-ReliableMessaging, WS-Addressing, WS-Coordination y WS-Atomic Transaction.

La utilización de Axis2 permite el manejo óptimo de la capa de comunicación del motor de reglas, facilitando su creación y permitiendo establecer políticas de seguridad en el envío de información, entre otras cosas.

3.6.4 Base de datos

Actualmente y por años, Telefónica del Sur, utiliza como gestor de base de datos Oracle 9i. Oracle en la actualidad es la referencia a seguir en el mundo de los gestores de bases de datos. Rápida, segura, profundamente administrable y configurable, robusta son sólo algunas de las características que hacen de Oracle la base de datos más demandada en los entornos empresariales. Además cuenta con un conjunto de aplicaciones y funcionalidades adicionales para añadir aún más valor a su paquete.

3.7 Factibilidad de Herramientas Seleccionadas

Con el fin de analizar el comportamiento del conjunto de herramientas seleccionadas, las cuales fueron descritas anteriormente, se decidió la implementación de un pequeño prototipo para decidir si era factible la implementación del motor de reglas de negocio, entre los factores de interés se consideró:

- Flexibilidad del lenguaje declarativo del motor seleccionado.
- Capacidad de la interfaz Web del motor frente a un caso real.
- Tiempos de respuesta asociados a la ejecución en un ambiente de pruebas físicamente distribuido.
- Tiempos de respuesta asociados al hardware involucrado en las pruebas.

Para esto se implementó la funcionalidad de validación de llamadas locales, la cual fue seleccionada por los siguientes factores:

- La estructura de esta funcionalidad de validación es la adecuada para la implementación en base a reglas, ya que en su mayoría las validaciones son de la forma IF-THEN con el retorno de un flag de aceptación o rechazo.
- La implementación de las validaciones es una parte importante de la tasación de llamadas, por lo tanto ésta es un aporte a la comprensión del sistema de tasación.

Para cada llamada a tasar se realiza una previa validación, la cual asegura que es un dato (registro) consistente para la futura tasación. En la Figura 13 se muestran las validaciones implementadas, con sus parámetros de entrada.

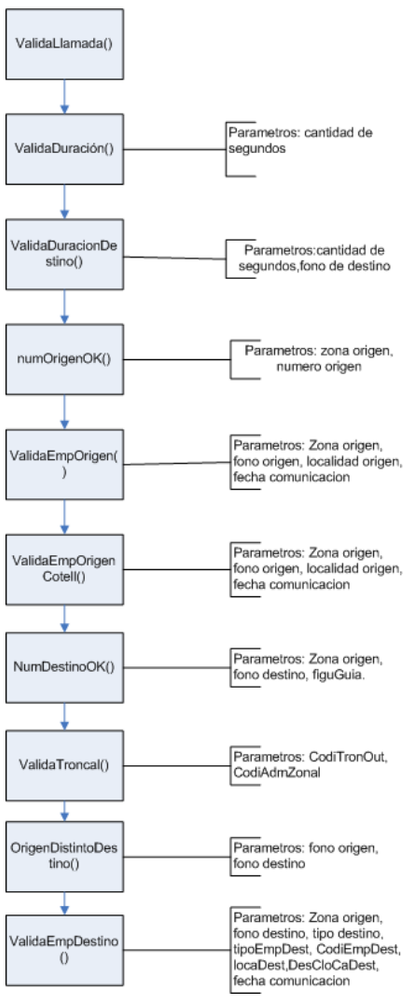


Figura 13: Validaciones implementadas en el prototipo

Los resultados obtenidos nos muestran que tanto la solución propuesta como las tecnologías utilizadas son factibles a la hora de implementar el motor de reglas de negocio, ya que se obtuvieron los mismos resultados en comparación a como se realiza actualmente y los tiempos de respuesta son aproximadamente iguales con un 10% de tolerancia por sobre o debajo de los tiempos obtenidos sin usar motor de reglas.

En la Tabla 5 se detallan los tiempos promedio en minutos obtenidos para cada cantidad se registros con un total de treinta y seis reglas implementadas.

Cantidad de Registros	Sin Motor de Reglas (actualmente)	Con Motor de Reglas
146203	47,5	46,1
50000	16,9	18,0
3370	1,3	1,1
1000	0,3	0,3

Tabla 5: Tiempos prototipo validación llamadas

CAPITULO IV: IMPLEMENTACION

4.1 Descripción del Capitulo

Ya demostrada la factibilidad del uso de Jboss Rules, en el presente capitulo se mostrará la implementación de un prototipo de sistema de tasación de llamadas locales basado en el motor de reglas y demás tecnologías seleccionadas. Concretamente, el capitulo contiene la descripción del sistema a modelar además de su respectiva documentación de análisis, diseño e implementación.

4.2 Conceptos Generales

En los siguientes capitulos se mencionan los siguientes términos, los cuales se aclaran a continuación:

SLM – Servicio Local Medido: El servicio local medido es la tarifa que se cobra por llamadas del tipo local.

Horario SLM: Cuando se habla de horarios SLM se está haciendo referencia a los tramos horarios en que Telefónica del Sur segmenta las llamadas locales. Estos son: Horario Normal, Horario Reducido y Horario Nocturno

TL-Tramo Local: El tramo local es el valor que se paga por la utilización de la línea de Telefonía local cuando llama a un Teléfono Móvil.

CACC-Cargo de Acceso: Son tarifas que carga la compañía telefónica a otras compañías de teléfonos por el uso de su red de telefonía.

Centrex: Es una red privada virtual de comunicaciones que permiten interconectar todas las sucursales y dependencias de una empresa como una central Telefónica. Toda la comunicación entre estas dependencias y sucursales no tienen costo ya que es vía anexo.

Planes de Servicio: Son aquellos servicios con valor agregado para empresas como lo son la línea 600 y 800, Servicio DDA, que se refiere al discado directo de anexos o telefonía IP.

Grupos de Facturación: Un grupo de facturación es un conjunto de servicios asociados a un mismo cliente, este conjunto de servicios es facturado de manera distinta que si los servicios fueran contratados individualmente, por lo tanto tienen un tratamiento especial a la hora de tasar su tráfico.

Código de Concepto: El código de concepto es un identificador único que representa el valor en pesos de un segundo de comunicación. El código de concepto a aplicar en una llamada depende del área tarifaria del teléfono de origen, y del horario en que se realizó la llamada.

DSL - Domain Specific Languages: Lenguaje declarativo que sirve para describir una situación en específico, ayudando a la comprensión de lo que se esta diciendo.

:

4.3 Especificación de requisitos

Como un primer paso para el modelado de la solución se hizo un levantamiento de requisitos, los cuales están separados en requisitos funcionales y no funcionales, y a la vez organizados de acuerdo a la iteración en que fueron implementados.

R1	Requisitos no funcionales
R1.1	Las reglas de negocio deben estar separadas del código fuente de la aplicación
R1.2	El tiempo de ejecución del sistema implementado no debe superar en un 50% el tiempo de respuesta del sistema actual.
R1.3	La comunicación entre el motor de reglas y la aplicación cliente debe ser realizada por medio de protocolos abiertos.
R1.4	El sistema debe proporcionar mecanismos de seguridad y encriptación de la comunicación para prevenir el uso malicioso de la información utilizada por el motor de reglas.
R1.5	Las reglas de negocio deben poder ser editadas vía Web
R1.6	El sistema debe asegurar que la comunicación entre el motor de reglas y la aplicación cliente no se vea afectada por la inclusión de sistemas de seguridad en la red
R1.7	El sistema debepermanecer disponible ante eventuales fallos del servidor.
R1.8	El conjunto de reglas asociadas al proceso de tasación de llamadas locales deben quedar a disposición de cualquier aplicación que necesite el servicio.

Tabla 6: Listado de Requisitos no Funcionales

R2	Requisitos funcionales generales
R2.1	El sistema debe ser capaz de efectuar la tasación en un periodo de tiempo diario

R2.2	El sistema debe diferenciar entre las llamadas que serán tasadas y las que no lo serán.
R2.3	El sistema tiene que permitir el ingreso de nuevas reglas de negocio
R2.4	El sistema debe permitir la edición y mantenimiento de las reglas existentes.
R2.5	El sistema debe ser capaz de tasar: las llamadas no tasadas de días anteriores, las llamadas con un perfil de cliente especifico y de un tipo de destino especifico (Local o Móvil)
R2.6	El sistema debe generar un log con la ejecución de la tasación, registrando si la tarea fue ejecutada con éxito o si se produjo algún error, cual error, las fechas de inicio y termino de las tareas, y cantidad de llamadas tasadas.
R2.7	El sistema debe registrar: el sentido de la llamada, el tipo de día, el día de la semana, el mes, la fecha y hora de inicio de la comunicación, la duración de la llamada, el cliente al que se le debe cobrar, el concepto que se le cobra y el valor correspondiente.

Tabla 7: Listado de requisitos funcionales generales

R3	Requisitos funcionales iteración 1
R3.1	El sistema debe registrar como no tasable las llamadas cuya duración sea inferior a tres segundos.
R3.2	El sistema debe registrar con valor tasable igual a cero, las llamadas cuyo número de origen y destino pertenezcan al mismo grupo centrex.
R3.3	El sistema debe registrar con valor tasable igual a cero, las llamadas de teléfonos cuya facilidad sea la de prepago.
R3.4	El sistema debe registrar con valor tasable igual a cero, las llamadas cuyo número destino sea un número frecuente.
R3.5	El sistema debe registrar con valor tasable igual a cero, las llamadas que sean originadas por un fono que tenga la facilidad de plan ilimitado

Tabla 8: Listado de Requisitos funcionales iteración 1

R4	Requisitos funcionales iteración 2 planes
R4.1	El paquete de reglas de tasación debe devolver el código de concepto por el cual se tasará la llamada, siendo este el código de concepto de cobro estándar o el código de concepto correspondiente a llamadas dentro de plan.
R4.2	Si una llamada sobrepasa uno o más tramos horarios, ésta debe ser dividida en dos o más comunicaciones, las cuales serán tasadas de manera independiente.
R4..3	Las llamadas que están dentro del tiempo contemplado para cada facilidad deben marcarse como tasadas con valor igual a cero.
R4.4	Si la duración de la llamada sobrepasa la cantidad de segundos disponibles en una facilidad contratada, esta debe ser dividida en 2 comunicaciones, la primera tendrá una duración que va desde su inicio hasta completada la cantidad de segundos de la facilidad y será tasada con valor 0, la segunda comunicación tendrá una duración desde el fin de la facilidad hasta el termino de de la llamada y será tasada con el valor correspondiente a su concepto.
R4.5	Si el cliente contrata o renuncia a un plan dentro del mes, la cantidad total de minutos con facilidad será de acuerdo a la proporción de días de vigencia de la facilidad en cuestión.

R4.6	El sistema debe registrar si la llamada fue efectuada en horario normal, reducido o vespertino.
R4.7	El sistema debe ir guardando los minutos consumidos de una facilidad por cada número telefónico.

Tabla 9: Listado de Requisitos funcionales iteración 2

R5	Requisitos funcionales iteración 3 tasación
R5.1	El sistema debe calcular el valor de la comunicación de acuerdo a la duración de ésta y valor del concepto por la que fue tasada.
R5.2	El sistema debe registrar las comunicaciones dentro de plan y fuera de plan.
R5.3	El sistema debe manejar los segundos ocupados de la facilidad en cada horario.
R5.4	Cuando la comunicación es a un teléfono móvil, el sistema debe registrar el cobro de tramo local y el cobro de cargo de acceso por separado.

Tabla 10: Listado de Requisitos funcionales iteración 3

4.4 Planificación

4.4.1 Metodología

La metodología utilizada para el desarrollo del proyecto fue Scrum. Esta metodología se categoriza dentro las llamadas metodologías agiles, las cuales destacan por ser capaces de ajustarse mas fácilmente a cambios en durante la ejecución del proyecto, además de permitir desarrollos incrementales donde el cliente puede ver avances reales en corto tiempo.

En Scrum los proyectos se dividen en partes llamadas Sprints las cuales tipicamente duran una, dos semanas o hasta un mes, periodo después del cual se obtiene un entregable y se planea la forma de abordar el siguiente Sprint, pudiendo organizar de forma concreta los siguientes pasos a seguir. Esta metodología tambien contempla reuniones de carácter diario al comienzo de la jornada, en las cuales se organizan las tareas que cada miembro del equipo debe realizar durante el día [SCR]. Lo anteriormente expuesto se puede visualizar de mejor forma en la figura 14.

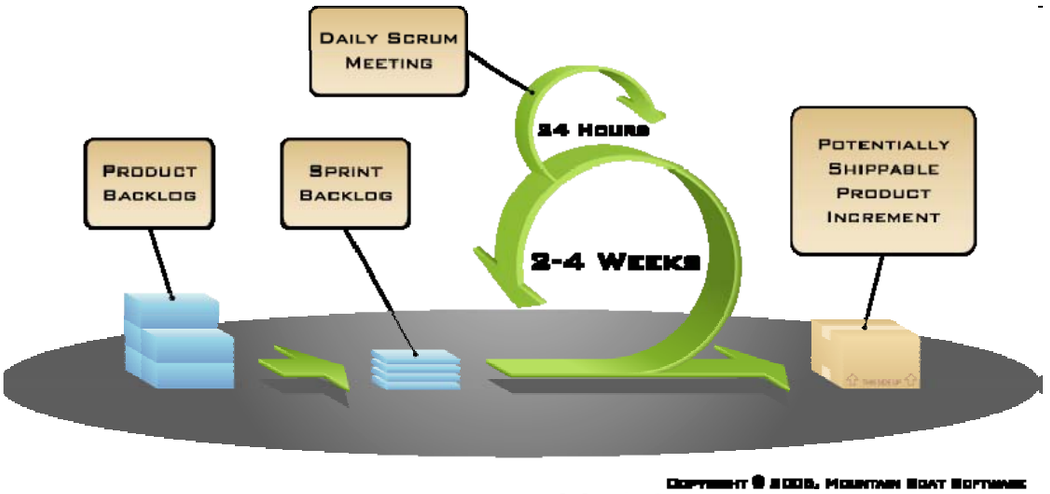


Figura 14**:** Metodología Scrum

Se decidio utilizar esta metodología porque el problema es divisible en conjuntos de reglas, y por lo tanto implementables en distintas iteraciones, donde al final de cada una se evaluó las distintas dificultades y se planeó cómo organizar de mejor manera la siguiente iteración.

4.4.2 Diagrama de Gantt

En la siguiente figura se puede encontrar un diagrama de Gantt resumido, que muestra el proceso de desarrollo real del proyecto.

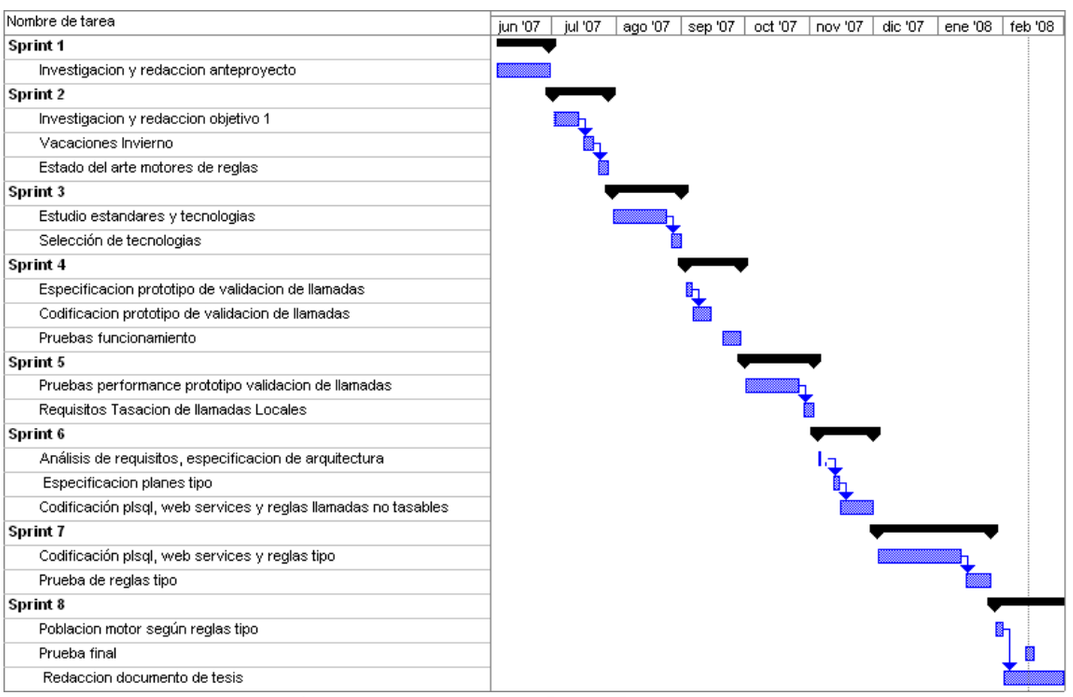


Figura 15: Resumen Carta Gantt Proyecto

4.5 Análisis

4.5.1 Descripción del sistema actual y solución propuesta

Actualmente, el sistema de tasación de llamadas locales se basa en un conjunto de procesos codificados en procedimientos almacenados en Oracle. A grandes rasgos, se puede decir que éstos son cuatro:

- La carga de datos (locp_tar_detalle), es un proceso que se ejecuta de manera manual, cargando las comunicaciones locales y móviles de las distintas zonales de la compañía. Este proceso realiza validaciones básicas de consistencia de datos y tasa todas las comunicaciones indistintamente de acuerdo a su tipo y tarifas vigentes.
- La postcarga (locp_tar_postcarga), que se encarga de tomar las llamadas tasadas en el proceso anterior y saca de facturación los tráficos que por omisión no deben ser tasados, estos tráficos son:
    - Llamadas menores a 4 segundos
    - Centrex
    - Prepago
    - Descuentos a empresas
    - Llamadas a carabineros
    - Rentas planas
    - Números frecuentes.

  Este proceso corre de manera mensual y se ejecuta una vez que estén realizadas todas las cargas (proceso anterior) del mes correspondiente.
- Descuentos (locp_tar_descuentos), Este proceso se ejecuta después de la postcarga y su función es tasar en cero las llamadas que estén cubiertas por las facilidades contratadas por el cliente, estas facilidades pueden ser al fono, al rut o a un grupo de facturación. También se encarga de tasar en cero los tráficos de planes ONNET y OFFNET.
- Planes PHS (locp_tar_planesphs), Este proceso aplica todos los planes que están creados para teléfonos PHS, planes con control, planes sin control. Los planes pueden ser aplicados con minutos para cada tipo de horario (Normal, Económico, Vespertino) o en todo horario.

Problemática

Estos procesos mencionados anteriormente, apoyados con un conjunto de tablas, concentran las reglas de negocio las cuales son aplicadas por los procedimientos. Además, estos procesos corren de manera mensual y, por esta razón, el cliente no tiene acceso al detalle de su cuenta hasta fin de mes, lo cual no le permite llevar un control de la utilización de las facilidades que tiene contratadas. En la figura siguiente se puede apreciar de mejor forma el flujo de información actual.

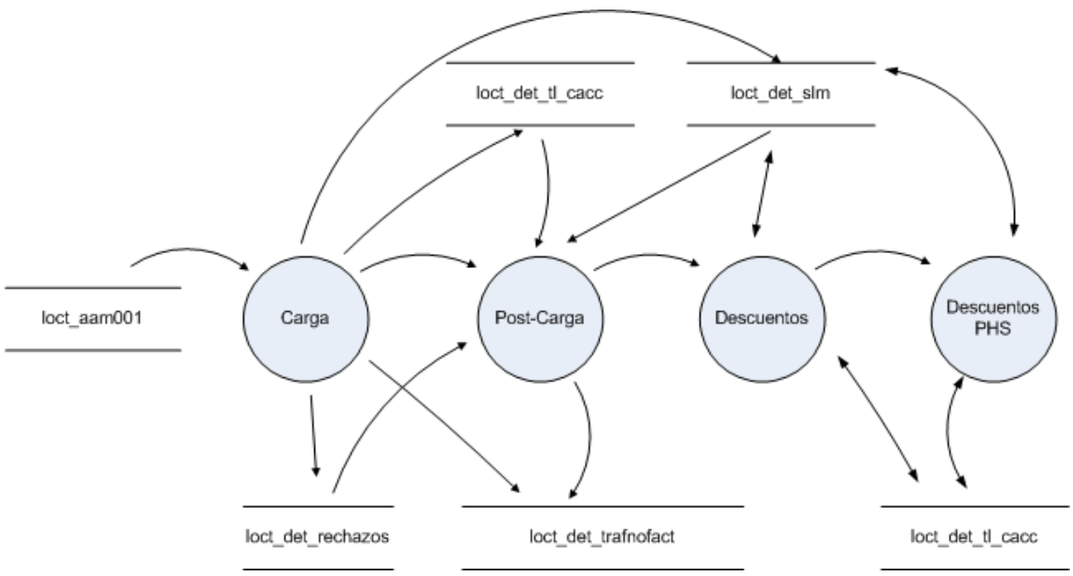


Figura 16: Diagrama de Flujo de datos Actual

Solución propuesta

En la solución propuesta, se considera la existencia de un único proceso que realice la tasación diaria, este proceso alimentará al motor de reglas con los datos necesarios para obtener la correcta valorización para cada una de las comunicaciones lo que afecta al flujo del proceso actual

Ademas de lo anteriormente descrito se propone modificar la forma en que se almacena el resultado de la tasación, registrando los resultados en una única tabla, a través de la cual estará disponible para el sistema de facturación y para la eventual implementación de un sistema de información de clientes mediante el cual estos puedan revisar el estado de sus facilidades. En la Figura 17 se puede apreciar el diagrama de flujo de datos propuesto y en la Figura 18 un diagrama de secuencia del sistema.

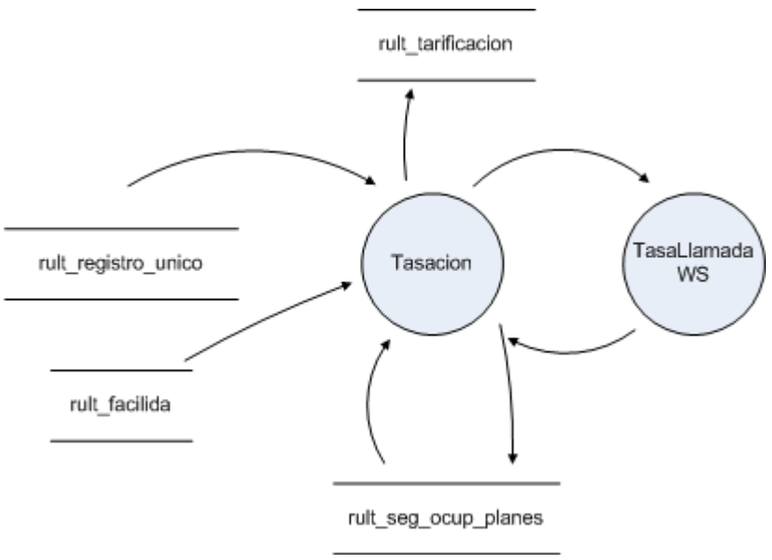


Figura 17: Diagrama de Flujo Propuesto

Diagrama de secuencia del sistema

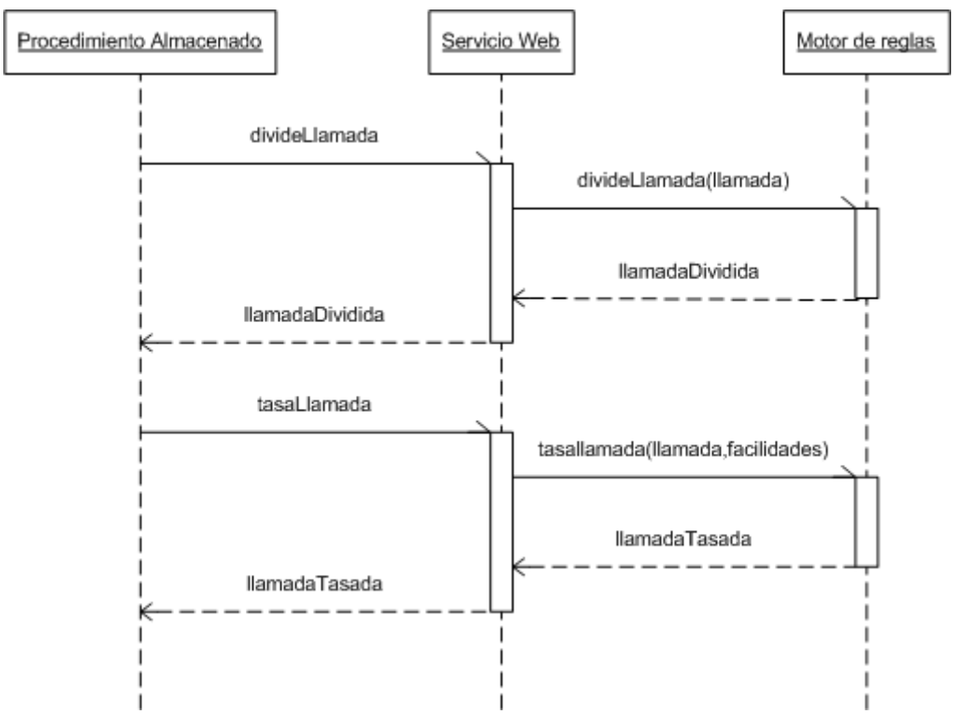


Figura 18: Diagrama de secuencia propuesto

Las tablas rult_registro_unico y rult_facilida alimentan el proceso de tasación con las llamadas efectuadas en el periodo y las facilidades que posee la llamada. Para efectos de este prototipo se consideraran las reglas correspondientes a facilidades de un rut. Las facilidades correspondientes a empresas y grupos de facturación no se

implementaran. Una vez ejecutado el proceso de tasación en la tabla rult_seg_ocup_planes se registran la cantidad de segundos ocupados de un cliente por cada una de sus facilidades en un mes específico. El resultado de la tasación se registra únicamente en rult_tarificacion. Una mayor descripción de cada una de las tablas se puede encontrar en el capitulo 7.1 del anexo.

La implantación de esta solución tiene varios beneficios:

- El ingreso de una nueva facilidad o la modificación de una ya creada y la creación de reglas, se realiza de manera gráfica por medio de una interfaz Web capaz de manejar y crear las reglas
- Dichas modificaciones pueden ser realizadas por personal externo a informática, liberando la carga de este departamento.
- La tasación diaria da pie a ofrecer un nuevo servicio al cliente, otorgando un valor agregado que va en favor de la fidelidad del cliente a la compañía.
- El volumen de código se ve reducido, debido a que la mayor parte de la lógica se encuentra codificada en forma de reglas, lo cual lleva a una rápida comprensión y mantenibilidad del sistema.

La incorporación de esta tecnología también tiene algunos costos y desventajas:

- Se necesita capacitación a los usuarios de negocio en cuanto a la lógica de especificación de reglas y el uso de la interfaz Web.
- La interfaz de comunicación entre el motor de reglas y el proceso de tasación está construida a base de servicios Web, por lo que se necesita conocimiento de esta tecnología por parte del personal que la mantiene en este caso el area de informatica.
- El uso de servicios Web como interfaz involucra un aumento en los tiempos de respuesta frente al actual sistema basado en procedimientos almacenados, lo cual lleva a un aumento en el consumo de recursos de hardware para compensar el aumento de tiempo.

4.5.2 Descripción de planes tipo

Los planes modelados en la implementación del sistema de reglas de negocio fueron los correspondientes a llamadas a teléfonos locales (tráfico SLM) y a teléfonos móviles.

El detalle de los “planes tipo” se muestra en la Tabla 11, donde se menciona el nombre del tipo de plan, a qué teléfonos se le aplica y cuál es el tráfico que no se factura de las llamadas dentro de plan, el que es marcado con una equis “x”.

Tipo Plan	**Aplicación Teléfono**	**Trafico No Facturable**		
		SLM	**Móvil**	**TL**
Plan	Fijo	X		
Mixto	PHS	X	X	X
Onnet	Fijo	X		
Prepago	Fijo, PHS	X	X	X
Renta Plana	Fijo, PHS	X		
SLMQLP	PHS	X	X	X

Tabla 11: Descripción Planes Tipos

4.6 Diseño

4.6.1 Diagrama de secuencia

La Figura 19 muestra la interacción que se produce entre los distintos objetos. Por un lado están los “packages”, que se aprecian al lado izquierdo de la figura, que se comunican a través de los procedimientos almacenados, y en el lado derecho el servicio Web, en el que se da la interacción entre distintas clases para lograr la ejecución del motor de reglas.

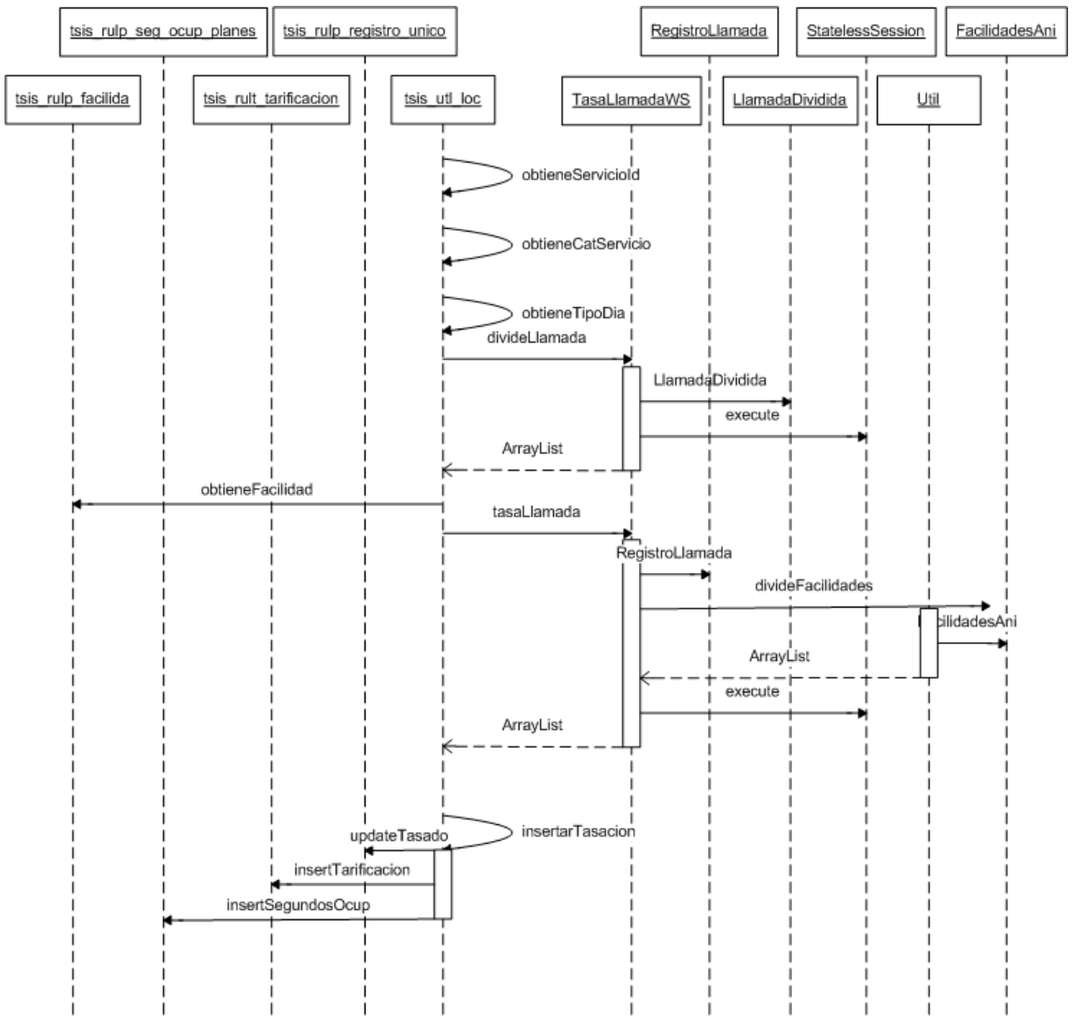


Figura 19: Diagrama de Secuencia

4.6.2 Diagrama de clases

Como el sistema fue desarrollado en dos tecnologías distintas, una orientada a objetos y otra procedural, se tomó la opción de organizar los “packages” de tal forma de que cada uno agrupe las funciones de acceso a cada una de las tablas que utiliza el sistema y de esta forma se logró relacionar ambas tecnologías y llegar a un único diagrama de clases, el cual es mostrado en la Figura 20.

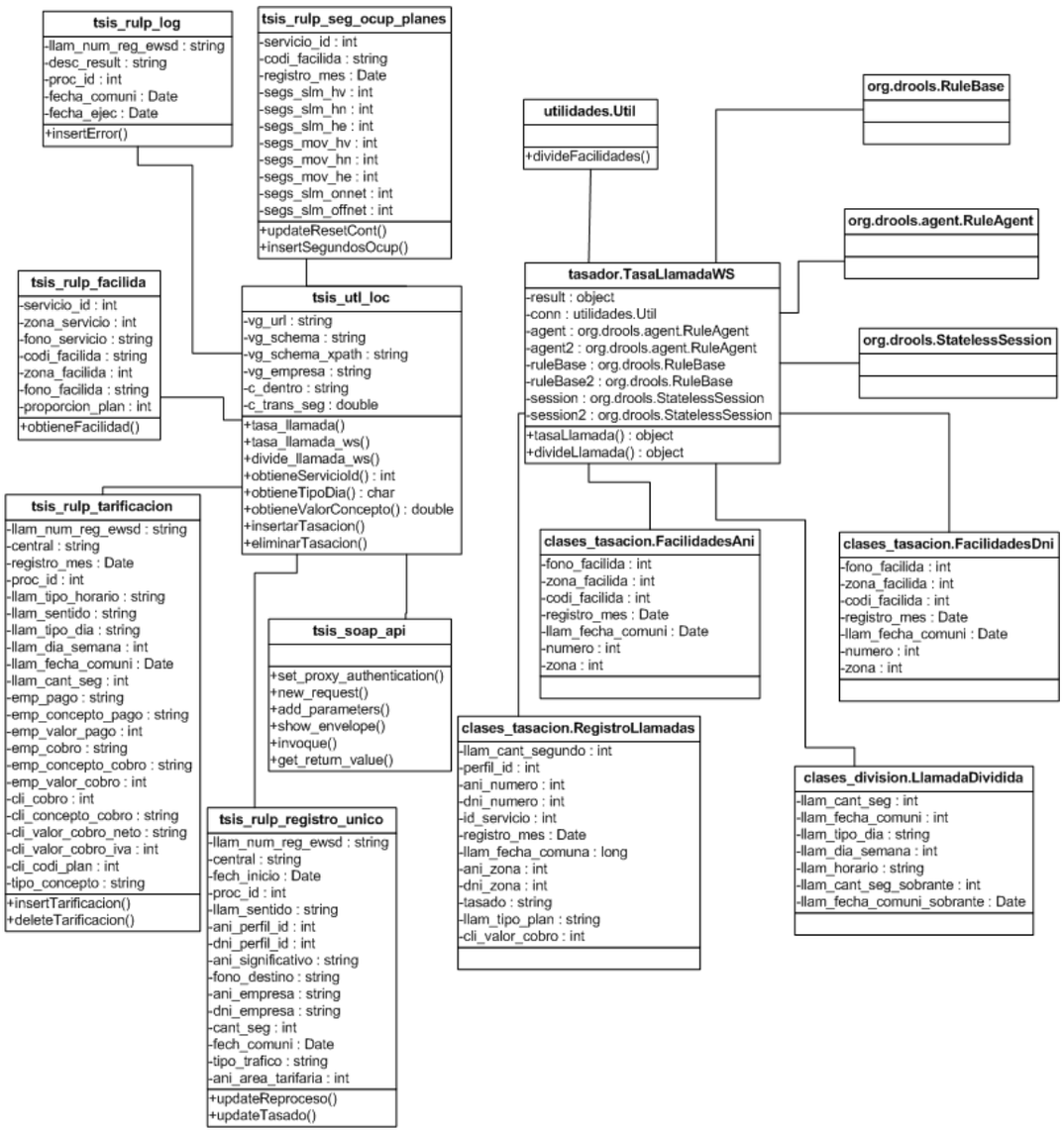


Figura 20: Diagrama de Clases

4.6.3 Diagrama de componentes

Uno de los requisitos no funcionales del sistema hace referencia a la capacidad del motor de reglas de comunicarse a través de protocolos abiertos con el resto de las aplicaciones, esto debido a la amplia gama de tecnologías que conviven en los sistemas de la empresa. Por esto se decidió utilizar una arquitectura orientada a servicios como lo muestra la Figura 21, en la cual cada paquete de reglas es publicado como una función de un servicio Web, quedando este disponible para cualquier aplicación que lo necesite.

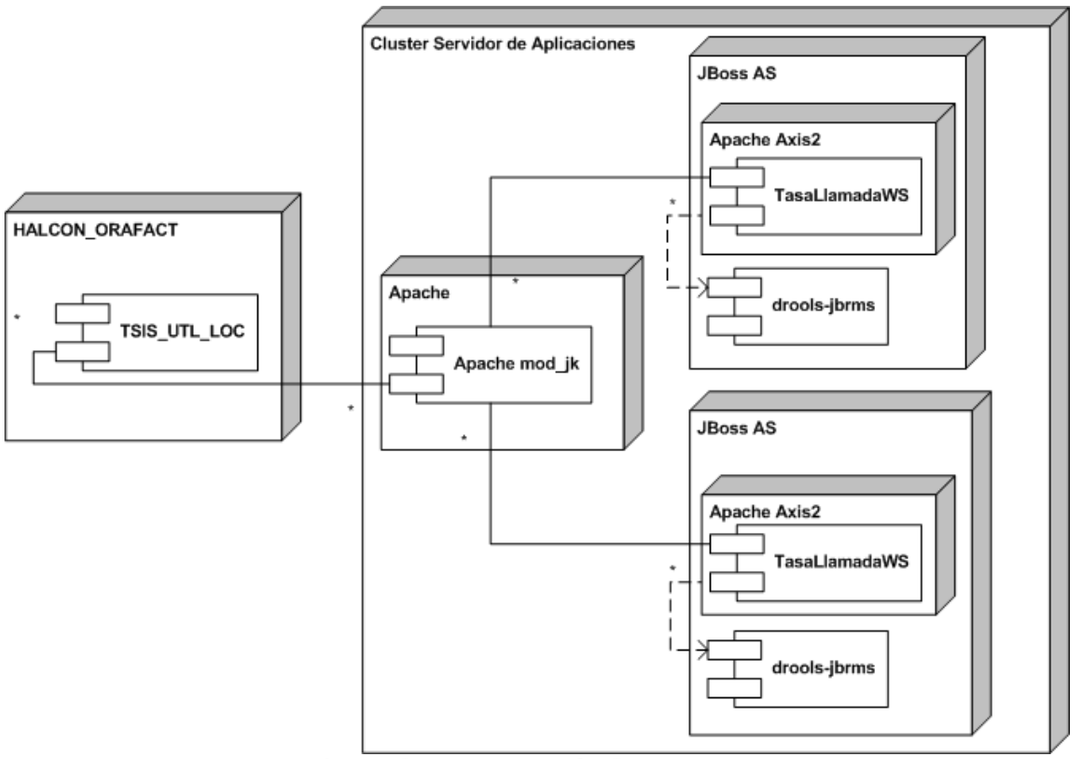


Figura 21: Diagrama de Componentes

Como se puede ver en la figura, para el tasador de llamadas locales se optó por implementar un cluster del servidor de aplicaciones JBoss AS, ya que la utilización de servicios Web hace que el proceso haga un uso extensivo tanto de recursos de red como de procesamiento, y de esta forma se logra una velocidad adecuada para lograr el objetivo de la tasación diaria. Otra ventaja de la disposición en cluster es que reduce la probabilidad de falla, en este caso al fallar alguno de los nodos, el mediador desvía automáticamente las peticiones al nodo que está funcionando, reduciendo la probabilidad de que no se ejecute de manera satisfactoria la tasación.

4.6.4 Especificación de Servicios Web

Para la implementación del tasador de llamadas locales se utilizaron dos paquetes de reglas y cada uno de ellos es expuesto por un método en el servicio Web que se detalla a continuación.

TasaLlamadaWS

Servicio Web que contiene los servicios de decisión que permiten la tasación de llamadas locales. Este servicio contiene dos funciones que se encargan de exponer los dos paquetes de reglas creados para tal efecto:

TasaLlamada		
Descripción	Método que recibe los parámetros necesarios para tasar una comunicación y retorna el código de concepto correspondiente.	
Nombre Parámetro	**Tipo de dato**	**Descripción**
ani_numero	**Integer**	Numero de origen
dni_numero	**Integer**	Numero de destino
llam_cant_seg	**Integer**	Cantidad de segundos de la comunicación
llam_horario	**String**	Horario en que se realizo la comunicación (V=vespertino, N= normal, E= económico)
llam_tipo_destino	**String**	Tipo de destino de llamada (local, móvil)
llam_ani_area_tarifaria	**Integer**	Área tarifaria del fono de origen
ani_empresa_numero	**String**	Empresa del numero de origen
dni_empresa_numero	**String**	Empresa del numero de destino
cat_servicio	**String**	Categoría del servicio (PHS = 38)
Facilidades	**String**	String separado por espacios y punto y coma que contiene las facilidades del cliente y cuanto lleva usado de cada una de ellas
Return	**Object[]**	Arreglo de objetos que contiene: - Si llamada fue tasada o no (>4 seg. o no) - Cantidad segundos dentro plan - Cantidad segundos fuera plan - Concepto Cobro - Concepto de Cobro cargo acceso - Código de facilidad - Tipo Fono (Fijo, móvil) - Flag que indica si la llamada pertenece a un plan on,off-net

Tabla 12: Descripción servicio Web de tasa llamada

divideLlamada		
Descripción	Método que recibe los parámetros para decidir el tramo horario al que pertenece una llamada, si esta se extiende por mas de un tramo, el método retorna la cantidad de segundos en el tramo que se tasará y la cantidad de segundos y fecha de inicio del tramo siguiente.	
Nombre Parámetro	**Tipo de dato**	**Descripción**
llam_cant_seg	**Integer**	Cantidad de segundos que duro la comunicación
llam_fecha_comuni	**String**	Fecha de inicio de la comunicación
llam_tipo_dia	**String**	Tipo de día en que se realizo la comunicación (H= Hábil, F= Festivo)

llam_dia_semana	**int**	Día de la semana en que se inicio la comunicación (1 = Lunes, 7 = Domingo)
Return	**Object[]**	Arreglo de objetos que contiene: - Fecha de la comunicación a tasar - Cantidad de segundos de la comunicación a tasar - Horario de la comunicación a tasar (Vespertino, Normal, Económico) - Fecha inicio de la comunicación sobrante, es el horario de inicio del siguiente tramo. Si la comunicación pasa de un día a otro, de un mes a otro, o de un año a otro, la fecha también se ve afectada. - Cantidad de segundos que están en el horario siguiente. Esta cantidad podría sobrepasar el tramo horario siguiente, pero esta junto a la fecha de comunicación sobrante se vuelven a dividir para verificar esto.

Tabla 13: Descripción servicio Web de divide llamada

4.7 Implementación

4.7.1 Implementación motor de reglas

En esta sección se mostrará la lógica con que se implementó las reglas de tasación así como una descripción del funcionamiento de la interfaz Web.

4.7.1.1 Descripción estructura de reglas tipo

Existen dos paquetes de reglas implementados en el BRMS que son:

DivideLlamadaTasacion:

Este paquete de reglas tiene por objetivo dividir por tramo horario las llamadas, es decir, si una llamada forma parte de N horarios, esa llamada se divide en N llamadas respectivamente.

TasacionLlamadas:

Este paquete a su vez, lo podemos dividir en tres partes (conjunto de reglas), por prioridad de ejecución.

- Reglas Primera Prioridad:

Corresponden al conjunto de reglas que tienen la primera prioridad de ejecución, ya que ellas determinan el valor de muchos objetos que más tarde se ocuparan en las siguientes reglas. Este conjunto de reglas fueron implementadas con el objetivo de definir el tipo de llamada que entra, por ejemplo si es una llamada a móvil o a teléfono fijo, definir los conceptos por cargo de acceso de llamadas a celulares y a red rurales, entre otras.

- Reglas Prioridad Intermedia

Este conjunto de reglas detallan los planes de la compañía cuyo tráfico (SLM, TL y Cargo de Acceso) no se tasa de acuerdo al tipo de plan, los cuales poseen cierta cantidad de minutos gratis en algún horario, ya sea, horario normal, económico, vespertino o todo horario. Si el cliente tiene una facilidad y la llamada sobrepasa el límite de duración de esa facilidad, las reglas dividen la comunicación en dos y retorna la tasación correspondiente a la parte dentro de plan y la cantidad de segundos sobrantes para ser tasados nuevamente

- Reglas Última Prioridad

Son aquellas reglas para tasar las llamadas que están fuera de plan o para los clientes que no tienen ningún plan contratado, además de las llamadas que no están contempladas dentro de plan, como son las llamadas a líneas 600 y llamadas a red rural.

La descripción en detalle de las Reglas Tipo, con los objetos de entrada y salida se ven en el capitulo 7.2 del anexo.

4.7.1.2 Implementación de Reglas en BRMS

La página inicial de administración mostrada por el BRMS para la creación de las reglas, una vez logeado, se puede ver en la Figura 22.

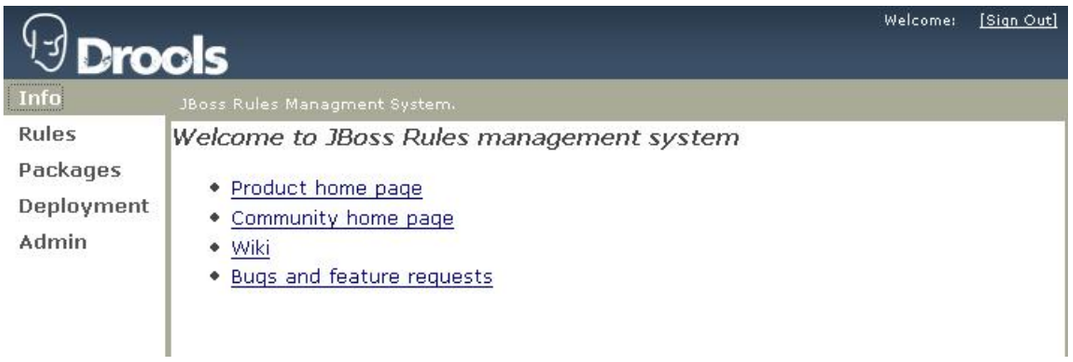


Figura 22: BRMS, Pagina Principal

A continuación se detalla cada etapa necesaria en la creación e implementación de cada regla:

Creación de categorías

La creación de categorías tiene por objetivo agrupar las reglas que son comunes, con el fin de dar un mejor ordenamiento a las reglas. Para ello es necesario ir al menú "Admin" de la figura anterior, el que despliega la página de administración de categorías que se muestra en la Figura 23.

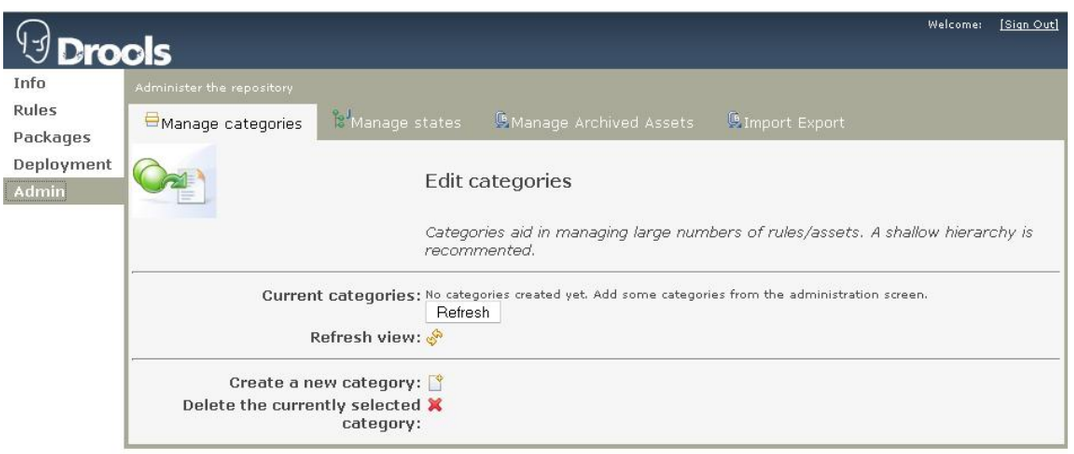


Figura 23: BRMS, Creación de Categorías

Para crear la categoría pinchamos en "Create a new category" el que nos despliega una ventana donde se coloca el nombre de la categoría que queremos crear y una descripción de la misma. Se procede de la misma forma si queremos crear más categorías o sub-categorías. En la figura siguiente se muestra una estructura de categorías como ejemplo.

Figura 24: Ejemplo de Estructura de Categorías en BRMS

Las categorías implementadas para la tasación local en esta tesis fueron:

- No Tasables: agrupa las reglas cuyas llamadas no están contempladas dentro de algún plan y no son tasables.

- Planes SLM: contiene las reglas de las llamadas dentro de plan cuyo tráfico SLM no se tasa.

- Planes Mixtos: agrupa las reglas de las llamadas dentro de plan cuyo trafico SLM, TL y Cargo de Acceso (CACC) no se tasan.

- Planes Onnet: contiene las reglas de las llamadas dentro de plan Onnet (llamadas entre teléfonos de la compañía) y planes Offnet (llamadas a teléfonos de otras compañías), donde no se factura el trafico SLM.

- Planes SLMQLP: agrupa las reglas de las llamadas dentro de plan cuyo tráfico SLM, TL y Cargo de Acceso (CACC) en todo horario no se facturan. Valido para teléfonos PHS.

- Planes Renta Plana: contiene las reglas de las llamadas dentro de plan renta plana, donde no se cobra ningún trafico .

- Planes Estándar SLM: agrupa aquellas reglas cuyas llamadas SLM están fuera de plan o no tienen plan, cuyo tráfico SLM se factura.

- Planes Estándar Móvil: agrupa aquellas reglas cuyas llamadas a móviles están fuera de plan o no tienen plan, cuyo tráfico TL y CACC se facturan.

- Planes Estándar 600: contiene aquellas reglas para las llamadas a línea 600, cuyo tráfico se factura de acuerdo al código de concepto correspondiente.

- Planes Estándar Rural: agrupa aquellas reglas para las llamadas a red rural, cuyo tráfico TL y CACC se facturan

Dentro de las categorías de planes antes mencionados, excluyendo las categorías "Estándar", hay dos sub-tipos de categorías que son:

- Dentro de Plan: Engloba todas las reglas para llamadas que están dentro de la cantidad de minutos del plan correspondiente.

- Fuera de Plan: Reglas para aquellas llamadas cuya duración excede la cantidad de minutos del plan, teniendo una cantidad de segundos dentro y fuera de plan, las cuales tienen un valor distinto para efectos de la tasación

Creación de Packages

Luego de crear las categorías, es necesaria la creación de los "packages" que contienen las reglas, DSL[9], las funciones y el modelo de clases, que es un archivo jar, para ello se pincha el menú "Packages" el que despliega una página donde muestra una barra con iconos en la parte superior como lo muestra la Figura 25, en ella pinchamos el ícono  para crear el package, al que le daremos un nombre, en

[9] Domain Specific Languages

nuestro caso creamos dos, uno llamado DivisionLlamadas y otro llamado TasacionPrueba, como se aprecia en la Figura 26.

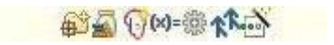


Figura 25: Barra de explorador de reglas en BRMS

Figura 26: Vista de Pakages en BRMS

Creación DSL

Antes de llevar a cabo la implementación de las reglas de negocio es necesario crear el lenguaje DSL, el que nos proporciona la creación de reglas en lenguaje común. Para ello debemos pinchar el icono  de la barra que muestra la Figura 25, el que despliega una ventana donde se da el nombre al lenguaje, en nuestro caso se le llamo "Tasación" y finalmente muestra la pagina donde se escribe el lenguaje DSL, que se aprecia en la Figura 27.

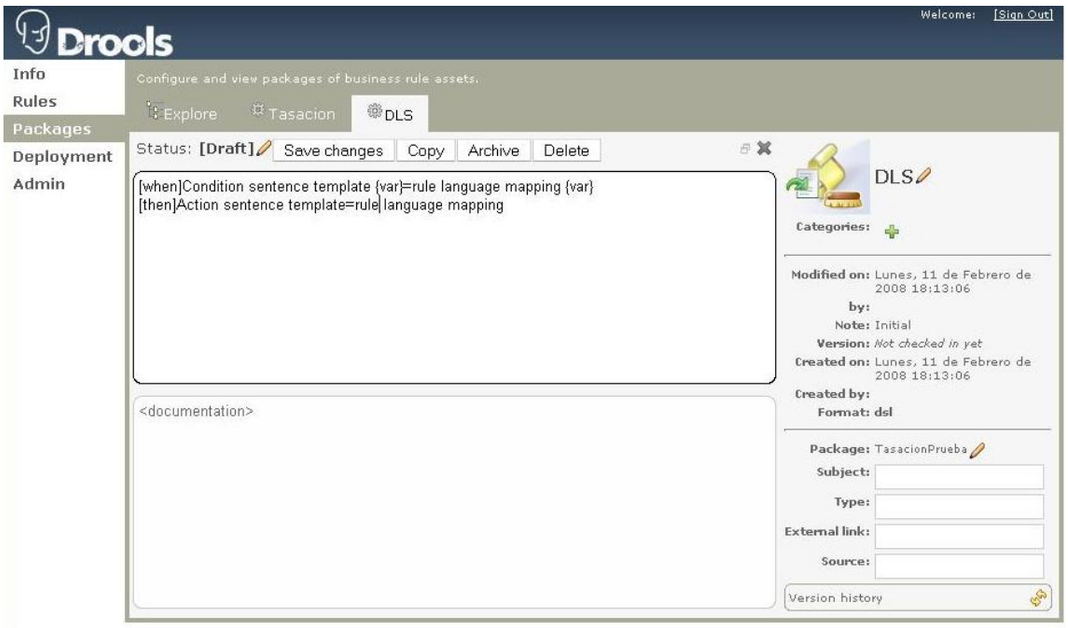


Figura 27: Página de Creación de DSL en BRMS

La estructura para la creación de cada regla es la siguiente.

```
[when/then] - Lenguaje común = lenguaje de reglas de jboss rules
```

A **B** **C** **D** **E**

Donde:

A: La sentencia dentro de estos caracteres [] nos dice si la expresión es parte de LHS o RHS de la regla, siendo un “when” o “then” respectivamente.

B: El caracter “-” es opcional, cuyo objetivo es separar expresiones para un mejor entendimiento de las reglas. Para usarlo es necesario declarar antes en una sentencia la clase java que será usada por las sentencias con éste caracter.

C: Corresponde a la cara visible, para los usuarios, de la expresión para la creación de la regla. Para ocupar valores de entrada en las reglas se ocupa el caracter de llaves {}, quedando su valor en la variable que esta dentro de ellas.

D: Es el delimitador entre el lenguaje común y el lenguaje de reglas.

E: Corresponde al lenguaje proporcionado por Jboss Rules para escribir la regla de negocio, dependiendo si se esta hablando de LHS o RHS, ya que en este ultimo, permite sentencias Java.

A modo de ejemplificar lo descrito anteriormente se muestra un ejemplo sencillo de escribir una regla en lenguaje DSL, la que nos dice que cuando una llamada sea menor a cierta cantidad se segundos definida por el usuario, en este caso cuatro segundos, entonces no se cobra esa llamada, retornando una variable con una “N” como valor:

```
[when]En una llamada = r:RegistroLlamadas()
[when]-La llamada es menor a {segundos} segundos= cantidad_segundos < {segundos}
[then]No se cobra la llamada = r.setCantidad_segundos(“N”); Update(r);
```

La regla escrita en lenguaje DSL queda de la siguiente forma:

```
When
      En una llamada
         -La llamada es menor a 4 segundos
Then
      No se cobra la llamada
```

La regla de negocio, para la interpretación del motor de reglas queda de la siguiente forma:

```
When
      r:RegistroLlamadas(cantidad_segundos < 4)

then
      r.setCantidad_segundos(“N”);
      Update(r);
```

De esta forma se escribe todo el lenguaje apropiado para detallar las reglas de negocio que se necesiten implementar. Todo el lenguaje DSL escrito en este proyecto se detalla en el capitulo 7.3 del anexo.

Creación de reglas

El BRMS ofrece la interfaz que muestra la Figura 28 para la creación de reglas.

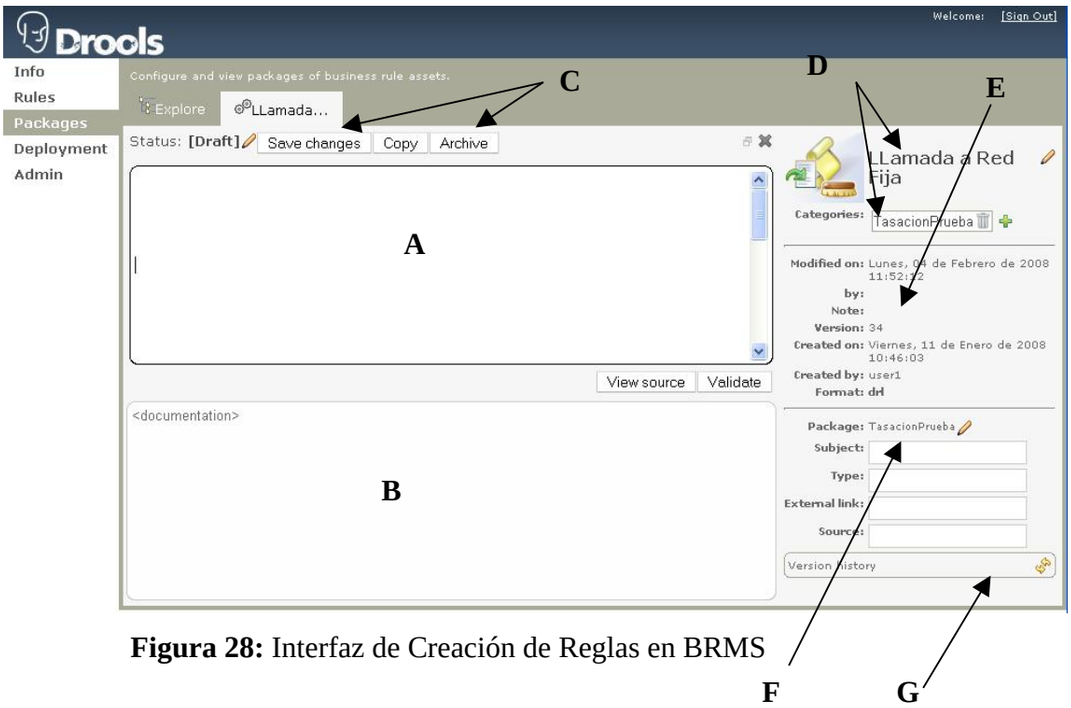


Figura 28: Interfaz de Creación de Reglas en BRMS

Donde:

A: Editor para las reglas, donde se escribe el lenguaje de reglas de jboss rules o en DSL, este editor varía de acuerdo al formato de regla que uno desea crear

B: Corresponde al área de documentación de la regla.

C: Son las acciones para guardar, copiar y archivar (borrar) la regla creada.

D: Corresponde al nombre de la regla junto a la categoría a la que pertenece, pudiendo también modificar ambas cosas.

E: Contiene información relevante a la creación o modificación de la regla.

F: Muestra el "package" al que pertenece la regla.

G: Muestra las versiones históricas de la regla, pudiendo ir a una versión mas antigua si es necesario.

Además el BRMS ofrece distintas formas de crear una regla de negocio al pinchar el icono de la barra que se muestra en la Figura 25, el que despliega una ventana donde se le coloca el nombre de la regla, se selecciona la categoría a la que pertenece y el package, además de elegir el formato con que se quiere crear la regla, los que se detallan a continuación:

DRL Rule (Technical Rule- Text Editor), este formato es más técnico para la creación de reglas, es un simple editor, donde se van escribiendo las reglas con el lenguaje proporcionado por jboss rules ocupando los objetos correspondientes.

El editor de este formato lo observamos en la Figura 28, obteniendo reglas de la forma como se muestra en la siguiente Figura.

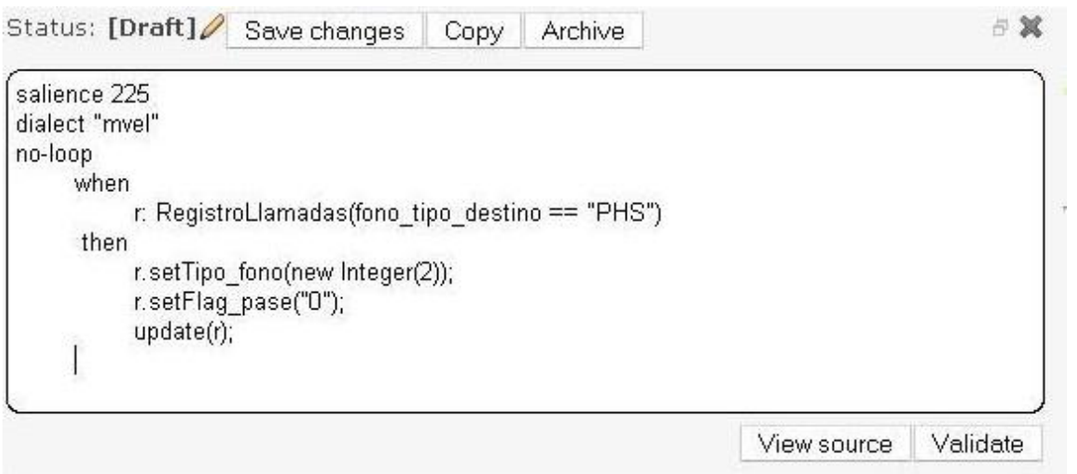


Figura 29: Ejemplo de Creación de Reglas con DRL Rule

Business Rules Using Dsl, formato para escribir las reglas ocupando el lenguaje dsl. El icono nos despliega las sentencias de condición para la regla, mientras que el icono despliega las sentencias de acción, formando así la estructura de la regla como apreciamos en la Figura 30.

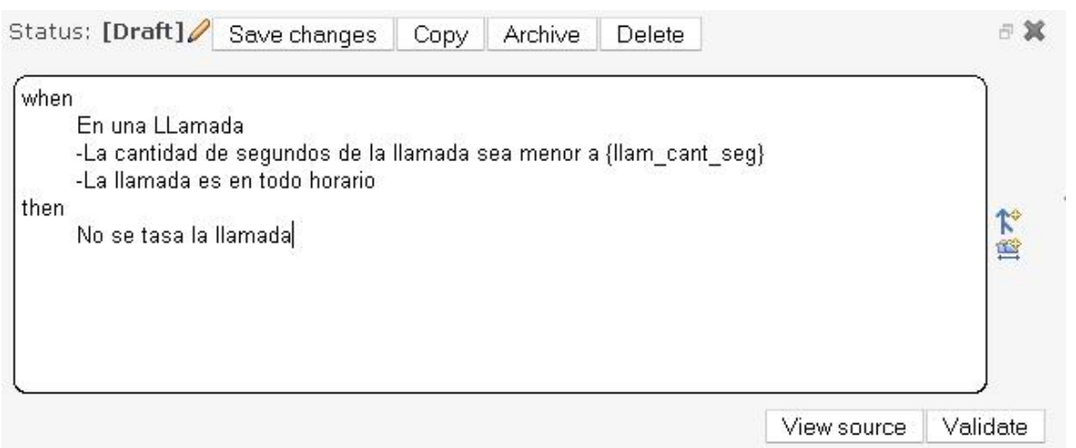


Figura 30: Ejemplo de Creación de Reglas con Bussines Rules Using Dsl

Business Rules-using Rule Editor, este formato engloba los dos anteriores, permite crear las reglas con la ayuda de un editor guiado, el que puede ser ocupando el DSL o las clases y variables del modelo que se tiene, su interfaz se aprecia en la Figura 31.

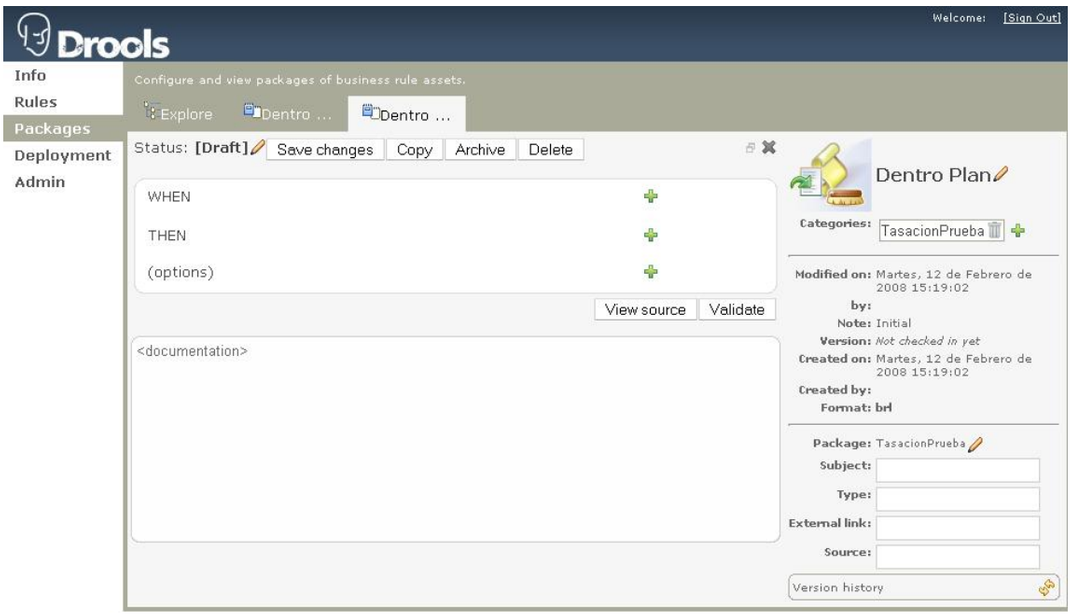


Figura 31: Interfaz de creación de reglas con Business Rules-using Rule Editor

La anterior muestra la vista inicial de este formato, el cual al pinchar el icono nos despliega una ventana con las opciones de ayuda del editor como se aprecia en la Figura 32, donde podemos elegir las sentencias DSL o los objetos que tenemos en el modelo para crear la reglas.

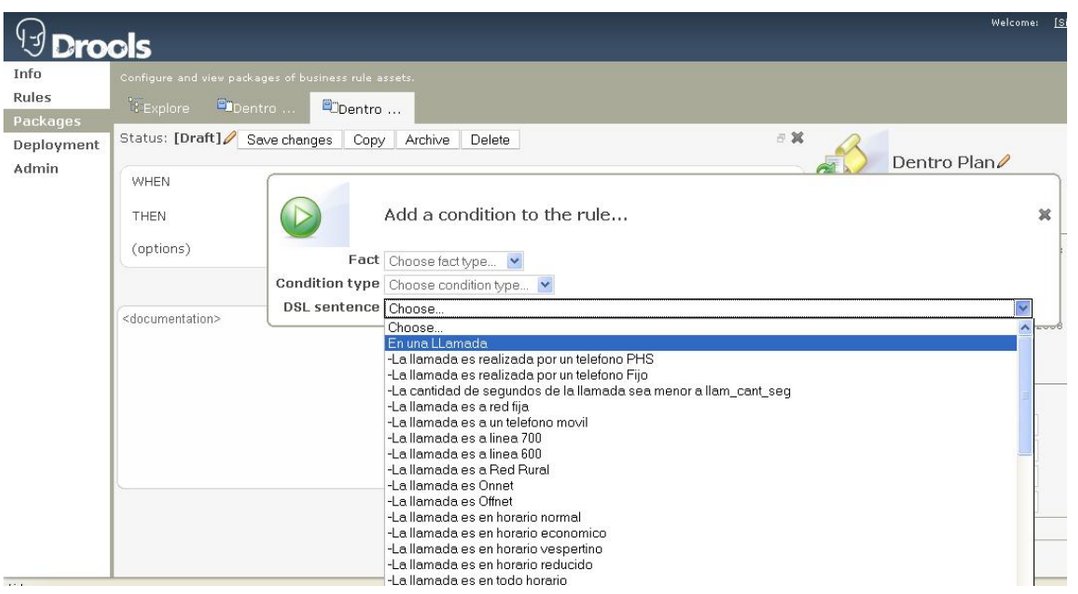


Figura 32: Ejemplo de agregar condiciones a una regla usando Rule Editor

De esta forma generamos las reglas, quedando de la forma como se aprecia en la Figura 33 al ocupar las sentencias DSL o como se observa en la Figura 34 cuando ocupamos los objetos de nuestro modelo.

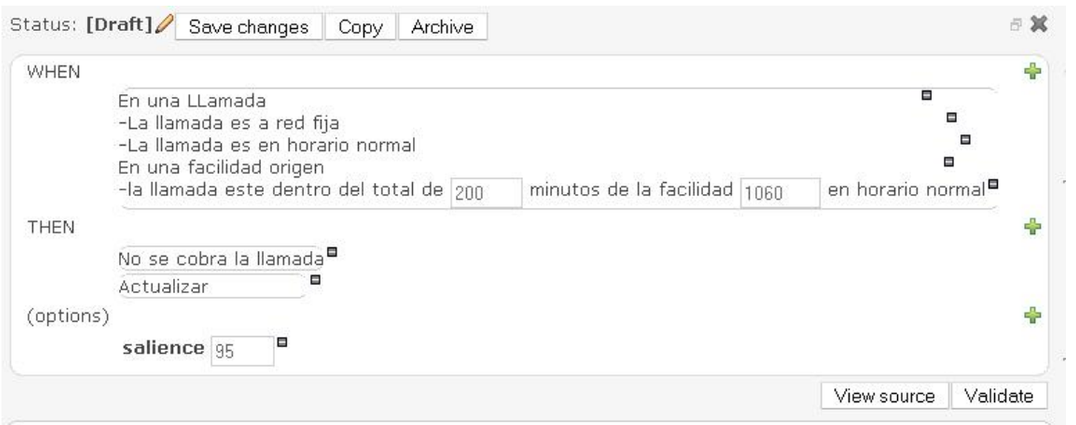


Figura 33: Ejemplo de reglas con Rule editor usando DSL

Figura 34: Ejemplo de reglas con Rule editor usando objetos de nuestras clases java

En todos los formatos de creación de reglas descritos anteriormente está la opción de validar nuestra regla, para ello hay que pinchar el botón con el nombre “Validate” el que nos dirá si está todo correcto o hay algún error. También está el botón “View Source” el que nos muestra el código nativo de la regla, es decir, el que es interpretado por el motor de reglas.

Construir el Package Binario

Teniendo escritas y validadas las reglas sólo basta construir el package binario con las reglas, para ello pinchamos en “Package”, seleccionamos el que nos interesa y pinchamos el botón que dice “Build Package” como se aprecia en la Figura 35

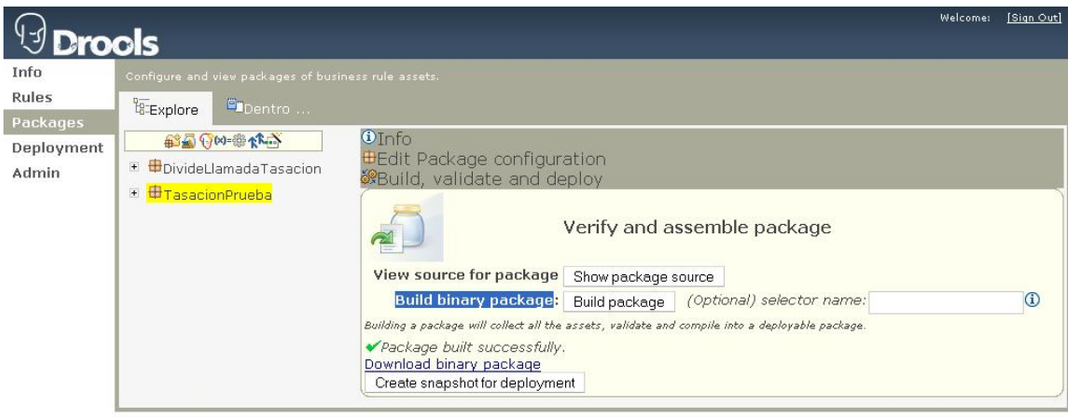


Figura 35: Generación de Package Binario

Luego creamos su Snapshot pinchando el botón: “Create snapshot deployment” y con esto hemos finalizado la creación de reglas.

Mayor detalle de las funcionalidades del BRMS y del lenguaje DSL se puede revisar en la documentación de Jboss rules [JBO] y en el anexo, capitulo 7.3, respectivamente

4.7.1.3 Tabla resumen reglas implementadas

Un resumen por package con la cantidad de reglas de negocio por categoría, se muestra a continuación:

Package “DivideLlamadaTasacion”

Categoría	Total
División Horario	16

Tabla 14: Reglas por categoría package “DivideLlamadaTasacion”

Package “TasacionPrueba”

Categoría	Dentro Plan	Divide Llamada	Fuera Plan	Sub total
Auxiliares	38	-	-	38

No Tasables	5	-	-	5
Planes SLM	88	88	54	230
Planes Mixtos	26	25	-	51
Planes Onnet	15	8	-	23
Planes Estándar SLM	21	-	-	21
Planes Estándar Móvil	33	-	-	33
Planes Estándar 600	21	-	-	21
Planes SLMQLP	40	40	-	80
Planes Renta Plan	28	21	-	49
Planes Estándar Rural	33	-	-	33
			Total	584

Tabla 15: Reglas por categoría del package “TasacionPrueba”

4.7.2 Implementación PL/SQL

En esta sección se muestra como los procedimientos almacenados interactúan con el servicio Web que invoca al motor de reglas.

El procedimiento tasa_llamada_ws es el encargado de invocar al método tasaLlamada del servicio Web, el cual utiliza el motor de reglas para tasar una comunicación de acuerdo a los datos que se le envían desde la base de datos. La llamada a este procedimiento se realiza como sigue:

```
tasa_llamada_ws (p_ani_numero in varchar2,
            p_dni_numero in varchar2,
            p_ani_servicio_id in number,
            p_dni_servicio_id in number,
            p_llam_cant_seg in number,
            p_llam_tipo_horario in varchar2,
            p_llam_tipo_destino in varchar2,
            p_llam_ani_area_tarifaria in number,
            p_registro_mes in date,
            p_ani_empresa_numero in varchar2,
            p_dni_empresa_numero in varchar2,
            p_cat_servicio in varchar2,
            p_tasado out varchar2,
            p_cant_seg_dentplan out number,
```

```
p_cant_seg_fueraplan out number,
p_cli_concepto_cobro out varchar2,
p_cli_concepto_cobro_cacc out varchar2,
p_codigo_facilidad out number,
p_fono_tipo out number,
p_onoff out number);
```

Los parámetros declarados como “in” son los que contienen la información que se necesita para tasar una comunicación, mientras que los declarados como “out” son los que reciben el resultado de la tasación. Para realizar la invocación del servicio se utilizó la librería SOAP_API, la cual mediante funciones se encarga de ensamblar el XML que se enviará al servicio, invocarlo, y rescatar los valores del XML de respuesta del servicio. El siguiente segmento de código muestra como se agrega un parámetro de entrada al servicio:

```
tsis_soap_api.add_parameter(ol_req,'llam_cant_seg','Integer',p_llam_can
t_seg);
```

En este caso la variable “ol_req” es la que almacena el XML que se está ensamblando, el texto “lam_cant_seg” es el nombre del nodo XML, “Integer” es el tipo de dato de la variable y “p_llam_cant_seg” es el valor de la variable. El resultado de la ejecución de esta función es la concatenación a “ol_req” del nodo XML que se muestra a continuación:

```
<llam_cant_seg xsi:type="Integer"> valor </llam_cant_seg>
```

Una vez que son agregados todos los parámetros de entrada, se procede a invocar al servicio Web mediante la función “invoke” de “soap_api”, los parámetros son: el XML de entrada donde están los parámetros, en este caso “ol_req”; la URL del servicio Web, que en este caso es la variable global “vg_url”; y el nombre de la función del servicio que se desea invocar que para el caso mostrado es tasaLlamada.

```
ol_resp := tsis_soap_api.invoke(ol_req,vg_url,v_funcion);
```

El XML de respuesta del servicio Web queda almacenado en “ol_resp”. Para obtener su valor existe la función “get_return_value” de “soap_api” la cual se utiliza como sigue.

```
p_cli_concepto_cobro:=tsis_soap_api.get_return_value(ol_resp,
'return[1]',vg_schema_xpath);
```

En este caso el parámetro “ol_resp” contiene el XML de respuesta, el parámetro “return[1]” le indica a la función que obtenga el valor del primer Nodo XML y la variable global “vg_schema_xpath” contiene el XML schema de la respuesta del servicio.

La invocación del servicio divideLlamada sigue la misma lógica, pero reuniendo la información que se necesita para esa operación. Una descripción detallada de los procedimientos se puede ver en el apartado 7.1.2 del capitulo Anexo.

4.8 Instalación y configuración

Para la implementación del motor de reglas es necesario instalar y configurar las siguientes herramientas con sus versiones, como se detalla a continuación.

4.8.1 Servidor Web Apache

Para la instalación de este servidor no se requiere ningún conocimiento especial, ya que se lleva a cabo mediante un wizard, salvo a la hora de configurarlo para determinados servicios que se requieran como el que se detalla a continuación, la versión ocupada en este proyecto es la 2.2.4.

4.8.2 Modulo Apache mod_jk

Para la instalación de este módulo es necesario bajar la librería mod_jk-apache[APA], la que depende del servidor apache que tengamos instalado, en nuestro caso es la versión mod_jk-apache-2.2.4.so.

A continuación, se modifica el nombre del archivo mencionado anteriormente el cual llamaremos: mod_jk.so y lo copiaremos en la ruta de nuestro servidor apache: <APACHE_HOME>/modules

Una vez copiado, abriremos el archivo de configuración de apache: <APACHE_HOME>/conf/httpd.conf e incluiremos la siguiente línea al final del

archivo:

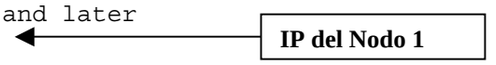
```
#Include mod_jk configuration file
Include conf/mod-jk.conf
```

A continuación se crea el archivo mod-jk.conf en la ruta: <APACHE_HOME>/conf como se indicó en el archivo de configuración anterior. El contenido del archivo es:

```
# Load mod_jk module
# Specify the filename of the mod_jk lib
LoadModule jk_module modules/mod_jk.so

# Where to find workers.properties
JkWorkersFile conf/workers.properties

# Where to put jk logs
JkLogFile logs/mod_jk.log

# Set the jk log level [debug/error/info]
JkLogLevel info

# Select the log format
JkLogStampFormat "[%a %b %d %H:%M:%S %Y]"

# JkOptions indicates to send SSK KEY SIZE
JkOptions +ForwardKeySize +ForwardURICompat -ForwardDirectories

# JkRequestLogFormat
JkRequestLogFormat "%w %V %T"

# Mount your applications
JkMount /application/* loadbalancer

# You can use external file for mount points.
# It will be checked for updates each 60 seconds.
# The format of the file is: /url=worker
# /examples/*=loadbalancer
JkMountFile conf/uriworkermap.properties

# Add shared memory.
# This directive is present with 1.2.10 and
# later versions of mod_jk, and is needed for
# for load balancing to work properly
JkShmFile logs/jk.shm

# Add jkstatus for managing runtime data
<Location /jkstatus/>
JkMount status
Order deny,allow
Deny from all
Allow from 127.0.0.1
</Location>
```

Ahora, creamos el archivo workers.properties en el directorio <APACHE_HOME>/conf. En este archivo se configura la dirección ip de los nodos en los que vamos a balancear la carga.

```
# Define list of workers that will be used
# for mapping requests
# The configuration directives are valid
# for the mod_jk version 1.2.18 and later
worker.list=loadbalancer,status
```



```
# Definimos el nodo Nodo1
# Puerto del conector ajp de nuestro tomcat (JBoss)
worker.nodo1.port=8009
# Ip del nodo 1.
worker.nodo1.host=192.168.1.7
worker.nodo1.type=ajp13
# Peso de nuestro nodo. A más peso, más peticiones
worker.nodo1.lbfactor=1

# Definimos el nodo Nodo2
worker.nodo2.port=8009
worker.nodo2.host=192.168.1.13
worker.nodo2.type=ajp13
worker.nodo2.lbfactor=1

# Load-balancing behavior
worker.loadbalancer.type=lb
worker.loadbalancer.balance_workers=nodo1,nodo2

# Status worker for managing load balancer
worker.status.type=status
```

IP del Nodo 2

Por último creamos el archivo uriworkermap.properties en la ruta <APACHE_HOME>/conf, en donde se configuran las aplicaciones que vamos a balancear, en nuestro caso es la aplicación “WSTasadorLlamadas”.

```
# Simple worker configuration file
# Mount the Servlet context to the ajp13 worker
/jmx-console=loadbalancer
/jmx-console/*=loadbalancer
/web-console=loadbalancer
/web-console/*=loadbalancer
/WSTasadorLlamadas=loadbalancer

/WSTasadorLlamadas/*=loadbalancer
```

Con esto tenemos instalado y configurado Apache y el módulo mod_jk. Para comprobar nuestra instalación ejecutamos el monitor del servidor apache el cual debe mostrarnos en la parte inferior que el mod_jk está cargado, como se aprecia en la figura siguiente.

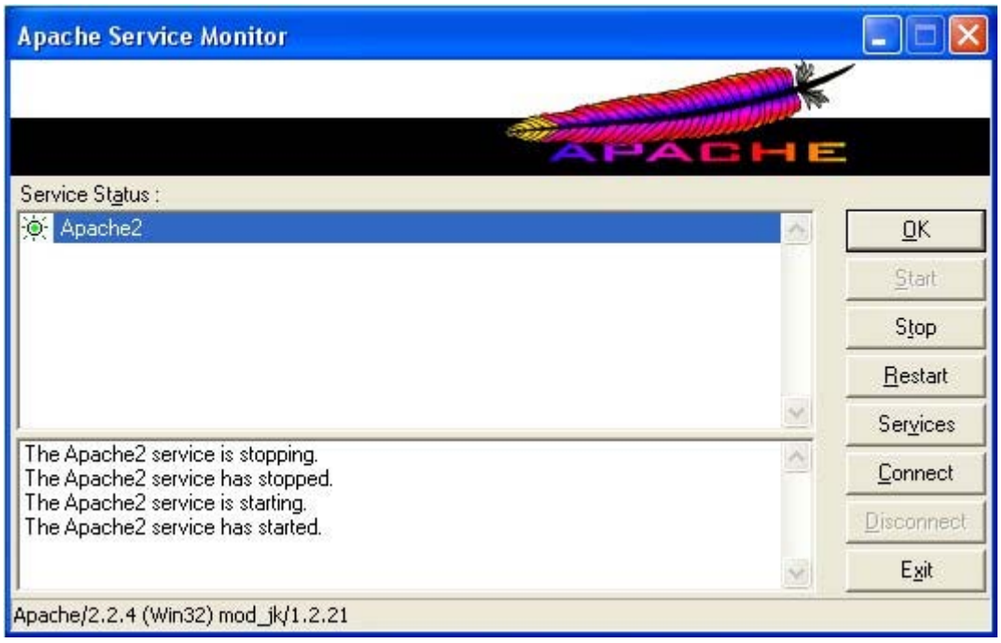


Figura 36: Monitor del servidor apache

4.8.3 JBoss AS

Este servidor de aplicaciones lo descargamos de la página oficial de Jboss [JBO], eligiendo la versión que queremos como servidor de aplicaciones. En este proyecto se trabajó con la versión jboss-4.2.3.GA.

Una vez descargado nuestro servidor, sólo nos queda copiarlo en el directorio designado para nuestro servidor de aplicaciones.

Para trabajar en cluster, tenemos que trabajar con la configuración “all” del servidor JBoss, es decir ejecutarlo con el comando “run.bat -b 0.0.0.0 -c all”.

4.8.4 JBoss rules IDE plug-in

En este proyecto se trabajó con esta funcionalidad a modo de probar sus funcionalidades y bondades, no dejando de ser importante ya que permite las mismas funcionalidades que el BRMS, pero en un entorno de desarrollo Java, ejemplo Eclipse, estando orientado a los profesionales de la informática a diferencia del BRMS.

La versión del plug-in es “Drools 4.0.7 Eclipse 3.2 Workbench” que se descarga de la página oficial de Jboss Rules [JBO] y su instalación corresponde básicamente en copiar el archivo “org.drools.eclipse_4.0.7.jar” en la carpeta: <Eclipse_Home>/plugins.

4.8.5 JBRMS

El BRMS lo obtenemos de la dirección [JBO], el que una vez descargado lo copiamos en nuestro servidor de aplicaciones Jboss-4.2.3.GA en la ruta: <JBOSS_HOME>/jboss-4.2.3.GA\server\default\deploy y con esto esta instalado nuestro BRMS con las configuraciones por defecto,, al que accedemos desde la direccion: http://localhost:8080/drools-jbrms. La versión del BRMS ocupada en este proyecto es “Drools 4.0.7 BRMS.”

4.9 Validación

En este apartado se detalla el plan de prueba que se llevo a cabo junto con su objetivo, para luego en el siguiente punto, dar a conocer los resultados por cada objetivo descrito en el plan de pruebas.

4.9.1 Plan de pruebas

El plan de pruebas se centra en validar que las reglas de negocio implementadas representen las reglas reales en las que la empresa basa sus operaciones a la hora de tasar llamadas locales. Las validaciones por paquete de reglas fueron las siguientes:

Paquete de división de horarios

Se creó una comunicación ficticia realizada por un número ficticio. Luego se envió al motor dicha comunicación variando los siguientes parámetros:

- Hora de realización de la llamada
  Objetivo 1: Probar la correcta determinación de los distintos horarios, Vespertino, Normal y Económico
- Duración de la comunicación.

Objetivo 2: Verificar que el motor divida correctamente la comunicación de acuerdo a los tramos horarios establecidos en las reglas.

El paquete de reglas de tasación de llamadas es bastante más extenso, por lo cual su desarrollo se dividió en tres iteraciones, en cada una de las cuales se crearon reglas que representan a los distintos tipos de planes. Una vez terminadas las reglas tipo, se procedió a verificar su correcto funcionamiento y una vez probadas se pobló el motor con las reglas de los demás planes basadas en las reglas tipo.

Paquete de tasación de llamadas con reglas tipo

Se creó una comunicación ficticia realizada por un número ficticio al cual se le asoció la facilidad tipo que se desea probar. Luego se envió dicha comunicación al motor de reglas variando los siguientes parámetros:

- Horario de la comunicación
  Objetivo 3: Verificar que la comunicación sea tasada en el horario correcto
- Destino de la comunicación
  Objetivo 4: En el caso de las facilidades que poseen minutos a móviles, verificar que la llamada sea tasada de acuerdo a los conceptos correspondientes.
- Minutos consumidos de la facilidad
  Objetivo 5: Verificar que se tasa correctamente cuando la comunicación esta dentro de plan, fuera de plan, o con una parte dentro de plan y la otra fuera.

Paquete de tasación de llamadas con el total de reglas

Se contó con acceso a la tabla rult_registro_unico y rult_facilida con datos reales, ejecutando la tasación de un día determinado con el nuevo tasador de llamadas locales.

Objetivo 6: Comparar el resultado de la tasación utilizando el motor de reglas versus el resultado obtenido con el sistema actual. La comparación se realizó tomando en cuenta clientes que posean solo facilidades implementadas en el motor de reglas.

Objetivo 7: Evaluar los tiempos de respuesta obtenidos con el total de reglas implementados en relación a los tiempos obtenidos sin motor de reglas.

4.9.2 Resultados

Los resultados por objetivos descritos en el punto anterior fueron los siguientes

Objetivo 1: El motor determina correctamente el horario en que se realiza la llamada basado en los tramos horarios vigentes, es capaz de determinar si la llamada corresponde a horario Normal, Económico, o Vespertino según el día de la semana en que se realiza la llamada.

Objetivo 2: El motor determina correctamente la fecha de inicio de la comunicación a tasar, la cantidad de segundos que se tasaran, y el tramo horario al que pertenece la comunicación a tasar. Además, la hora de inicio del siguiente tramo de la comunicación y la cantidad de segundos restantes. Cuando la llamada no se divide, la cantidad de segundos restantes es 0 segundos.

Objetivo 3: La llamada disparó la regla correspondiente al horario en que fue realizada, por lo tanto, se obtuvo el código de concepto correcto.

Objetivo 4: En el caso en que se probó con destino móvil, el motor retornó el código de concepto de tramo local y de cargo de acceso correctamente, en el caso en que el número de destino es fijo retornó el código de concepto SLM correspondiente y el código de cargo de acceso con un valor por defecto igual al carácter equis “x”.

Objetivo 5: Cuando la comunicación está dentro de plan, el motor retornó el código de concepto correspondiente a llamadas dentro de plan, el cual tiene un valor 0 y el código de facilidad por el cual fue tasado. Cuando la comunicación tiene parte dentro y parte fuera de plan, se retornaron los códigos descritos más la cantidad de segundos que están fuera plan. Cuando la llamada está fuera de plan, se retornó el código de concepto que corresponde a la tarifa estándar de acuerdo al área tarifaria y tipo de destino de la comunicación.

Objetivo 6: Para ilustrar los resultados obtenidos con el tasador basado en reglas, se muestra en la Tabla 16 la tasación de un cliente con un plan implementado en el motor de reglas, mientras que en la Tabla 17 se muestra el resultado obtenido para el mismo cliente pero con el sistema actual.

	Fecha comunicación	Cantidad de segundos	Concepto cobro	Tipo concepto	Valor Neto	Valor IVA
1	03/01/2008 11:30:50	131	G6SG	TL	0	0
2	03/01/2008 11:30:50	131	G6RY	CACC	0	0
3	03/01/2008 12:17:11	139	G6SG	TL	0	0
4	03/01/2008 12:17:11	139	G6RY	CACC	0	0
5	03/01/2008 18:58:42	32	G6SG	TL	0	0
6	03/01/2008 18:58:42	32	G6RY	CACC	0	0

Tabla 16**:** Resultado de tasación de un cliente con motor de reglas

	Fecha comunicación	Cant seg	Cpto TL	Cpto Cacc	Neto Cacc	IVA Cacc	Neto TL	IVA TL
1	03/01/2008 11:30:50	131	G6SG	G6RY	0	0	0	0
2	03/01/2008 12:17:11	139	G6SG	G6RY	0	0	0	0
3	03/01/2008 18:58:42	32	G6SG	G6RY	0	0	0	0

Tabla 17: Resultado de tasación de un cliente sin motor de reglas

Objetivo 7: como resultado de la implementación del cluster con dos nodos, los tiempos en horas obtenidos para un total de 600 reglas se detallan en la siguiente tabla:

Registros	Con Motor	Sin Motor
1.000.000	7.4	6.2
700.000	5.25	4.90
400.000	4.3	2.9
200.000	1.5	1.23

Tabla 18**:** Tiempos por cantidad de registros con y sin motor de reglas

Como se aprecia los tiempos no son mayores a un 30% por sobre los tiempos sin el uso del motor de reglas, lo que demuestra que es una opción muy viable a la hora de separar la lógica de negocio de los sistemas.

4.9.3 Análisis

El plan de pruebas se realizó de manera incremental, de acuerdo al avance del desarrollo. Si bien en un principio esto hizo que el avance fuera un poco más lento, esta estrategia permitió depurar separadamente cada una de las reglas tipo, permitiendo advertir y solucionar los errores en un ambiente controlado, distinto al que se presenta al

finalizar el proyecto donde la cantidad de reglas implementadas hace más compleja la búsqueda de errores.

El paquete de división de reglas no presentó grandes problemas, cabe mencionar que tiene una cantidad reducida de reglas lo que hizo que las pruebas fueran cumplidas rápidamente. Durante el desarrollo hubo un cambio de versión de motor de reglas, y sólo en este momento surgieron problemas ya que en la nueva versión cambió la forma de evaluar expresiones numéricas, especialmente en lo que se refiere a asociatividad en operaciones de adición y multiplicación, lo cual fue solucionado agregando los paréntesis adecuados.

A la hora de probar las reglas tipo, surgieron algunas interrogantes que produjeron cambios en algunas reglas y de cierta forma en la estructura y cantidad de reglas calculadas en un principio, estos cambios permitieron reducir el número de reglas mediante la aplicación de prioridad y separación de algunas tareas específicas como la determinación del tipo de destino de una llamada.

En el ítem de pruebas globales, se muestra la tasación realizada mediante el motor de reglas en comparación a la tasación del sistema actual. Si bien en el sistema propuesto se guarda el valor del trafico local y del cargo de acceso en registros separados, los valores en dinero, tanto el neto como el valor con IVA, son iguales que los obtenidos en el sistema actual, esto está dado por el concepto de cobro el cual determina que las comunicaciones hechas por el cliente de ejemplo están dentro de plan y tienen un valor cero.

La prueba final se ilustra sólo para un cliente que tiene contratadas facilidades implementadas porque en el prototipo no se considera la aplicación de planes asociados a grupos de facturación y planes de empresas, por lo que los resultados globales del prototipo no son completamente comparables con los del sistema actual.

CAPITULO V: CONCLUSIONES

La implementación de este prototipo nos ha demostrado la viabilidad del uso de motores de reglas de negocios como solución para la implementación de sistemas donde se requiera separar la lógica del negocio de los códigos de las aplicaciones. Además, los proyectos de código abierto son una buena alternativa a la hora de reducir costos teniendo en este caso los mismos resultados que un producto comercial.

El impacto en el proceso de realización de cambios en las reglas de negocio es alto, sobre todo considerando el traspaso de responsabilidades que se logra al dejar a cargo al usuario de negocio de controlar la lógica de las aplicaciones. Si bien, precisamente esa es una de las motivaciones de esta tesis, la puesta en marcha de estos cambios debe hacerse cuidadosamente, considerando las capacitaciones adecuadas al nuevo personal responsable y asignando inequivocamente los roles que cumple cada uno de los actores implicados en el proceso.

La ventaja de poder intervenir las reglas de negocio en cualquier momento debe ser tomada con cautela, ya que los cambios en la lógica de los sistemas podría llevar a inconsistencias si es que no se hacen en el momento ni en la forma debida.

El uso de servicios Web como interfaz entre el motor de reglas y los demás sistemas, tiene beneficios clave como la capacidad de entregar servicios de decisión a aplicaciones desarrolladas con distintas tecnologías y mediante un protocolo abierto que otorga flexibilidad en el consumo de los servicios.

La aplicación desarrollada en esta tesis mostró a su vez una de las debilidades del uso de servicios Web, que es justamente cuando los servicios son utilizados para procesamiento intensivo de datos, ya que a diferencia de una aplicación no basada en servicios, por cada dato que se quiere procesar existe un tiempo adicional en la comunicación con el servidor, tiempo que a la larga repercute en una menor cantidad de datos procesados en el mismo tiempo. No obstante, existen soluciones a esta situación, en el caso de esta tesis se optó por paralelizar el proceso. Esta tarea se llevó a cabo mediante un cluster del servidor de aplicaciones, lo que llevo a disminuir los tiempos hasta los niveles esperados.

En esta tarea se pudo observar que al aumentar de uno a dos nodos, el poder de procesamiento aumentó en aproximadamente un 90%, reduciendo los tiempos prácticamente a la mitad. No se determinó un número óptimo de nodos ya que con dos se logró el objetivo, y este cálculo escapa al alcance de esta tesis.

La separación de la lógica de negocio del código permite tener una visión mucho más clara del funcionamiento general de la aplicación. En el caso del tasador de llamadas locales desarrollado en esta tesis, el código pl/sql desarrollado se encarga de reunir correctamente los datos necesarios para ser enviados a los servicios Web quienes simplemente utilizan el motor para obtener el resultado que es devuelto a la aplicación pl/sql que se encarga de registrar esto en la base de datos. Esta separación en capas, extracción de datos, interfaz de comunicación estándar y motor de reglas, hace que los costos de mantención se reduzcan otorgando un aislamiento entre las distintas partes de la aplicación.

Esta división en capas hizo que el trabajo fuera divisible entre los dos alumnos tesistas, en donde uno se encargó de realizar la capa de datos y comunicación y otro del desarrollo de la lógica de negocio, debiendo interactuar entre si a la hora de definir los datos de entrada y salida al motor para obtener los resultados esperados.

El prototipo de tasación de llamadas locales obtiene resultados correctos para el conjunto de datos seleccionados, cabe mencionar que el prototipo incluye sólo los planes asociados a un servicio, por lo que los planes asociados a grupos de facturación y empresas no fueron incluidos.

El producto utilizado provee una interfaz amigable que permite organizar las reglas para encontrarlas de manera rápida, haciendo más eficiente la realización de cambios en comparación al sistema actual. Además, la herramienta cuenta con un sistema de control de cambios, mediante el cual se puede volver atrás cualquier modificación que haya repercutido negativamente en el funcionamiento de la aplicación.

La utilización de un motor de reglas es transversal a todas las áreas de la compañía, lo cual deja abierta la posibilidad de implementar futuros proyectos basados en este prototipo.

CAPITULO VI: REFERENCIAS

[Tri.] Trilles Chief Juanjo, Reglas de Negocio (BR) y Gestión por Procesos de Negocio (BPM) disponible en: http://www.club-bpm.com/Noticias/art0007.htm

[BRG] Bussiness rules group (2007), Semantics of Business Vocabulary and Business Rules (SBVR) disponible en: http://www.businessrulesgroup.org/sbvr.shtml

[OMG] The Object Management Group (1997-2008), disponible en: http://www.omg.org

[RIC01], Artificial Intelligence, Claine Rich, Kevin knight.

[CHA88], Chatain J.N. & Dussauchoy A. (1988). Sistemas Expertos, métodos y herramientas.

[GIA01]Giarratano J. & Riley C. (2001).Sistemas expertos, principios y programación.

[CIE] CIENTEC S.A. (n.d.) BRMS: REGLAS MÁS SINTONIZADAS CON EL NEGOCIO disponible en: http://www.cientec.com/Management/Management19.asp

[DDC]Departamento de Computación - Facultad de Ciencias Exactas y Naturales - Universidad de Buenos Aires, Sincronización de sistemas distribuidos disponible en: http://www-2.dc.uba.ar/materias/so/datos/cap23.pdf

[UAL] Universidad de Alicante. Web Services, disponible en: http://cv1.cpd.ua.es/ws/default.asp?wOpcion=7

[SM] Sun Microsystems (1995-2007), An Overview of RMI Applications disponible en: http://java.sun.com/docs/books/tutorial/rmi/overview.html

[PEC] Programación en castellano (1998-2007), RMI mano a mano con SSL: construyendo aplicaciones distribuidas seguras disponible en: http://www.programacion.com/articulo/joa_rmissl/

[OAS] oasis (1993-2008), Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)

[DSI]Departamento de Sistemas Informáticos y computación, REST v7s Web services disponible en: http://www.dsic.upv.es/~rnavarro/NewWeb/docs/RestVsWebServices.pdf

[OBR] Oracle Business Rules, dispobible en: http://www.oracle.com/technology/products/ias/business_rules/index.html

[OPE]Open Rules, Business Rules Management System, disponible en: http://openrules.com/

[AGI]AgilePartner S.A. (2004), NxBre .NET Business Rule Engine disponible en: http://www.agilepartner.net/oss/nxbre/

[JBO] Jboss, Drools disponible en http://labs.jboss.com/drools/

[SCR] Metodologia Scrum disponible en http://scrummethodology.com/

[FAI] Fair Isaac, Blaze Advisor disponible en http://www.fairisaac.com/fic/en/product-service/product-index/blaze-advisor/

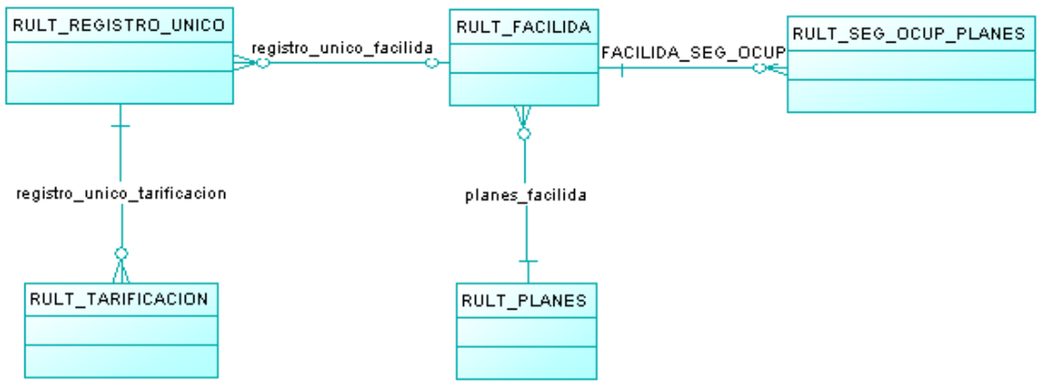
[MIC] Microsoft, Microsoft BizTalk Server, disponible en http://www.microsoft.com/spain/biztalk/default.mspx

[GLA] GlassFish, disponible en https://glassfish.dev.java.net/

[APA] Apache Software Foundation, The Apache Tomcat Connector disponible en http://tomcat.apache.org/connectors-doc/

CAPITULO VII: ANEXO

7.1 Modelo de datos



7.1.1 Descripción

RULT_REGISTRO_UNICO: Esta tabla contiene todas las comunicaciones, y es aquí donde se debe registrar si la comunicación es tasada o no. Los campos utilizados en el sistema son:

Nombre campo	Tipo de dato	Descripción
llam_num_reg_ewsd	**varchar2 (10)**	Identificador de comunicación dado por la central
central	**varchar2 (4)**	Central
fech_inicio	**date**	Mes al que pertenece la comunicación
proc_id	**number(8)**	Id. de proceso
llam_sentido	**varchar2 (1)**	Sentido de la llamada (entrante, saliente)
ani_perfil_id	**number (7)**	Id. de perfil de fono origen
dni_perfil_id	**number (7)**	Id. de perfil de fono destino (solo fono pertenece a telsur)
ani_significativo	**varchar2** (4)	Zona del número que realizó la comunicación.
fono_destino	**varchar2** (20)	número telefónico de destino de la comunicación
ani_empresa	**varchar2 (5)**	Numero de empresa de origen
dni_empresa	**varchar2 (5)**	Numero de empresa de destino
cant_seg	**number** (7)	cantidad de segundos de duración de la comunicación
fech_comuni	**date**	fecha de inicio de la comunicación, la que contiene año, mes, día, hora, minutos y segundos
tipo_trafico	**varchar2 (30)**	Tipo de destino de la llamada
ani_area_tarifaria	**number (2)**	Área tarifaria del fono de origen

RULT_FACILIDA: Contiene las facilidades que posee cada servicio de la compañía. Los campos utilizados para el sistema son:

Nombre campo	Tipo de dato	Descripción
servicio_id	**number**	Id. del servicio que posee la facilidad
zona_servicio	**number** (2)	Zona del fono de la facilidad
fono_servicio	**char**(10)	Fono de la facilidad
codi_facilida	**varchar2**(4)	Código de la facilidad del fono
zona_facilida	**number** (2)	Si la facilidad es número frecuente, zona_facilida almacena la zona del fono frecuente al que representa la facilidad.
fono_facilida	**varchar2(50)**	Si la facilidad es numero frecuente, fono_facilida almacena el numero frecuente al que hace referencia la facilidad
proporcion_plan	**number**	Es la proporción de plan que tiene disponible el servicio en el mes actual

RULT_PLANES: Contiene la información relativa a las facilidades disponibles y que son asociadas a los números telefónicos. Los campos utilizados son:

Nombre campo	Tipo de dato	Descripción
codi_facilida	**varchar2** (4)	código de la facilidad
nomb_facilida	**varchar2**(100)	nombre de la facilidad
fech_ini	**date**	fecha de inicio de vigencia de la facilidad
fech_fin	**date**	fecha de fin de vigencia de la facilidad

RULT_TARIFICACION: Esta tabla contiene las llamadas que son tasadas,Sus campos son:

Nombre campo	Tipo Dato	Descripción
LLAM_NUM_REG_EWSD	**varchar2**(10)	Identificador único del registro entregado por la central
CENTRAL	**varchar2**(4)	Nombre de la central a la que pertenece el registro
REGISTRO_MES	**date**	Mes al que pertenece la comunicación
PROC_ID	**number** (8)	Id del proceso
LLAM_TIPO_HORARIO	**varchar2**(1)	Horario en el que se realizó la llamada, (N=Normal, R=Reducido, V=Vespertino)
LLAM_SENTIDO	**varchar2**(1)	Sentido de la llamada, (E=Entrante, S=Saliente)
LLAM_TIPO_DIA	**varchar2**(1)	Tipo de día (H= Habil, F= Festivo)
LLAM_DIA_SEMANA	**number**(1)	Día de la semana
LLAM_FECHA_COMUNI	**date**	Fecha y hora de inicio en la comunicación en este horario
LLAM_CANT_SEG	**number**(7)	Tiempo en segundos de la llamada en el horario.
EMP_PAGO	**varchar2**(5)	Empresa a la cual se le debe cancelar

EMP_CONCEPTO_PAGO	**varchar2**(4)	Concepto que se debe cancelar
EMP_VALOR_PAGO	**number**(9)	Valor que debe ser pagado
EMP_COBRO	**varchar2**(5)	Empresa a la cual se le debe cobrar
EMP_CONCEPTO_COBRO	**varchar2**(4)	Concepto que debe ser cobrado
EMP_VALOR_COBRO	**number**(9)	Valor que debe ser cobrado
CLI_COBRO	**varchar2**(20)	Cliente al que se le debe cobrar (ID_PERFIL_CLIENTE)
CLI_CONCEPTO_COBRO	**varchar2**(4)	Concepto que debe ser cobrado
CLI_VALOR_COBRO_NETO	**number**	Valor neto que debe ser cobrado
CLI_VALOR_COBRO_IVA	**number**	Valor neto que debe ser cobrado
CLI_CODI_PLAN	**varchar2(30)**	Codigo de facilidad de plan
TIPO_CONCEPTO	**number**	Tipo de concepto que se taso

RULT_SEG_OCUP_PLANES: Tabla que sirve para almacenar la cantidad de segundos que un servicio ha ocupado durante el mes, según esto se analiza si las comunicaciones están o no dentro de plan.

Nombre campo	Tipo de dato	Descripción
servicio_id	**number**	Id. de servicio
codi_facilida	**varchar2(4)**	código de la facilidad
registro_mes	**date**	Mes en que se ocuparon los segundos registrados
Segs_slm_hv	**number(7)**	Cantidad de segundos a fijo ocupados en el mes actual en horario vespertino
Segs_slm_hn	**number(7)**	Cantidad de segundos a fijo ocupados en el mes actual en horario normal
Segs_slm_he	**number(7)**	Cantidad de segundos a fijo ocupados en el mes actual en horario económico
Segs_mov_hv	**number(7)**	Cantidad de segundos a móvil ocupados en el mes actual en horario vespertino
Segs_mov_hn	**number(7)**	Cantidad de segundos a móvil ocupados en el mes actual en horario normal
Segs_mov_he	**number(7)**	Cantidad de segundos a móvil ocupados en el mes actual en horario economico
Segs_slm_onnet	**number(7)**	Cantidad de segundos ocupados en planes on-net
Segs_slm_offnet	**number(7)**	Cantidad de segundos ocupados en planes off-net

TSIS_RULT_LOG: Tabla que almacena los errores producidos durante el proceso de tasacion.

Nombre campo	Tipo de dato	Descripción
llam_num_reg_ewsd	**varchar2(10)**	Id. de la comunicación
desc_result	**varchar2(700)**	Descripción del error ocurrido
Proc_id	**number**	Id. del proceso de tasación
fecha_comuni	**date**	Fecha de la comunicación que produjo error

Fecha_ejec	**date**	Fecha en que se ejecuto la tasación que produjo error

LOCT_CPTOSLOC: Tabla que contiene los codigos de concepto y sus valores del tipo SLM, SPH, TL y TPH.

Nombre campo	Tipo de dato	Descripción
tipo_concepto	**varchar2(3)**	Tipo de concepto
tipo_horario	**char(1)**	Horario de vigencia del concepto
area_tarifaria	**varchar2(4)**	Area tarifaria de validez del concepto
codi_plan	**varchar2(4)**	Codigo de plan
codi_concepto	**number(6)**	Codigo del concepto
codi_interno	**varchar2(4)**	Denominación interna para el concepto
codi_agrupaci	**integer**	Codigo de agrupación
codi_area	**integer**	Codigo de área
desc_concepto	**varchar2(80)**	Descripción del concepto
flag_afectiva	**varchar2(4)**	
Fech_inivigen	**date**	Fecha inicio de vigencia del concepto
Fech_finvigen	**date**	Fecha de fin de vigencia del concepto
vlor_concepto	**number**	Valor del concepto

7.1.2 Paquetes y procedimientos almacenados

Convenciones

Se establecen los siguientes prefijos para la identificación de variables y constantes:

- p_: corresponde a parámetros de funciones y procedimientos almacenados.
- v_: corresponde a variables internas de una función o procedimiento.
- c_: corresponde a constantes utilizadas en funciones y procedimientos.
- vg_: corresponde a variables globales de un package.

Package: **tsis_utl_loc**

Descripción: Paquete que contiene el procedimiento principal así como otras utilidades que no corresponden a una tabla especifica.

Variables y constantes globales:

```
-- Contiene la dirección del servicio web
 vg_url VARCHAR2 (255):= 'http://172.16.78.121:8080/WSTasadorLlamadas/
services/TasaLlamadaWS?wsdl';

-- El schema que define la petición al servicio web
  vg_schema VARCHAR2 (50):= 'xmlns="http://tasador/xsd"';

-- El schema necesario para acceder mediante XPath a la respuesta del
servicio web
  vg_schema_xpath VARCHAR2 (50):= 'xmlns:ns="http://tasador/xsd"';
```

```
-- Empresa de la cual se obtendran llamadas de registro unico
  vg_empresa VARCHAR2 (50):= 'TELSUR';

-- Flag que se utiliza para referenciar a una llamada dentro de plan
  c_dentro char (1):= 'D';

-- Factor que transforma en segundos una cantidad dada en dias
  c_trans_seg number:= 1/ (24 *3600);
```

Procedimientos:

tasa_llamada		
Descripción	Procedimiento principal, obtiene un cursor con las llamadas locales desde registro único y solicita el valor de su tasación para cada una de ellas al servicio Web que despliega el motor de reglas	
Nombre Parámetro	**Tipo de dato**	**Descripción**
p_fech_ini	**in date**	Fecha de inicio a tasar, un valor null tasa todas las llamadas que no estén tasadas sin límite hacia atrás.
p_fech_fin	**in date**	Fecha de fin tasación, un valor null tasa todas las llamadas desde p_fech_ini hasta la fecha actual.
p_registro_mes	**in date**	Mes en el cual se quiere tasar o retasar, sirve como filtro complementario a fech_ini, fech_fin
p_cgo_perfil_id	**in number**	Perfil de cliente que se quiere tasar
p_llam_tipo_destino	**in varchar2**	Tipo de destino que se quiere tasar que puede ser “Local Telsur”, “Móvil” u otro.
p_central	**in varchar2**	Codigo de la central que se desar tasar
p_zona_origen	**in number**	Zona que se desea tasar
p_llam_num_reg_ewsd	**in varchar2**	Comunicación especifica que se desea tasar
p_retasacion	**in number**	Indica si el procedimiento se ejecuta para retasar

tasa_llamada_ws		
Descripción	Procedimiento que se encarga exclusivamente de llamar al servicio Web tasaLlamada. Posee variables de salida las cuales reciben la respuesta de la tasación del motor de reglas.	
Nombre Parámetro	**Tipo de dato**	**Descripción**
p_ani_numero	**in varchar2**	Numero origen de la comunicación.
p_dni_numero	**in varchar2**	Numero destino de la comunicación.
p_ani_perfil_id	**in number**	Id. de perfil de usuario de número de origen
p_dni_perfil_id	**in number**	Id. de perfil de usuario de número de destino
p_llam_cant_seg	**in number**	Cantidad de segundos de la comunicación.
p_llam_horario	**in varchar2**	Horario en que se realiza la comunicación.
p_llam_tipo_destino	**in varchar2**	Indica si la llamada fue realizada hacia un móvil un local Telsur u otro local.
p_llam_ani_area_tarifaria	**in number**	Área tarifaria del fono de origen.

p_registro_mes	**in date**	Fecha del primer día del mes que se esta tasando
p_ani_empresa_numero	**in varchar2**	Numero (código) de la empresa a la que pertenece el fono origen
p_dni_empresa_numero	**in varchar2**	Numero (código) de la empresa a la que pertenece el fono destino
p_cat_servicio	**in varchar2**	Categoría del servicio
p_tasado	**out varchar2**	Indica si la comunicación fue tasada o no.
p_cant_seg_dentplan	**out number**	Cantidad de segundos que fueron tasados dentro de plan
p_cant_seg_fueraplan	**out number**	Cantidad de segundos que fueron tasados fuera de plan
p_cli_concepto_cobro	**out varchar2**	Concepto por el que se tasó dentro de plan.
p_cli_concepto_cobro_cacc	**out varchar2**	Concepto de cargo de acceso cuando la llamada es a móvil.
p_codigo_facilidad	**out number**	Código de la facilidad que fue utilizada.
p_fono_tipo	**out number**	Tipo de fono, ya sea local o móvil (0 o 1).
p_onoff	**out number**	Indica si la llamada entro a un plan on-net u off-net

divide_llamada_ws		
Descripción	Procedimiento que invoca al servicio web que expone el paquete de reglas de division de llamadas de acuerdo a los tramos horarios vigentes. En las variables de salida quedan los valores correspondientes a la tasación actual, y en llam_cant_seg y llam_fecha_comuni el tramo horario siguiente	
Nombre Parámetro	**Tipo de dato**	**Descripción**
p_llam_cant_seg	**in number**	Cantidad de segundos de la comunicación.
p_llam_fecha_comuni	**in date**	Fecha de la comunicación.
p_llam_tipo_dia	**in varchar2**	Tipo de día (H= Hábil, F= Festivo)
p_llam_dia_semana	**in number**	Día de la semana (1= Lunes, 7= Domingo)
p_llam_fecha_comuni_a_tasar	**out date**	Fecha de inicio de la comunicación que se tasará
p_llam_cant_seg_a_tasar	**out number**	Cantidad de segundos de la comunicación que se tasara, la cual esta solo en un tramo horario
p_llam_horario_a_tasar	**out varchar2**	Horario en que esta la comunicación a tasar (V= Vespertino, N= Normal, E= Económico)

obtieneServicioId	
Descripción	Función que obtiene el servicio_id de la tabla rult_perfil_cliente a partir de su perfil_id

Return	Number	
Nombre Parámetro	**Tipo de dato**	**Descripción**
p_id_perfil	**in number**	Numero de perfil de cliente del cual se obtendrá el servicio_id.

obtieneTipoDia		
Descripción	Función que obtiene el tipo de día que se esta tasando, este puede ser H = Hábil o F= Festivo	
Return	Varchar2(1)	
Nombre Parámetro	**Tipo de dato**	**Descripción**
p_llam_fecha_comuni	**in date**	Fecha de comunicación de la cual se desea saber si es día hábil o festivo

obtieneValorConcepto		
Descripción	Función que obtiene el valor de un concepto ya sea SLM para las llamadas locales o TL para las llamadas a móviles. Este valor es extraído de la tabla loct_cptosloc	
Return	Number	
Nombre Parámetro	**Tipo de dato**	**Descripción**
p_codi_interno	**in varchar2**	Código de concepto del cual se quiere saber el valor
p_llam_fecha_comuni	**in date**	Fecha donde debe ser valido el concepto
P_tipo_fono	**in number**	Tipo Fono (Local, Movil)
p_cat_servicio	**in number**	Categoría de servicio (PHS = 38)
p_valor_concepto	**out number**	Valor obtenido
p_tipo_concepto	**out varchar2**	Tipo de concepto (SLM,TL,SPH,TPH)

obtieneValorConceptoCacc		
Descripción	Función que obtiene el valor de un concepto de cargo de acceso cuando las llamadas son a teléfonos móviles. Este valor es extraído de la tabla loct_cptoscpp.	
Return	Number	
Nombre Parámetro	**Tipo de dato**	**Descripción**
p_codi_interno	**in varchar2**	Código de concepto del cual se quiere saber el valor
p_llam_fecha_comuni	**in date**	Fecha donde debe ser valido el concepto

insertarTasacion		
Descripción	Procedimiento que obtiene el valor de un concepto de cargo de acceso cuando las llamadas son a teléfonos móviles. Este valor es extraído de la tabla loct_cptoscpp.	
Nombre Parámetro	**Tipo de dato**	**Descripción**
p_llam_num_reg_ewsd	**in varchar2**	Identificador de comunicación proveniente de la central
p_central	**in varchar2**	Código de la central
p_registro_mes	**in date**	Mes en que se produce la comunicación
p_proc_id	**in number**	Id de proceso

p_llam_tipo_horario	**in varchar2**	Tipo Horario de la comunicación
p_llam_sentido	**in varchar2**	Sentido de la comunicación(entrante, saliente)
p_llam_tipo_dia	**in varchar2**	Tipo de día (H= Hábil, F= Festivo)
p_llam_dia_semana	**in number**	Día de la semana (1= Lunes, 7= Domingo)
p_llam_fecha_comuni	**in date**	Fecha de la comunicación.
p_tasado	**in varchar2**	Indica si la llamada fue tasada o no
p_servicio_id	**in number**	Id. Servicio que realizo la llamada
p_llam_cant_seg_dentplan	**in number**	Cantidad de segundos dentro de plan.
p_llam_cant_seg_fueraplan	**in number**	Cantidad de segundos fuera de plan.
p_cli_concepto_cobro	**in varchar2**	Concepto de cobro (slm o tl)
p_cli_concepto_cobro_cacc	**in varchar**2	Concepto de cobro (cargo de acceso)
p_codigo_facilidad	**in number**	Código de Facilidad por la que se tasó
p_fono_tipo	**in number**	Tipo de fono, ya sea local o móvil (0 o 1).
p_onoff	**in number**	Indica si la llamada entro a un plan on-net u off-net
P_proc_id_tasacion	**in number**	Proc_id de la tasacion
P_cat_servicio	**in number**	Categoia de servicio

eliminarTasacion		
Descripción	Procedimiento que elimina la tasación dada por los parámetros de entrada. Este procedimiento actualiza el flag reproceso de la tabla rult_registro_unico, resetea los contadores de segundos ocupados y elimina las entradas correspondientes de la tabla rult_tarificacion	
Nombre Parámetro	**Tipo de dato**	**Descripción**
p_fecha_ini	**in varchar2**	Fecha de inicio a tasar, un valor null elimina todas las llamadas que no estén tasadas sin límite hacia atrás.
p_fecha_fin	**in varchar2**	Fecha de fin tasación, un valor null elimina todas las llamadas desde p_fech_ini hasta la fecha actual.
p_registro_mes	**in date**	Mes en el cual se quiere tasar o retasar, sirve como filtro complementario a fech_ini, fech_fin
p_llam_tipo_destino	**in varchar2**	Tipo de destino que se quiere eliminar que puede ser “Local Telsur”, “Móvil” u otro.
p_cgo_perfil_id	**in number**	Perfil del cliente que se quiere eliminar

Package: **tsis_rulp_seg_ocup_planes**

Descripción: paquete que contiene todas las operaciones de acceso a la tabla tsis_rult_seg_ocup_planes.

Variables y constantes Globales: No tiene.

Procedimientos:

updateResetCont		
Descripción	Procedimiento que reinicia los contadores de segundos ocupados en un mes especifico para todas las facilidades que posea el servicio. Este procedimiento se utiliza cuando se retasa.	
Nombre Parámetro	**Tipo de dato**	**Descripción**
p_servicio_id	**in number**	Id. servicio al que se le reiniciaran los contadores de llamadas
p_registro_mes	**in date**	Mes en que se le reiniciaran los contadores al servicio especificado

insertSegundosOcup		
Descripción	Procedimiento que actualiza la cantidad de segundos que lleva ocupados un servicio en un mes específico y una facilidad especifica. Si no existe registro en ese mes para esa facilidad de ese servicio, este se inserta.	
Nombre Parámetro	**Tipo de dato**	**Descripción**
p_campo_actualizar	**in varchar2**	Id. servicio al que se le reiniciaran los contadores de llamadas.
p_servicio_id	**in number**	Id. Servicio que se desea insertar o actualizar.
p_facilidad	**in varchar2**	Facilidad que se desea actualizar o insertar.
p_registro_mes	**in date**	Mes para el que se desea actualizar o insertar.
p_cant_seg	**in number**	Cantidad de segundos que se insertaran o actualizaran.

Package: **tsis_rulp_facilida**

Descripción: Paquete que contiene las operaciones sobre la tabla tsis_rult_facilida

Variables y constantes Globales: No tiene.

Procedimientos:

obtieneFacilidad		
Descripción	Función que retorna un cursor con: las facilidades de un número telefónico y la cantidad de segundos consumidos de cada facilidad durante el mes actual.	
Nombre Parámetro	**Tipo de dato**	**Descripción**
p_perfil_id	**in number**	Perfil del que se obtendrán las facilidades
p_fecha_mes	**in varchar2**	Mes del que se necesitan las facilidades

Package: **tsis_rulp_registro_unico**

Descripción: Paquete que contiene las operaciones sobre la tabla tsis_rult_registro_unico.

Variables y constantes Globales: No tiene.

Procedimientos:

updateReproceso		
Descripción	Procedimiento que setea en “S” el flag de reproceso de registro único	
Nombre Parámetro	**Tipo de dato**	**Descripción**
p_fecha_ini	**in varchar2**	Fecha de inicio del rango de llamadas que se desea reprocesar.
p_fecha_fin	**in varchar2**	Fecha de fin del rango de llamadas que se desea reprocesar.
p_registro_mes	**in date**	Mes en el cual se quiere reprocesar
p_llam_tipo_destino	**in varchar2**	Tipo de destino que se quiere reprocesar.
p_cgo_perfil_id	**in number**	Perfil del cliente que se quiere reprocesar

updateTasado		
Descripción	Procedimiento que setea el flag tasado a la comunicación dada	
Nombre Parámetro	**Tipo de dato**	**Descripción**
p_llam_num_reg_ewsd	**in number**	Identificador de comunicación que se desea setear.
p_tasado	**in varchar2**	Valor con el que se seteara el flag tasado.

Package: **tsis_rulp_tarificacion**

Descripción: Paquete que contiene las operaciones sobre la tabla tsis_rult_tarificacion.

Variables y constantes Globales: No tiene.

Procedimientos:

insertTarificacion		
Descripción	Procedimiento que realiza insert en la tabla rult_tarificacion.	
Nombre Parámetro	**Tipo de dato**	**Descripción**
p_llam_num_reg_ewsd	**In varchar2**	Identificador de comunicación proveniente de la central
p_central	**in varchar2**	Código de la central
p_registro_mes	**in date**	Mes en que se produce la comunicación
p_proc_id	**in number**	Id. de proceso
p_llam_tipo_horario	**in varchar2**	Tipo Horario de la comunicación
p_llam_sentido	**in varchar2**	Sentido de la comunicación(entrante, saliente)
p_llam_tipo_dia	**in varchar2**	Tipo de día (H= Hábil, F= Festivo)
p_llam_dia_semana	**in number**	Día de la semana (1= Lunes, 7= Domingo)
p_llam_fecha_comuni	**in date**	Fecha de la comunicación.
p_llam_cant_seg	**in number**	Cantidad de segundos de la comunicación.

p_cli_concepto_cobro	**in varchar2**	Concepto de cobro.
p_vlor_concepto	**in number**	Valor del concepto a cobrar.
p_codigo_facilidad	**in number**	Código de Facilidad por la que se tasó.
p_tipo_concepto	**in varchar2**	Tipo de concepto

deleteTarificacion		
Descripción	Procedimiento que realiza delete de registros de la tabla rult_tarificacion de acuerdo al rango dado en los parámetros.	
Nombre Parámetro	**Tipo de dato**	**Descripción**
p_fecha_ini	**in varchar2**	Fecha de inicio del rango de llamadas que se desea eliminar.
p_fecha_fin	**in varchar2**	Fecha de fin del rango de llamadas que se desea eliminar.
p_registro_mes	**in date**	Mes en el cual se quiere eliminar
p_llam_tipo_destino	**in varchar2**	Tipo de destino que se quiere eliminar.
p_cgo_perfil_id	**in number**	Perfil del cliente que se quiere eliminar.

7.2 Descripción de Reglas tipos

A continuación se describen con mayor detalle las reglas tipos implementadas, detallando los objetos de entrada y salida.

7.2.1 Reglas Auxiliares

Reglas que se ejecutan al principio del paquete de reglas Tasación Llamadas, con el objetivo de identificar las llamadas para “setear” con el valor apropiado los objetos a utilizar en las reglas de planes.

Nombre Regla	Llamada a “Red”
Objetos Entrada	RegistroLlamadas.fono_tipo_destino
	RegistroLlamadas.dni_empresa_numero
	RegistroLlamadas.flag_pase
Objetos Salida	RegistroLlamadas.tipo_fono
	RegistroLlamadas.flag_pase
Prioridad	220

Descripción

Regla para identificar si la llamada es a red fija, red rural y línea 600
Nombre Regla: Llamada a “Red”(Red Fija, Red Rural o Línea 600)

El detalle de los objetos retornados es:

- Tipo_fono: Valor en “0” que indica que la llamada es a red fija, valor en “2” si la llamada es a línea 700, valor en “3” si la llamada es a linea 600, valor en “4” si la llamada es a red rural.
- Flag_pase: Valor en “0”, da el pase a las siguientes reglas

Nombre Regla	CACC Movil “Empresa_Movil” “Horario”
Objetos Entrada	RegistroLlamadas.fono_tipo_destino
	RegistroLlamadas.dni_empresa_numero
	RegistroLlamadas.llam_horario
	RegistroLlamadas.flag_pase
Objetos Salida	RegistroLlamadas.cli_concepto_cobro_cacc
	RegistroLlamadas.tipo_fono
	RegistroLlamadas.flag_pase
Prioridad	220

Descripción

Regla para asignar el valor del concepto por cargo de acceso a las llamadas a celular.

Nombre Regla: CACC Movil “Empresa_Movil”(empresa de celular) “Horario”(HN: horario normal, HE: horario economico, HV: horario vespertino)

El detalle de los objetos retornados es:

- cli_concepto_cobro_cacc : Valor del concepto de cobro por cargo de acceso de la empresa movil
- Tipo_fono: Valor en “1” que indica que la llamada es a móviles
- Flag_pase: Valor en 0, da el pase a las siguientes reglas

Nombre Regla	CACC Rural “Empresa_Telefonia_Rural” “Horario”
Objetos Entrada	
	RegistroLlamadas.dni_empresa_numero
	RegistroLlamadas.llam_horario
	RegistroLlamadas.flag_pase
Objetos Salida	RegistroLlamadas.cli_concepto_cobro_cacc
	RegistroLlamadas.tipo_fono
	RegistroLlamadas.flag_pase
Prioridad	220
Descripción	
Regla para asignar el valor del concepto por cargo de acceso a las llamadas a red rural. Nombre Regla: CACC Rural “Empresa_Rural”(empresa de telefoniaRural) “Horario”(HN: horario normal, HE: horario economico, HV: horario vespertino) El detalle de los objetos retornados es: - cli_concepto_cobro_cacc : Valor del concepto de cobro por cargo de acceso de la empresa rural - Tipo_fono: Valor en “4” que indica que la llamada es a red rural - Flag_pase: Valor en 0, da el pase a las siguientes reglas	

No tasables

Contempla aquellas reglas para llamadas que no son tasables.

Nombre reglas	No_Tasable_CantSegundos,
Objetos Entrada	RegistroLlamadas.llam_cant_seg
	RegistroLlamadas.flag_pase
Objetos Salida	RegistroLlamadas.tasado
	RegistroLlamadas.llam_cant_seg_dentplan
	RegistroLlamadas.llam_cant_seg_fueraplan
	RegistroLlamadas.cli_concepto_cobro
Prioridad	100
Descripción	
Regla para las llamadas menores o iguales a tres segundos, las cuales no se tasan, siendo su valor de tasación igual a N.	

Nombre Regla	NoTasable_Centrex
Objetos Entrada	RegistroLlamadas.ani_numero
	RegistroLlamadas.dni_numero
	RegistroLlamadas.tasado
	FacilidadesAni.numero
	FacilidadesAni.codi_facilida
	FacilidadesAni.fono_facilida
	FacilidadesDni.numero
	FacilidadesDni.codi_facilida
	FacilidadesDni.fono_facilida
Objetos Salida	RegistroLlamadas.llam_cant_seg_dentplan
	RegistroLlamadas.llam_cant_seg_fueraplan
	RegistroLlamadas.cli_concepto_cobro
	RegistroLlamadas.codigo_facilidad
	RegistroLlamadas.flag_pase
Prioridad	

Descripción

Si la llamada es entre números pertenecientes a mismo centrex, entonces el valor cobro es cero.

El detalle de los objetos retornados es:

- llam_cant_seg_dentplan: la cantidad de segundos de la llamada
- llam_cant_seg_fueraplan: Valor cero, ya que la llamada esta 100% dentro de plan
- cli_concepto_cobro: valor concepto a cobrar, en este caso es aquel de valor cero “EE39”
- codigo_facilidad: El código de la facilidad en el que cayo la llamada..

RN	NoTasable_Prepago
Objetos Entrada	RegistroLlamadas.ani_numero
	RegistroLlamadas.tasado
	FacilidadesAni.numero
	FacilidadesAni.codi_facilida
Objetos Salida	
	RegistroLlamadas.llam_cant_seg_dentplan
	RegistroLlamadas.llam_cant_seg_fueraplan
	RegistroLlamadas.cli_concepto_cobro
	RegistroLlamadas.codigo_facilidad
	RegistroLlamadas.flag_pase

Descripción

Regla para las llamadas efectuadas por un fono de prepago, cuyo valor de cobro es cero.

El detalle de los objetos retornados es:

- llam_cant_seg_dentplan: la cantidad de segundos de la llamada
- llam_cant_seg_fueraplan: Valor cero, ya que la llamada es prepago.
- cli_concepto_cobro: valor concepto a cobrar, en este caso es aquel de valor cero “EE39”
- codigo_facilidad: El codigo de la facilidad en el que cayo la llamada..

Nombre Regla	NoTasable_NumeroFrecuente
Objetos Entrada	RegistroLlamadas.ani_numero
	RegistroLlamadas.dni_numero
	RegistroLlamadas.tasado
	FacilidadesAni.numero
	FacilidadesAni.codi_facilida
	FacilidadesAni.fono_facilida
Objetos Salida	RegistroLlamadas.llam_cant_seg_dentplan
	RegistroLlamadas.llam_cant_seg_fueraplan
	RegistroLlamadas.cli_concepto_cobro
	RegistroLlamadas.codigo_facilidad
	RegistroLlamadas.flag_pase

Descripción

Regla para las llamadas a numeros fruecuentes, siendo el valor de cobros cero

El detalle de los objetos retornados es:

- llam_cant_seg_dentplan: la cantidad de segundos de la llamada
- llam_cant_seg_fueraplan: Valor cero, ya que la llamada es a numero fruecuente.
- cli_concepto_cobro: valor concepto a cobrar, en este caso es aquel de valor cero “EE39”
- codigo_facilidad: El codigo de la facilidad en el que cayo la llamada..

Planes SLM

Nombre Regla	Dentro Plan “**Nombre de Plan**” “**Horario**”
Objetos Entrada	
	RegistroLlamadas.`llam_cant_seg`
	RegistroLlamadas.`tipo_fono`
	RegistroLlamadas.`llam_horario`
	RegistroLlamadas.`flag_pase`
	FacilidadesAni.codigo_facilidad
	FacilidadesAni.segs_slm_hn
	FacilidadesAni.segs_slm_he
	FacilidadesAni.segs_slm_hv
	FacilidadesAni.proporcion
Objetos Salida	RegistroLlamadas.llam_cant_seg_dentplan
	RegistroLlamadas.llam_cant_seg_fueraplan
	RegistroLlamadas.cli_concepto_cobro
	RegistroLlamadas.codigo_facilidad
	RegistroLlamadas.flag_pase
Prioridad	95

Descripción

Regla para las llamadas de clientes que están dentro de plan, las cuales no se cobran.

Nombre Regla: Dentro Plan “Nombre del Plan” “Horario”(HN: horario normal, HE: horario económico, HV: horario vespertino).

El detalle de los objetos retornados es:

- llam_cant_seg_dentplan: Cantidad de segundos de la llamada
- llam_cant_seg_fueraplan: Valor en cero, ya que la llamada esta 100% dentro de plan
- cli_concepto_cobro: valor concepto a cobrar, en este caso es aquel de valor cero “EE39”
- codigo_facilidad: Código de la facilidad en el que cayó la llamada.

Nombre Regla	Divide Llamada “**Nombre de Plan**” “**Horario**”
Objetos Entrada	RegistroLlamadas.`llam_cant_seg`
	RegistroLlamadas.`tipo_fono`
	RegistroLlamadas.`llam_horario`
	RegistroLlamadas.`flag_pase`
	FacilidadesAni.codigo_facilidad
	FacilidadesAni.segs_slm_hn
	FacilidadesAni.segs_slm_he
	FacilidadesAni.segs_slm_hv
	FacilidadesAni.proporcion
Objetos Salida	RegistroLlamadas.llam_cant_seg_dentplan
	RegistroLlamadas.llam_cant_seg_fueraplan
	RegistroLlamadas.cli_concepto_cobro
	RegistroLlamadas.codigo_facilidad
	RegistroLlamadas.flag_pase
Prioridad	90

Descripción

Regla para las llamadas que sobrepasan la cantidad de segundos de la facilidad, teniendo segundos dentro de plan y fuera de plan.

Nombre Regla: Divide Llamada “Nombre de Plan” “Horario”(HN: horario normal, HE: horario económico, HV: horario vespertino)

El detalle de los objetos que se retornan son:

- llam_cant_seg_dentplan: Parte de los segundos de la llamada dentro de plan.
- llam_cant_seg_fueraplan: Parte de los segundos de la llamada fuera de plan.
- cli_concepto_cobro: valor del concepto a cobrar, en este caso es aquel de valor cero “EE39”
- codigo_facilidad: Codigo de la facilidad que se activo

Nombre Regla	Fuera “**Nombre Plan**” “**Horario**”
Objetos Entrada	
	RegistroLlamadas.`llam_cant_seg`
	RegistroLlamadas.`tipo_fono`
	RegistroLlamadas.area_tarifaria
	RegistroLlamadas.`llam_horario`
	RegistroLlamadas.`flag_pase`
	FacilidadesAni.codigo_facilidad
	FacilidadesAni.segs_slm_hn
	FacilidadesAni.segs_slm_he
	FacilidadesAni.segs_slm_hv
	FacilidadesAni.proporcion
Objetos Salida	RegistroLlamadas.llam_cant_seg_dentplan
	RegistroLlamadas.llam_cant_seg_fueraplan
	RegistroLlamadas.cli_concepto_cobro
	RegistroLlamadas.codigo_facilidad
	RegistroLlamadas.flag_pase
Prioridad	95

Descripción

Regla para las llamadas SLM de teléfonos fijos que están fuera de plan (se consumieron los minutos del plan), donde se le cobra la cantidad de segundos de la llamada de acuerdo al valor de concepto por área tarifaria.

Nombre Regla: Fuera “Nombre Plan” “Horario”(HN:horario normal, HE: horario económico, HV: horario vespertino)

El detalle de los objetos retornados es:

- llam_cant_seg_dentplan: Valor en cero, la llamada esta fuera de plan.
- llam_cant_seg_fueraplan: Cantidad de segundos de la llamada.
- cli_concepto_cobro: valor concepto a cobrar, en este caso es aquel correspondiente al plan de acuerdo al area tarifaria.
- codigo_facilidad: El codigo de la facilidad en el que cayó la llamada.

Planes Mixtos

Nombre Regla	Dentro Plan “**Nombre de Plan**” “**Horario**” “**Red**”
Objetos Entrada	
	RegistroLlamadas.`llam_cant_seg`
	RegistroLlamadas.`cat_servicio (*)`
	RegistroLlamadas.`tipo_fono`
	RegistroLlamadas.`llam_horario`
	RegistroLlamadas.`flag_pase`
	FacilidadesAni.codigo_facilidad
	FacilidadesAni.segs_slm_hn (**)
	FacilidadesAni.segs_slm_he (**)
	FacilidadesAni.segs_slm_hv (**)
	FacilidadesAni.segs_mov_hn (***)
	FacilidadesAni.segs_mov_he (***)
	FacilidadesAni.segs_mov_hv (***)
	FacilidadesAni.proporcion
Objetos Salida	RegistroLlamadas.llam_cant_seg_dentplan
	RegistroLlamadas.llam_cant_seg_fueraplan
	RegistroLlamadas.cli_concepto_cobro
	RegistroLlamadas.codigo_facilidad
	RegistroLlamadas.flag_pase
Prioridad	95

Descripción

Regla para las llamadas dentro de plan mixto, donde no se cobra los segundos de la llamada.

Nombre Regla: Dentro Plan “Nombre del Plan” “Horario”(HN: horario normal, HE: horario económico, HV: horario vespertino) “Red”(red SLM o Movil).

El detalle de los objetos retornados es:

- llam_cant_seg_dentplan: la cantidad de segundos de la llamada dentro de plan
- llam_cant_seg_fueraplan: Valor cero, ya que la llamada esta 100% dentro de plan
- cli_concepto_cobro: valor concepto a cobrar, en este caso es aquel de valor cero “EE39”.
- codigo_facilidad: El codigo de la facilidad en el que cayó la llamada.
-

(*) variable que hace la diferencia entre teléfonos phs y fijos. seria conveniente que no fuera así deja la opción de dar a los dos tipos de teléfonos esa facilidad. Y no la restringe a un solo tipo de telefono.
(**) Variables cuando la llamada es de red fija
(***) Variable cuando la llamada es de red Movil

Nombre Regla	Divide LLamada “**Nombre de Plan**” “**Horario**” “**Red**”
Objetos Entrada	RegistroLlamadas.`llam_cant_seg`
	RegistroLlamadas.`cat_servicio`
	RegistroLlamadas.`tipo_fono`
	RegistroLlamadas.`llam_horario`
	RegistroLlamadas.`flag_pase`
	FacilidadesAni.codigo_facilidad
	FacilidadesAni.segs_slm_hn (*)
	FacilidadesAni.segs_slm_he (*)
	FacilidadesAni.segs_slm_hv (*)
	FacilidadesAni.segs_mov_hn (**)
	FacilidadesAni.segs_mov_he (**)
	FacilidadesAni.segs_mov_hv (**)
	FacilidadesAni.proporcion
Objetos Salida	RegistroLlamadas.llam_cant_seg_dentplan
	RegistroLlamadas.llam_cant_seg_fueraplan
	RegistroLlamadas.cli_concepto_cobro
	RegistroLlamadas.codigo_facilidad
	RegistroLlamadas.flag_pase
Prioridad	90

Descripción

Regla para las llamadas que sobrepasan la cantidad de segundos de la facilidad, teniendo segundos dentro de plan y fuera de plan los que se guardan en los objetos correspondientes.

Nombre Regla: Divide Llamada “Nombre de Plan” “Horario”(HN: horario normal, HE: horario económico, HV: horario vespertino) “Red”(red SLM o Móvil).

El detalle de los objetos que se retornan son:

- llam_cant_seg_dentplan: la cantidad de segundos de la llamada dentro de plan
- llam_cant_seg_fueraplan: cantidad de segundos fuera de plan.
- cli_concepto_cobro: valor concepto a cobrar, en este caso es aquel de valor cero “EE39”.
- codigo_facilidad: El codigo de la facilidad en el que cayó la llamada.

(*) Variables cuando la llamada es de red fija
(**) Variable cuando la llamada es de red Movil

Planes Onnet

Nombre Regla	Dentro Plan “**Nombre de Plan**” “**Horario**” “**Tipo_LLamada**”
Objetos Entrada	
	RegistroLlamadas.`llam_cant_seg`
	RegistroLlamadas.fono`_tipo_destino`
	RegistroLlamadas.dni_empresa_numero
	RegistroLlamadas.`llam_horario`
	RegistroLlamadas.`flag_pase`
	FacilidadesAni.codigo_facilidad
	FacilidadesAni.segs_slm_onnet (*)
	FacilidadesAni.segs_slm_offnet (**)
	FacilidadesAni.proporcion
Objetos Salida	RegistroLlamadas.llam_cant_seg_dentplan
	RegistroLlamadas.llam_cant_seg_fueraplan
	RegistroLlamadas.cli_concepto_cobro
	RegistroLlamadas.codigo_facilidad
	RegistroLlamadas.flag_pase
	RegistroLlamadas.`onoff`
Prioridad	95
Descripción	

Regla para las llamadas Onnet y Offnet, donde no se cobra los segundos de la llamada que están dentro de plan.

Nombre Regla: Dentro Plan “Nombre del Plan” “Horario”(HN: horario normal, HE: horario económico, HV: horario vespertino) “Tipo_Llamada”(Onnet o Offnetl).

El detalle de los objetos retornados es:

- llam_cant_seg_dentplan: la cantidad de segundos de la llamada dentro de plan
- llam_cant_seg_fueraplan: Valor cero, ya que la llamada esta 100% dentro de plan
- cli_concepto_cobro: valor concepto a cobrar, en este caso es aquel de valor cero “EE39”.
- codigo_facilidad: El codigo de la facilidad en el que cayo la llamada.
- Onoff: Valor en “0” para las llamadas onnet, y valor en “0” para las llamadas offnet.

(*) Variable para los segundos dentro de plan onnet
(**) Variable para los segundos dentro de plan offnet

Nombre Regla	Divide LLamada “**Nombre de Plan**” “**Horario**” “**Tipo_llamada**”
Objetos Entrada	
	RegistroLlamadas.`llam_cant_seg`
	RegistroLlamadas.`fono_tipo_destino`
	RegistroLlamadas.`dni_empresa_numero`
	RegistroLlamadas.`llam_horario`
	RegistroLlamadas.`flag_pase`
	FacilidadesAni.codigo_facilidad
	FacilidadesAni.segs_slm_onnet
	FacilidadesAni.segs_slm_offnet
	FacilidadesAni.proporcion
Objetos Salida	RegistroLlamadas.llam_cant_seg_dentplan
	RegistroLlamadas.llam_cant_seg_fueraplan
	RegistroLlamadas.cli_concepto_cobro
	RegistroLlamadas.codigo_facilidad
	RegistroLlamadas.flag_pase
	RegistroLlamadas.`onoff`
Prioridad	90

Descripción

Regla para las llamadas que sobrepasan la cantidad de segundos de la facilidad, teniendo segundos dentro de plan y fuera de plan los que se guardan en los objetos correspondientes.

Nombre Regla: Divide Llamada “Nombre del Plan” “Horario” “Tipo_Llamada”(Onnet u Offnet).

El detalle de los objetos retornados es:

- llam_cant_seg_dentplan: la cantidad de segundos de la llamada dentro de plan
- llam_cant_seg_fueraplan: cantidad de segundos fuera de plan.
- cli_concepto_cobro: valor concepto a cobrar, en este caso es aquel de valor cero “EE39”.
- codigo_facilidad: El codigo de la facilidad en el que cayó la llamada.
- Onoff: Valor en “0” cuando la llamada es onnet y en “1” cuando es offnet.

Plan Estandar SLM

Nombre Regla	Llamada “**Tipo_SLM**” “**Horario**” “**Area**”
Objetos Entrada	
	RegistroLlamadas.`llam_cant_seg`
	RegistroLlamadas.`tipo_fono`
	RegistroLlamadas.`cat_servicio`
	RegistroLlamadas.area_tarifaria
	RegistroLlamadas.`llam_horario`
	RegistroLlamadas.`flag_pase`
Objetos Salida	RegistroLlamadas.llam_cant_seg_dentplan
	RegistroLlamadas.llam_cant_seg_fueraplan
	RegistroLlamadas.cli_concepto_cobro
	RegistroLlamadas.codigo_facilidad
	RegistroLlamadas.flag_pase
Prioridad	1

Descripción

Regla para las llamadas SLM de teléfonos fijos o telefonos PHS de clientes que no tienen plan u aquellas llamadas que están fuera de plan (se consumieron los minutos del plan), donde se le cobra la cantidad de segundos de la llamada de acuerdo al valor de concepto por area tarifaria.

Nombre Regla: Llamada “ Tipo_SLM”(SLM: para telefonos fijos, SPH: para telefonos PHS) “Horario”(HN:horario normal, HE: horario económico, HV: horario vespertino) “Area”(Area tarifaria: AT1,AT2…),

El detalle de los objetos retornados es:

- llam_cant_seg_dentplan:Valor en cero, ya que la llamada no cae en ninfun plan o esta fuera de plan.
- llam_cant_seg_fueraplan: Cantidad de segundos de la llamada.
- cli_concepto_cobro: Valor del concepto a cobrar, de acuerdo al area tarifaria y horario.
- codigo_facilidad: El codigo de la facilidad en el que cayo la llamada.

Plan Estandar Movil

Nombre Regla	Llamada a “**Tipo_Tramo_Local**” “**Horario**” “**Area**”
Objetos Entrada	
	RegistroLlamadas.`llam_cant_seg`
	RegistroLlamadas.`tipo_fono`
	RegistroLlamadas.`cat_servicio`
	RegistroLlamadas.area_tarifaria
	RegistroLlamadas.`llam_horario`
	RegistroLlamadas.`flag_pase`
Objetos Salida	RegistroLlamadas.llam_cant_seg_dentplan
	RegistroLlamadas.llam_cant_seg_fueraplan
	RegistroLlamadas.cli_concepto_cobro
	RegistroLlamadas.codigo_facilidad
	RegistroLlamadas.flag_pase
Prioridad	1

Descripción

Regla para las llamadas a Moviles de telefonos fijos o phs de clientes que no tienen plan u aquellas llamadas que están fuera de plan (se consumieron los minutos del plan), donde se le cobra la cantidad de segundos de la llamada de acuerdo al valor de concepto por area tarifaria.

Nombre Regla: Llamada a “Tipo_Tramo_Local”(TL, TPH) “Horario”(HN:horario normal, HE: horario económico, HV: horario vespertino) “Area”(Area tarifaria: AT1,AT2…),

El detalle de los objetos retornados es:

- llam_cant_seg_dentplan: la cantidad de segundos de la llamada dentro de plan
- llam_cant_seg_fueraplan: Valor cero, ya que la llamada esta 100% dentro de plan
- cli_concepto_cobro: valor concepto a cobrar, en este caso es aquel de valor cero “EE39” ya que la cantidad de segundos fuera de plan vuelve a pasar por las reglas.
- codigo_facilidad: El codigo de la facilidad en el que cayo la llamada.

Planes Estandar 600

Nombre Regla	Llamada 600 “**Horario**” “**Area**”
Objetos Entrada	
	RegistroLlamadas.`llam_cant_seg`
	RegistroLlamadas.`tipo_fono`
	RegistroLlamadas.area_tarifaria
	RegistroLlamadas.`llam_horario`
	RegistroLlamadas.`flag_pase`
Objetos Salida	RegistroLlamadas.llam_cant_seg_dentplan
	RegistroLlamadas.llam_cant_seg_fueraplan
	RegistroLlamadas.cli_concepto_cobro
	RegistroLlamadas.codigo_facilidad
	RegistroLlamadas.flag_pase
Prioridad	95

Descripción

Regla para las llamadas a línea 600, donde se le cobra la cantidad de segundos de la llamada de acuerdo al valor de concepto por área tarifaria.

Nombre Regla: Llamada 600 “Horario”(HN:horario normal, HE: horario económico, HV: horario vespertino) “Area”(Area tarifaria: AT1,AT2…),

El detalle de los objetos retornados es:

- llam_cant_seg_dentplan: Valor cero, ya que la llamada esta fuera de plan
- llam_cant_seg_fueraplan: la cantidad de segundos de la llamada a línea 600
- cli_concepto_cobro: valor concepto a cobrar para la llamada 600.
- codigo_facilidad: El codigo de la facilidad en el que cayó la llamada.

Planes SLMQLP

Nombre Regla	Dentro “**Nombre Plan**” “**Horario**” “Movil”
Objetos Entrada	
	RegistroLlamadas.`llam_cant_seg`
	RegistroLlamadas.`llam_horario`
	RegistroLlamadas.`flag_pase`
	FacilidadesAni.codigo_facilidad
	FacilidadesAni.segs_slm_hn
	FacilidadesAni.segs_slm_he
	FacilidadesAni.segs_slm_hv
	FacilidadesAni.segs_mov_hn
	FacilidadesAni.segs_mov_he
	FacilidadesAni.segs_mov_hv
	FacilidadesAni.proporcion
Objetos Salida	RegistroLlamadas.llam_cant_seg_dentplan
	RegistroLlamadas.llam_cant_seg_fueraplan
	RegistroLlamadas.cli_concepto_cobro
	RegistroLlamadas.codigo_facilidad
	RegistroLlamadas.flag_pase
Prioridad	95

Descripción

Regla para los planes con minutos incluidos para llamadas a telefonos fijos y moviles desde teléfonos PHS, donde la proporcion en minutos de una llamada a red fija y a moviles es 1:3 (uno es a tres) respectivamente.

Nombre Regla: Dentro “Nombre del plan” “Horario”(HN:horario normal, HE: horario económico, HV: horario vespertino, TH: todo horario) “Movil”(si la llamada es a un movil, en caso contrario no va nada)

El detalle de los objetos retornados es:

- llam_cant_seg_dentplan: cantidad de segundos de la llamada.
- llam_cant_seg_fueraplan: Valor en cero, ya que la llamada esta dentro de plan.
- cli_concepto_cobro: si la llamada es de un telefono de red fija en codigo de concepto es “EE39”y si la llamada es a un telefono movil el codigo de concepto y el codigo por cargo de acceso es “G6SG” y respectivamente “G6RY”
- codigo_facilidad: El codigo de la facilidad en el que cayó la llamada.

Nombre Regla	Divide LLamada “**Nombre de Plan**” “**Horario**” “Movil”
Objetos Entrada	
	RegistroLlamadas.`llam_cant_seg`
	RegistroLlamadas.`llam_horario`
	RegistroLlamadas.`flag_pase`
	FacilidadesAni.codigo_facilidad
	FacilidadesAni.segs_slm_hn
	FacilidadesAni.segs_slm_he
	FacilidadesAni.segs_slm_hv
	FacilidadesAni.segs_mov_hn
	FacilidadesAni.segs_mov_he
	FacilidadesAni.segs_mov_hv
	FacilidadesAni.proporcion
Objetos Salida	RegistroLlamadas.llam_cant_seg_dentplan
	RegistroLlamadas.llam_cant_seg_fueraplan
	RegistroLlamadas.cli_concepto_cobro
	RegistroLlamadas.codigo_facilidad
	RegistroLlamadas.flag_pase
Prioridad	90

Descripción

Regla para las llamadas que sobrepasan el tipo de plan SLQL.

Nombre Regla: Divide Llamada “Nombre del Plan” “Horario” (HN:horario normal, HE: horario económico, HV: horario vespertino, TH: todo horario) “Movil” (si la llamada es a un movil, en caso contrario no va nada)
El detalle de los objetos retornados es:

- llam_cant_seg_dentplan: la cantidad de segundos de la llamada dentro de plan
- llam_cant_seg_fueraplan: cantidad de segundos fuera de plan.
- cli_concepto_cobro: si la llamada es de un telefono de red fija en codigo de concepto es “EE39”y si la llamada es a un telefono movil el codigo de concepto y el codigo por cargo de acceso es “G6SG” y respectivamente “G6RY”
- codigo_facilidad: El codigo de la facilidad en el que cayó la llamada.

Planes Renta Plana:

Nombre Regla	Dentro “**Nombre Plan**” “**Horario**”
Objetos Entrada	
	RegistroLlamadas.`llam_cant_seg`
	RegistroLlamadas.`tipo_fono`
	RegistroLlamadas.`llam_horario`
	RegistroLlamadas.`flag_pase`
	FacilidadesAni.codigo_facilidad
	FacilidadesAni.segs_slm_hn
	FacilidadesAni.segs_slm_he
	FacilidadesAni.segs_slm_hv
	FacilidadesAni.proporcion
Objetos Salida	RegistroLlamadas.llam_cant_seg_dentplan
	RegistroLlamadas.llam_cant_seg_fueraplan
	RegistroLlamadas.cli_concepto_cobro
	RegistroLlamadas.codigo_facilidad
	RegistroLlamadas.flag_pase
Prioridad	95

Descripción

Regla para las llamadas dentro de plan renta plana, donde los minutos dentro del plan no se cobran.

Nombre Regla: Dentro “Nombre del plan” “Horario”(HN:horario normal, HE: horario económico, HV: horario vespertino, HT: todo horario)

El detalle de los objetos retornados es:

- llam_cant_seg_dentplan: Cantidad de segundos de la llamada.
- llam_cant_seg_fueraplan: Valor en cero, ya que la llamada esta dentro de plan.
- cli_concepto_cobro: valor concepto dentro de plan. “EE39”
- codigo_facilidad: El codigo de la facilidad en el que cayó la llamada.

Nombre Regla	Divide LLamada “**Nombre de Plan**” “**Horario**”
Objetos Entrada	
	RegistroLlamadas.`llam_cant_seg`
	RegistroLlamadas.`tipo_fono`
	RegistroLlamadas.`llam_horario`
	RegistroLlamadas.`flag_pase`
	FacilidadesAni.codigo_facilidad
	FacilidadesAni.segs_slm_hn
	FacilidadesAni.segs_slm_he
	FacilidadesAni.segs_slm_hv
	FacilidadesAni.proporcion
Objetos Salida	RegistroLlamadas.llam_cant_seg_dentplan
	RegistroLlamadas.llam_cant_seg_fueraplan
	RegistroLlamadas.cli_concepto_cobro
	RegistroLlamadas.codigo_facilidad
	RegistroLlamadas.flag_pase
Prioridad	90

Descripción

Regla para las llamadas que sobrepasan los minutos del plan renta plana.

Nombre Regla: Divide Llamada “Nombre del Plan” “Horario”

El detalle de los objetos retornados es:

- llam_cant_seg_dentplan: la cantidad de segundos de la llamada dentro de plan
- llam_cant_seg_fueraplan: cantidad de segundos fuera de plan.
- cli_concepto_cobro: valor concepto a cobrar, en este caso es aquel de valor cero “EE39” ya que la cantidad de segundos fuera de plan vuelve a pasar por las reglas.
- codigo_facilidad: El codigo de la facilidad en el que cayó la llamada.

Planes Estandar Rural

Nombre Regla	Llamada Rural “Trafico”“**Nombre Empresa**” “**Horario**” “Area Tarifaria”
Objetos Entrada	
	RegistroLlamadas.`llam_cant_seg`
	RegistroLlamadas.`tipo_fono`
	RegistroLlamadas.`llam_horario`
	RegistroLlamadas.`flag_pase`
	RegistroLlamadas.`area_tarifaria`
	RegistroLlamadas.`cat_servicio`
Objetos Salida	RegistroLlamadas.llam_cant_seg_dentplan
	RegistroLlamadas.llam_cant_seg_fueraplan
	RegistroLlamadas.cli_concepto_cobro
	RegistroLlamadas.codigo_facilidad
	RegistroLlamadas.flag_pase
Prioridad	1
Descripción	
Regla para las llamadas efectuadas a red rural, donde se cobran los segundos de acuerdo al valor del concepto por empresa. **Nombre Regla:** Llamada Rural “Trafico”(TL o TPH) “Nombre Empresa” “Horario”(HN:horario normal, HE: horario económico, HV: horario vespertino) El detalle de los objetos retornados es: - llam_cant_seg_dentplan: Valor en cero, la llamada se considera fuera de plan. - llam_cant_seg_fueraplan: Cantidad de segundos de la llamada. - cli_concepto_cobro: valor concepto a cobrar. - codigo_facilidad: El codigo de la facilidad en el que cayó la llamada.	

7.3 DSL

Los lenguajes DSL implementado por Package en el prototipo se detallan a continuación:

Package DivideLlamada

[when]Una llamada: = `l: LlamadaDividida($c:llam_cant_seg)`

[when]Es hecha entre Lunes y Viernes = `LlamadaDividida(eval(llam_dia_semana >= 1 && llam_dia_semana <= 5 && llam_tipo_dia == "H" ))`

[when]Es hecha un dia Sabado = `LlamadaDividida(eval(llam_dia_semana == 6 && llam_tipo_dia == "H"))`

[when]Es hecha un dia Domingo o Festivo = `LlamadaDividida(eval(llam_dia_semana == 7 || llam_tipo_dia == "F"))`

[when]Fue hecha entre las {hora} : {minutos} : {segundos} hrs y las {hora2} : {minutos2} : {segundos2} hrs =`LlamadaDividida(eval(llam_fecha_comuni >= (({hora} * 3600) + ({minutos} * 60) + {segundos})),eval(llam_fecha_comuni <= (({hora2} * 3600) + ({minutos2} * 60) + {segundos2})))`

[when]Termino despues de las {hora} : {minutos} : {segundos} hrs = `LlamadaDividida(eval((llam_fecha_comuni + llam_cant_seg) > (({hora} * 3600) + ({minutos} * 60) + {segundos})))`

[when]Termino antes de las {hora} : {minutos}: {segundos} hrs = `LlamadaDividida(eval((llam_fecha_comuni + llam_cant_seg) <= (({hora} * 3600) + ({minutos} * 60) + {segundos})))`

[then]Se guarda la cantidad de segundos a tasar antes y despues de las {hora} : {minutos} : {segundos} hrs = `l.setLlam_cant_seg_sobrante(($c + l.getLlam_fecha_comuni()) - (({hora} * 3600) + ({minutos} * 60) + {segundos}));l.setLlam_cant_seg((({hora} * 3600) + ({minutos} * 60) + {segundos}) - l.getLlam_fecha_comuni());`

[then]La hora de inicio de comunicacion sobrante es {hora} : {minutos} : {segundos} = `l.setLlam_hora_comuni_sobrante({hora},{minutos},{segundos});`

[then]La llamada no se divide = `l.setLlam_cant_seg_sobrante(0);`

[then]La llamada pertenece a horario vespertino =`l.setLlam_horario("V");`

[then]La llamada pertenece a horario normal =`l.setLlam_horario("N");`

[then]La llamada pertenece a horario economico =`l.setLlam_horario("E");`

[then]Print "{mensaje}" =`System.out.println("{mensaje}");`

[then]Print2 = `System.out.println("llam_fecha_comuni: " + l.getLlam_fecha_comuni());System.out.println("suma horas con 0:  " + (07*3600));System.out.println("suma minutos " + (50* 60));System.out.println("suma " + ((7*3600) + (59*60)));`

Package Tasacion Llamadas

[when]En una LLamada `= r: RegistroLlamadas($FonoAni: ani_numero,$FonoDni: dni_numero, flag_pase == "0", $cant_segundos :llam_cant_seg )`

[when]-La llamada es realizada por un telefono PHS `= cat_servicio == "38"`

[when]-La llamada es realizada por un telefono Fijo `= cat_servicio != "38"`

[when]-La cantidad de segundos de la llamada sea menor a {llam_cant_seg} `= llam_cant_seg < {llam_cant_seg}`

[when]-La llamada es a red fija `= tipo_fono == 0`

[when]-La llamada es a un telefono movil `= tipo_fono == 1`

[when]-La llamada es a linea 700`= tipo_fono == 2`

[when]-La llamada es a linea 600`= tipo_fono == 3`

[when]-La llamada es a Red Rural`= tipo_fono == 4`

[when]-La llamada es Onnet `= fono_tipo_destino == "LOCAL TELSUR" || dni_empresa_numero == 0`

[when]-La llamada es Offnet `= fono_tipo_destino != "LOCAL TELSUR" || dni_empresa_numero != 0, tipo_fono ==0`

[when]-La llamada es en horario normal `= llam_horario == "N"`

[when]-La llamada es en horario economico `= llam_horario == "E"`

[when]-La llamada es en horario vespertino `= llam_horario == "V"`

[when]-La llamada es en horario reducido `= llam_horario == "E" || llam_horario == "V"`

[when]-La llamada es en todo horario `= llam_horario == "N" || llam_horario == "E" || llam_horario == "V"`

[when]-La llamada es del area tarifaria {area_tarifaria} `= area_tarifaria == {area_tarifaria}`

[when]En una facilidad origen `= f: FacilidadesAni($fonofac: fono_facilidad,$segs_slm_hn: segs_slm_hn, $segs_slm_he: segs_slm_he, $segs_slm_hv: segs_slm_hv,$segs_mov_hn: segs_mov_hn, $segs_mov_he: segs_mov_he, $segs_mov_hv: segs_mov_hv,$proporcion: proporcion, $segs_slm_onnet: segs_slm_onnet, $segs_slm_offnet: segs_slm_offnet)`

[when]-el numero origen tiene la facilidad de numero frecuente `= $codigoFacilidad: codigo_facilidad, ani_numero == $FonoAni, codigo_facilidad == "1581" || codigo_facilidad == "1582" || codigo_facilidad == "1583" || codigo_facilidad == "1584" ||codigo_facilidad == "1585" || codigo_facilidad == "1714" || codigo_facilidad == "1715" ||codigo_facilidad == "1716" ||codigo_facilidad == "1717" ||codigo_facilidad == "1718" || codigo_facilidad == "715"`

[when]-el numero origen tiene la facilidad de prepago `= $codigoFacilidad: codigo_facilidad, ani_numero == $FonoAni, codigo_facilidad == "1260" || codigo_facilidad == "1261" || codigo_facilidad == "1278" || codigo_facilidad == "1324" ||codigo_facilidad == "1325" || codigo_facilidad == "1368" || codigo_facilidad == "1530" ||codigo_facilidad == "1628" ||codigo_facilidad == "1702" ||codigo_facilidad == "1909" || codigo_facilidad == "1710" || codigo_facilidad == "1711" || codigo_facilidad == "1712" || codigo_facilidad == "1719" ||codigo_facilidad == "1720" || codigo_facilidad == "1721" || codigo_facilidad == "1722" ||codigo_facilidad == "1723" ||codigo_facilidad == "1733" ||codigo_facilidad == "1766" || codigo_facilidad == "1767" || codigo_facilidad == "1768" ||codigo_facilidad == "1777" ||`

codigo_facilidad == "1839" || codigo_facilidad == "1840"
||codigo_facilidad == "1866" ||codigo_facilidad == "1867"
||codigo_facilidad == "1930" || codigo_facilidad == "1975" ||
codigo_facilidad == "1992" || codigo_facilidad == "1993" ||
codigo_facilidad == "1998" ||codigo_facilidad == "1999" ||
codigo_facilidad == "2000" || codigo_facilidad == "2018"
||codigo_facilidad == "2019" ||codigo_facilidad == "2020"
||codigo_facilidad == "2021" || codigo_facilidad == "2022"
||codigo_facilidad == "2023" ||codigo_facilidad == "888"

[when]-el numero origen esta en la facilidad {facilidad} = ani_numero == $FonoAni, $codigoFacilidad: codigo_facilidad == {facilidad}

[when]-el fono facilidad sea igual al fono destino = fono_facilidad == $FonoDni

[when]-la llamada este dentro del total de {minutos} minutos de la facilidad {codigo} en horario normal = $codigoFacilidad:codigo_facilidad == {codigo} , eval(($cant_segundos + $segs_slm_hn) <= ({minutos}*60 *proporcion))

[when]-la llamada este dentro del total de {minutos}minutos de la facilidad {codigo}en horario reducido = $codigoFacilidad:codigo_facilidad == {codigo} , eval(($cant_segundos + $segs_slm_he + $segs_slm_hv)<= ({minutos}*60*proporcion))

[when]-la llamada este dentro del total de {minutos}minutos de la facilidad {codigo}en todo horario = $codigoFacilidad:codigo_facilidad == {codigo} , eval(($cant_segundos + $segs_slm_hn + $segs_slm_he + $segs_slm_hv)<= ({minutos}*60*proporcion))

[when]-la llamada este dentro del total de {minutos}minutos a moviles de la facilidad {codigo}en todo horario = $codigoFacilidad:codigo_facilidad == {codigo} , eval(($cant_segundos + $segs_mov_hn + $segs_mov_he + $segs_mov_hv)<= ({minutos}*60*proporcion))

[when]-la llamada este dentro del total de {minutos}minutos del tipo plan SLMQLP de la facilidad {codigo}en todo horario = $codigoFacilidad:codigo_facilidad == {codigo} , eval((($cant_segundos*3) + $segs_slm_hn + $segs_slm_he + $segs_slm_hv + $segs_mov_hn + $segs_mov_he + $segs_mov_hv)<= ({minutos}*60*proporcion))

[when]-la llamada este dentro del total de {minutos}min. a red fija de la facilidad {codigo}en todo horario = $codigoFacilidad:codigo_facilidad == {codigo} , eval(($cant_segundos + $segs_slm_hn + $segs_slm_he + $segs_slm_hv + $segs_mov_hn + $segs_mov_he + $segs_mov_hv)<= ({minutos}*60*proporcion))

[when]-la llamada es Onnet y este dentro del total de {minutos}minutos de la facilidad {codigo}en todo horario = $codigoFacilidad:codigo_facilidad == {codigo} , eval(($cant_segundos + $segs_slm_onnet)<= ({minutos}*60*proporcion))

[when]-la llamada es Offnet y este dentro del total de {minutos}minutos de la facilidad {codigo}en todo horario = $codigoFacilidad:codigo_facilidad == {codigo} , eval(($cant_segundos + $segs_slm_offnet)<= ({minutos}*60*proporcion))

[when] - la cantidad de segundos es ilimitada en todo horario en la facilidad {codigo} = `$codigoFacilidad:codigo_facilidad == {codigo}, eval( ($cant_segundos + $segs_slm_hn + $segs_slm_he + $segs_slm_hv )<= (999999999*proporcion))`

[when] - la llamada sobrepase el total de {minutos}minutos de la facilidad {codigo} en horario normal = `$codigoFacilidad:codigo_facilidad == {codigo}, eval( $segs_slm_hn < ({minutos}*60*proporcion)), eval( ($cant_segundos + $segs_slm_hn)> ({minutos}*60*proporcion))`

[when] - la llamada sobrepase el total {minutos}minutos de la facilidad {codigo} en horario reducido = `$codigoFacilidad:codigo_facilidad == {codigo}, eval( ( $segs_slm_he + $segs_slm_hv ) < ({minutos}*60*proporcion) ), eval( ($cant_segundos + $segs_slm_he + $segs_slm_hv)> ({minutos}*60*proporcion) )`

[when] - la llamada sobrepase el total {minutos}minutos de la facilidad {codigo} en todo horario = `$codigoFacilidad:codigo_facilidad == {codigo}, eval( ( $segs_slm_hn +$segs_slm_he + $segs_slm_hv ) < ({minutos}*60*proporcion) ), eval( ($cant_segundos + $segs_slm_hn + $segs_slm_he+ $segs_slm_hv)> ({minutos}*60*proporcion) )`

[when] - la llamada sobrepase el total de {minutos}min. a moviles de la facilidad {codigo} en todo horario = `$codigoFacilidad:codigo_facilidad == {codigo}, eval( ( segs_mov_hn +segs_mov_he + segs_mov_hv ) < ({minutos}*60*proporcion) ), eval( ($cant_segundos + segs_mov_hn + segs_mov_he+ segs_mov_hv)> ({minutos}*60*proporcion) )`

[when] - la llamada sobrepase el total de {minutos}min. con llamadas a red fija de la facilidad {codigo} en todo horario = `$codigoFacilidad:codigo_facilidad == {codigo}, eval(($segs_slm_hn +$segs_slm_he + $segs_slm_hv + segs_mov_hn +segs_mov_he + segs_mov_hv) < ({minutos}*60*proporcion)), eval(($cant_segundos + $segs_slm_hn +$segs_slm_he + $segs_slm_hv + segs_mov_hn + segs_mov_he+ segs_mov_hv)> ({minutos}*60*proporcion))`

[when] - la llamada sobrepase el total de {minutos}min. con llamadas a moviles de la facilidad {codigo} en todo horario = `$codigoFacilidad:codigo_facilidad == {codigo}, eval(($segs_slm_hn +$segs_slm_he + $segs_slm_hv + segs_mov_hn +segs_mov_he + segs_mov_hv) < ({minutos}*60*proporcion)), eval((($cant_segundos*3) + segs_mov_hn + segs_mov_he+ segs_mov_hv+$segs_slm_hn +$segs_slm_he + $segs_slm_hv)> ({minutos} *60*proporcion))`

[when] - la llamada es Onnet y sobrepase el total de {minutos}minutos de la facilidad {codigo} en todo horario = `$codigoFacilidad:codigo_facilidad == {codigo}, eval( ( $segs_slm_onnet ) < ({minutos}*60*proporcion) ), eval( ($cant_segundos + $segs_slm_onnet)> ({minutos}*60*proporcion) )`

[when] - la llamada es Offnet y sobrepase el total de {minutos}minutos de la facilidad {codigo} en todo horario = `$codigoFacilidad:codigo_facilidad == {codigo}, eval( (`

$segs_slm_offnet) < ({minutos}*60*proporcion)), eval(($cant_segundos + $segs_slm_offnet)> ({minutos}*60*proporcion))

[when]-Se haya sobrepasado el total de {minutos}minutos de la facilidad {codigo} en horario normal = $codigoFacilidad:codigo_facilidad == {codigo}, eval(segs_slm_hn == ({minutos}*60*proporcion))

[when]-Se haya sobrepasado el total de {minutos}minutos de la facilidad {codigo} en horario reducido = $codigoFacilidad:codigo_facilidad == {codigo}, eval(($segs_slm_he + $segs_slm_hv) == ({minutos}*60*proporcion))

[when]-Se haya sobrepasado el total de {minutos}minutosde la facilidad {codigo} en todo horario = $codigoFacilidad:codigo_facilidad == {codigo}, eval(($segs_slm_hn + $segs_slm_he + $segs_slm_hv) == ({segundos}*proporcion))

[when]En una facilidad Destino = FacilidadesDni()

[when]-el numero destino esta en la facilidad {facilidad} = dni_numero == $FonoDni, $cod: codigo_facilidad == {facilidad}

[when]-el fono origen y fono destino pertenecen al mismo codigo centrex = fono_facilidad == $fonofac

[then]No se cobra "{message}"=System.out.println("{message}");

[then]No se tasa=r.setLlam_cant_seg_dentplan($cant_segundos);
r.setLlam_cant_seg_fueraplan(new Integer(0));
r.setCli_concepto_cobro("EE39"); r.setTasado("N");

[then]Salida variables= System.out.println(" Cantidad de segundos dentro de plan "+r.getLlam_cant_seg_dentplan()); System.out.println(" Cantidad de segundos fuera de plan "+r.getLlam_cant_seg_fueraplan());
System.out.println(" area tarifaria "+r.getArea_tarifaria());
System.out.println(" segundos acumulados en hn "+r.getSegs_slm_hn());System.out.println(" segundos sueltos "+r.getLlam_cant_seg ());

[then]No se cobra la llamada = r.setLlam_cant_seg_dentplan($cant_segundos);
r.setLlam_cant_seg_fueraplan(new Integer(0));
r.setCli_concepto_cobro("EE39");
r.setCodigo_facilidad($codigoFacilidad);

[then]No se cobra a moviles la llamada del tipo de plan SLMQLP =
r.setLlam_cant_seg_dentplan($cant_segundos*3);
r.setLlam_cant_seg_fueraplan(new Integer(0));
r.setCli_concepto_cobro("G6SG"); r.setCli_concepto_cobro_cacc("G6RY");
r.setCodigo_facilidad($codigoFacilidad);

[then]La llamada a Movil no se cobra =
r.setLlam_cant_seg_dentplan($cant_segundos);
r.setLlam_cant_seg_fueraplan(new Integer(0));
r.setCli_concepto_cobro("G6SG"); r.setCli_concepto_cobro_cacc("G6RY");
r.setCodigo_facilidad($codigoFacilidad);

[then]La llamada no se cobra es Onnet =

```
r.setLlam_cant_seg_dentplan($cant_segundos);
r.setLlam_cant_seg_fueraplan(new Integer(0));
r.setCli_concepto_cobro("EE39"); r.setCodigo_facilidad(
$codigoFacilidad); r.setOnoff(new Integer(0));
```

[then]La llamada no se cobra es Offnet =

```
r.setLlam_cant_seg_dentplan($cant_segundos);
r.setLlam_cant_seg_fueraplan(new Integer(0));
r.setCli_concepto_cobro("EE39"); r.setCodigo_facilidad(
$codigoFacilidad); r.setOnoff(new Integer(1));
```

[then]No se cobran los minutos en horario normal dentro del plan de {minutos} min.

```
=r.setLlam_cant_seg_dentplan(new Integer($cant_segundos-($cant_segundos
+ $segs_slm_hn - ({minutos}*60*$proporcion))));
r.setLlam_cant_seg_fueraplan(new Integer($cant_segundos + $segs_slm_hn
- ({minutos}*60*$proporcion))); r.setCodigo_facilidad(
$codigoFacilidad);  r.setCli_concepto_cobro("EE39");
```

[then]No se cobran los minutos en horario reducido dentro del plan de {minutos}min. =

```
r.setCli_concepto_cobro("EE39");    r.setLlam_cant_seg_fueraplan(new
Integer($cant_segundos + $segs_slm_he + $segs_slm_hv -
({minutos}*60*$proporcion)));  r.setLlam_cant_seg_dentplan(new
Integer($cant_segundos-($cant_segundos + $segs_slm_he + $segs_slm_hv -
({minutos}*60*$proporcion))));  r.setCodigo_facilidad(
$codigoFacilidad);
```

[then]No se cobran los minutos en todo horario dentro del plan de {minutos}min. =

```
r.setCli_concepto_cobro("EE39");  r.setLlam_cant_seg_dentplan(new
Integer($cant_segundos-($cant_segundos + $segs_slm_hn + $segs_slm_he +
$segs_slm_hv - ({minutos}*60*$proporcion))));
r.setLlam_cant_seg_fueraplan(new Integer($cant_segundos +$segs_slm_hn +
$segs_slm_he + $segs_slm_hv - ({minutos}*60*$proporcion)));
r.setCodigo_facilidad( $codigoFacilidad);
```

[then]No se cobran los minutos a moviles en todo horario dentro del plan de {minutos}min. =

```
r.setCli_concepto_cobro("EE39");  r.setLlam_cant_seg_dentplan(new
Integer($cant_segundos-($cant_segundos + $segs_mov_hn + $segs_mov_he +
$segs_mov_hv - ({minutos}*60*$proporcion))));
r.setLlam_cant_seg_fueraplan(new Integer($cant_segundos +$segs_mov_hn +
$segs_mov_he + $segs_mov_hv - ({minutos}*60*$proporcion)));
r.setCodigo_facilidad( $codigoFacilidad);
```

[then]No se cobran los minutos en todo horario dentro del plan SLMQLP de {minutos}min. =

```
r.setCli_concepto_cobro("G6SG"); r.setCli_concepto_cobro_cacc("G6RY");
r.setLlam_cant_seg_dentplan(new Integer($cant_segundos-($cant_segundos
+ $segs_slm_hn + $segs_slm_he + $segs_slm_hv + $segs_mov_hn +
$segs_mov_he + $segs_mov_hv - ({minutos}*60*$proporcion))));
r.setLlam_cant_seg_fueraplan(new Integer( $cant_segundos -
```

```
(r.getLlam_cant_seg_dentplan()/3) )); r.setCodigo_facilidad(
$codigoFacilidad); System.out.println("dentro de plan
"+r.getLlam_cant_seg_dentplan()); System.out.println("dentro de plan
"+r.getLlam_cant_seg_fueraplan());
```

[then]No se cobran los minutos de PHS a moviles en todo horario dentro del plan de {minutos}min. =

```
r.setCli_concepto_cobro("G6SG"); r.setCli_concepto_cobro_cacc("G6RY");
r.setLlam_cant_seg_dentplan(new Integer($cant_segundos-($cant_segundos
+ $segs_mov_hn + $segs_mov_he + $segs_mov_hv -
({minutos}*60*$proporcion)))); r.setLlam_cant_seg_fueraplan(new
Integer($cant_segundos +$segs_mov_hn + $segs_mov_he + $segs_mov_hv -
({minutos}*60*$proporcion))); r.setCodigo_facilidad( $codigoFacilidad);
```

[then]No se cobran los minutos Onnet en todo horario dentro del plan de {minutos}min. =

```
r.setCli_concepto_cobro("EE39");  r.setLlam_cant_seg_dentplan(new
Integer($cant_segundos-($cant_segundos + $segs_slm_onnet -
({minutos}*60*$proporcion)))); r.setLlam_cant_seg_fueraplan(new
Integer($cant_segundos +$segs_slm_onnet - ({minutos}*60*$proporcion)));
r.setCodigo_facilidad( $codigoFacilidad); r.setOnoff(new Integer(0));
```

[then]No se cobran los minutos Offnet en todo horario dentro del plan de {minutos}min. =

```
r.setCli_concepto_cobro("EE39");  r.setLlam_cant_seg_dentplan(new
Integer($cant_segundos-($cant_segundos + $segs_slm_offnet -
({minutos}*60*$proporcion)))); r.setLlam_cant_seg_fueraplan(new
Integer($cant_segundos +$segs_slm_offnet -
({minutos}*60*$proporcion))); r.setCodigo_facilidad( $codigoFacilidad);
r.setOnoff(new Integer(1));
```

[then]El codigo del concepto a cobrar es {codigo} para el plan Todo Horario de {minutos}min. =

```
r.setCli_concepto_cobro_fueraplan("{codigo}");
r.setCli_concepto_cobro_dentplan("EE39");
r.setLlam_cant_seg_fueraplan($cant_segundos +$segs_slm_hn +
$segs_slm_he + $segs_slm_hv - ({minutos}*60*$proporcion));
r.setLlam_cant_seg_dentplan($cant_segundos-($cant_segundos +
$segs_slm_hn + $segs_slm_he + $segs_slm_hv -
({minutos}*60*porporcion))); r.setCodigo_facilidad_fueraplan(new
Integer(731)); r.setCodigo_facilidad_dentplan( $codigoFacilidad);
```

[then]El codigo del concepto a cobrar es {codigo} =

```
r.setCli_concepto_cobro("{codigo}");
r.setLlam_cant_seg_fueraplan($cant_segundos);
r.setLlam_cant_seg_dentplan(new Integer(0)); r.setCodigo_facilidad(new
Integer(731));
```

[then]El codigo del concepto a cobrar por cargo de acceso es {codigo} =

```
r.setCli_concepto_cobro_cacc("{codigo}");
```

[then]El codigo del concepto a cobrar para la facilidad {facilidad} es {codigo} =

```
r.setCli_concepto_cobro("{codigo}");
r.setLlam_cant_seg_fueraplan($cant_segundos);
```

```
r.setLlam_cant_seg_dentplan(new Integer(0)); r.setCodigo_facilidad(new
Integer({facilidad}));
```

[then]Actualizar = `r.setFlag_pase("-1"); update(r);`