INTEGRACION DE BPMN2.0 EN EL ANALISIS DE COMUNICACIONES, SOPORTE SOBRE LA HERRAMIENTA CASE ORYX

Zulma Maritza Zambrana Dávila

Directores de Master: Oscar Pastor,
Sergio España.

AGRADECIMIENTOS.

Gracias a Dios, por permitir escribir en este momento. A mis padres por el apoyo constante y la confianza. De igual un agradecimiento profundo a mis tutores su paciencia.

Resumen

La importancia de los Procesos de Negocio ha crecido en el último tiempo, desde el área de negocios hasta el área informática. De igual manera la rápida evolución de los negocios requiere el uso de estándares para asegurar que todas las áreas y departamentos en una organización puedan entender los procesos de la misma manera. La clara comunicación y documentación de los mismos es el primer paso para poderlos automatizar.

En el presente trabajo aborda el estándar BPMN 2.O y técnicas del Análisis de Comunicaciones.

BPMN2.0 es un estándar de O.M.G. es lo suficientemente comprensible para ser aplicado a cualquier negocio. Al mismo tiempo la ingeniería de requisitos propone el modelado de procesos de negocio desde una perspectiva comunicacional, mediante una técnica vinculada al análisis comunicacional. Esta técnica son las estructuras comunicativas o estructuras de mensaje.

En este trabajo el análisis de comunicaciones adopta el objeto message de los diagramas de coregrafia de BPMN2.0 para asociar con la técnica de estructuras comunicativas o estructuras de mensajes.

Hasta la fecha existen herramientas CASE para BPMN2.0, mas no hay ninguno para dar soporte a esta asociación de notaciones. Este hecho impulsa a extender una herramienta CASE en este caso Oryx, un proyecto académico de código abierto, la ampliación de la funcionalidad del editor a través de un mecanismo de plugin (núcleo pequeño, casi todas las funciones se implementan como plugins), la tecnología utilizada es la representación SVG y XHTML y del lado del cliente código está escrito en Javascript, código de servidor está escrito en Java.

1. Tabla de contenido

2.2 Tabla de Imágenes

3.Ilustración de Tablas.

CAPITULO I

1. Introduccion.

En sistemas de información para la gestión, las interacciones entre el sistema y el entorno son comunicaciones que se establecen intercambiando mensajes.

Muy a menudo, el conjunto de requisitos se organizan como simples listas enumeradas. Se considera que los requisitos de una estructura adecuada para el desarrollo de sistemas de información, más apropiado que una lista.

Estos mensajes están asociados a sucesos constructores que comunican la ocurrencia de nuevos acontecimientos en el sistema objeto que interesa a una organización. Los procesos de negocio tienen que ver sobre todo con los cambios que se producen en un objeto de negocio.

La importancia de los Procesos de Negocio ha crecido en el último tiempo, desde el área de negocios hasta el área informática, donde se apunta a automatizar toda su gestión y ejecución, permitiendo adaptarlo fácilmente y mejorarlo de manera continua.

Tal es el caso de Business Process Modeling Notation (BPMN) es la notación estándar más utilizado para los procesos de negocio de dibujo, atualmente mantenida por el OMG - Object Management Group. Desde enero del 2009, este modelo ha sido objeto de una revisión a fondo y luego evolucionar a la versión 2.0.

BPMN2.0, trae consigo muchos cambios a la mesa, lo más importante el aumento de la coherencia y la integración de la orquestación y la coreografía de una manera que hace BPMN 2.0 una gran elección para los motores de los procesos de negocio.

Procesos que se utiliza principalmente para la captura de las actividades, la decisión responsabilidades, el control y flujo de datos en procesos de negocio dentro de una organización. Sin embargo, en toda la Organización existen ajustes que se concentran en la interacción comportamiento entre los distintos interlocutores implicados. Los socios individuales pueden aplicar internamente los procesos que deseen, siempre y cuando su comportamiento de interacción se ajusta a la coreografía que se acuerde. Especialmente cuando se basa en Mensajes

electrónicos como medio para la interacción entre los diferentes socios, una exacta definición de formatos de mensaje y las secuencias de interacción.

También cada interacción comunicativa es un requisito para el sistema y dentro la Ingeniería de requisitos el método de análisis de comunicaciones, este tiene dos técnicas complementarias como son los diagramas de sucesos comunicativos y las estructuras de mensajes para especificar la comunicación.

Las estructuras de mensaje asociado a BPMN 2.0, en el presente trabajo se adopta los diagramas de coreografía (BPMN2.0), hecho que logra una extensión de BPMN2.0 asociando a los mensajes una posibilidad de agregar la estructura de mensaje. En la actualidad no se tiene un soporte para realizar este propósito. Razon por el cual se extiende el código de la herramienta CASE, que en este caso es Oryx. Este es un editor basado en web para el modelado de diferentes diagramas y entre ellos los diagramas de coreografia para los procesos de negocio.

El resto del documento está estructurado de la siguiente manera: La sección La sección 2 el Estado del arte. La sección 3 Diagramas de coreografía del estándar BPMN2.0 desde una perspectiva de comunicacional y las técnicas de comunicación: estructuras de mensaje como parte del Análisis de Comunicaciones. La sección 4: Extensión de un diagrama de coreografía (BPMN 2.0). La sección 5: Implementación mediante Oryx. Y la Sección 6: Conclusiones y Recomendaciones.

1.1 Motivación.

La rápida evolución de los negocios requiere el uso de estándares para asegurar que todas las áreas y departamentos en una organización puedan entender los procesos de la misma manera. BPMN 2.0 (Business Process Modeling Notation), el estándar de OMG. BPMN 2.0 es lo suficientemente comprensible para ser aplicado en cualquier negocio, además de que los diagramas de coreografia permiten modelar grandes cantidades de procesos negocio. A pesar de ello seria aum mas representativa si permitiera la especificación de mensajes.

Este hecho lleva a investigar el nivel comunicacional en la ingeniería de requisitos.

El Análisis de Comunicación propone una estructura de requisitos que permite refinamiento paso a paso se acercan a la descripción del sistema de información (España, 2009).

Motivando así en el análisis de comunicaciones utilizar esta notación desde una perspectiva comunicacional como es la técnica de Estructuras de Mensaje. Ademas del soporte elegido para extender su código y asi integrar las estruturas de mesaje es altamente expresivo (Oryx en su versión Open Source).

1.2 Objetivos de Investigación y Desarrollo.

Adoptar la notación de los diagrama de coreografía de BPMN 2.0 en el Análisis de Comunicaciones y asociar con la técnica de estructuras de mensaje a través de la extensión de una herramienta case como es Oryx; lograr así extender los diagrama de coreografía, hecho que permitirá representar de manera completa el intercambio de mensajes de los procesos de negocio.

1.3 Justificación.

Por una parte, BPMN2.0 es el nuevo estándar para el modelado de procesos de negocio, sirve para representar gráficamente las diferentes etapas de los procesos de las compañías. Debido a su simplicidad y eficiencia se ha convertido en el estándar de facto.

BPMN2.0 junto a sus diagramas de coreografía que pueden representar gran cantidad de procesos por su forma compacta, con la extensión orientada a la comunicación es un complemento adecuado para este estándar.

Al mismo tiempo para el análisis de comunicaciones adoptar un estándar defacto permite más usabilidad e importancia.

Sin embargo esta complementación sin una herramienta de soporte seria incompleta, razón por el cual en esta tesina se elige Oryx para la integración de la técnica comunicacional y la notación (BPMN2.0).

Oryx es un marco académico de código abierto muy completo para el modelado gráfico de procesos, es extensible, la mayor parte de su funcionalidad esta realizada mediante plugins, entonces también permite adicionar nuevas funciones a través de un mecanismo de plugin.

1.4 Estructura de la tesis.

En este capítulo se han definido la motivación y la justificación para el desarrollo de la presente tesis de máster, el contexto de trabajo y los objetivos a cumplir. Para los siguientes capítulos la tesina queda estructurada de la siguiente forma: el capítulo I con Introducción que describe la parte motivación justificación y objetivo de la tesina, el capitulo 2 presenta el estado del arte de los temas relacionados con esta tesis de máster, el capitulo 3 el análisis de los diagramas de coreografía del estándar BPMN2.0 desde una perspectiva de comunicacional y las técnicas de comunicación: estructuras de mensaje como parte del Análisis de Comunicaciones, el capitulo 4 extensión de un diagrama de coreografía (BPMN 2.0) con elementos de un modelo de estructura de mensaje, en el capítulo 5 Implementación con la herramienta Oryx y pruebas. Sección 6: Conclusiones y Recomendaciones.

3.1.Metodologia de la Investigacion.

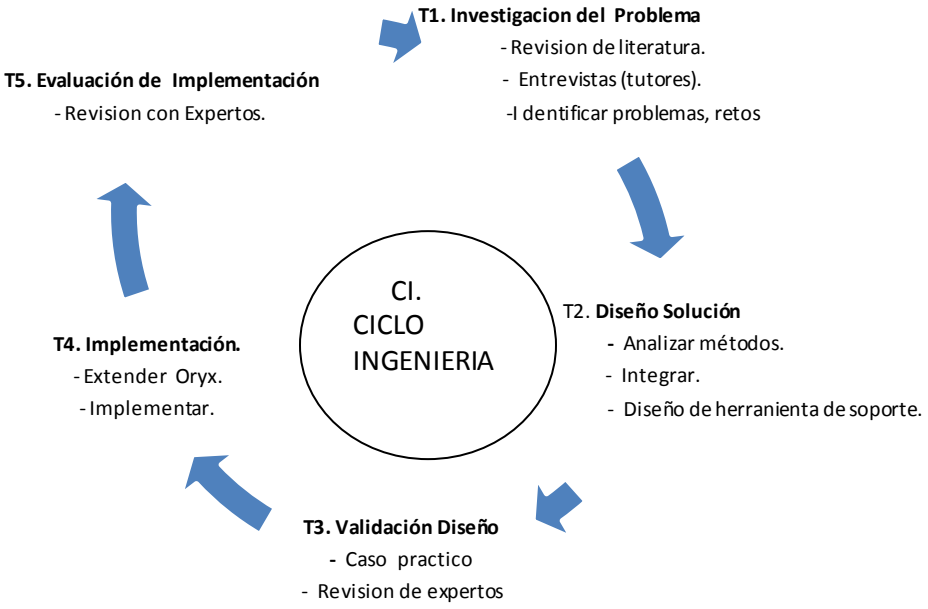


Fig. 1: Metodologia de Investigación seguida en la tesis

CI: Proponer una de integración para involucrar las técnicas del Analisis de Comunicaciones y el estandar BPMN2.0.

T1: Problemade investigación.

- Definir la motivación, el objetivo principal de la investigación (ver secciones 0 y 1.3).

T2: Diseño Solución.

- Estudio del estado del arte en notaciones de procesos de negocios, del soporte de los procesos de negocios disponibles (Ver capítulo 2).
- Diseño de un marco general de integración de involucrar el estándar con la técnica del Análisis de Comunicaciones.

T3: Validación de Diseño.

- Especificación de validación consiste en la aplicación de un caso práctico, revisión de los expertos (directores de master).

T4: Implementación.

- La extensión de la notación de BPMN 2.0 logrando integrar con la técnica de estructuras de mensaje dentro del Analisis de Comunicaciones.
- La extensión del código de la herramienta de soporte Oryx.

T5: Evaluación de la implementación.

- Revisión de la integración de notaciones de BPMN2.0 y Analisis de Comuniciones con expertos.

2. Estado del Arte.

En esta sección se presenta el estado del arte de los temas relacionados con el desarrollo de esta tesis de máster. Este estado del arte brinda la teoría necesaria acerca de los conceptos básicos requeridos para entender las diferentes notaciones de los procesos de negocio, asi como el método de análisis de comunicaciones y las herramientas case que puedan dar soporte.

2.1 Procesos de negocio.

Un proceso de negocio es un conjunto de actividades que se realizan para ofrecer valor a un cliente. Los procesos de negocio están siempre vinculados a sucesos y objetos. Un proceso describe el conjunto de cambios, o sucesos, a los que se somete un determinado objeto de negocio.

Un objeto de negocio es una estructura de información compleja que para los usuarios adquiere sentido por su composición y propiedades y por los procesos que lo manipulan.

Pueden ser objetos de negocio un pedido, una factura, un expediente de licitación o un préstamo. Cada uno de estos objetos de negocio podría ser descrito mediante un modelo de datos compuesto por varias clases de objetos.

Existen diferentes notaciones que permiten modelar procesos de negocio. Diagramas de Actividad de UML, Diagramas de Sucesos, BPMN, BPMN2.0. En este trabajo se estudia: BPMN2.0.

2.2. Diagramas de Actividad de UML.

UML (El lenguaje de modelado unificado) toma un perfil orientado a objetos en el modelado de aplicaciones.

Los Diagramas de Actividad son uno de los tres diagramas de UML(Unified Modeling Language), junto con los Diagramas de Estado y los Diagramas de Secuencia, utilizados para la descripción del comportamiento dinámico de un sistema.

Actualmente UML se encuentra en su versión 2.1.1.[17]. El paso de la versión 1.5 a la versión 2.0 supuso una gran revisión donde la parte que más se modificó es precisamente aquella que hace referencia a los Diagramas de Actividad [10].

2.2.1. Elementos de los Diagramas de Actividad.

En la versión actual los Diagramas están compuestos por una serie de elementos fundamentales, los nodos, que se pueden clasificar en:

Nodos de Acción: Realizan operaciones con los datos que reciben y pasan el control y datos a otras acciones. Además se distingue dos conceptos fundamentales, actividades y acciones.

Actividades: Agrupaciones de acciones que pueden poseer pre y post condición además de parámetros de entrada o de salida.

Acción: Son consumidores/productores de token que reciben datos y el control de flujo y traspasan estos elementos a otras acciones. Es la unidad mínima de comportamiento en los Diagramas de Actividad de UML 2.X.

Se observa un ejemplo de una actividad con parámetros, condiciones y las acciones que la componen en la

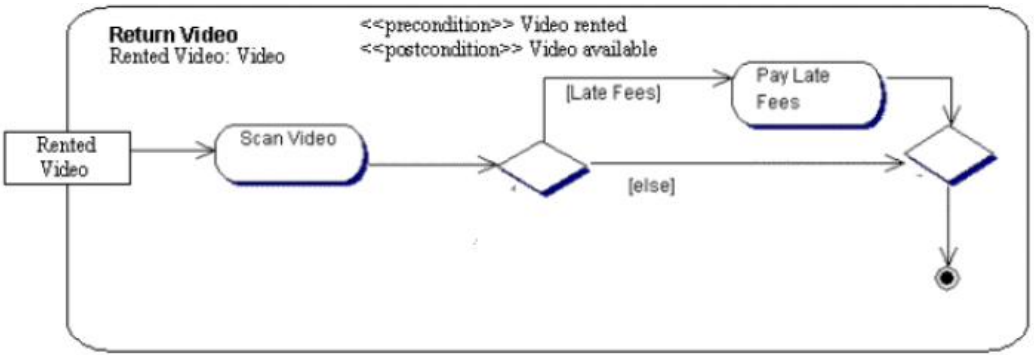


Fig. 2 Actividad con parámetros.

Nodos de Control: Distribuyen el control de la ejecución y los tokens a lo largo del diagrama. Existen distintos tipos de nodos de control como se ve en la Fig. 2.

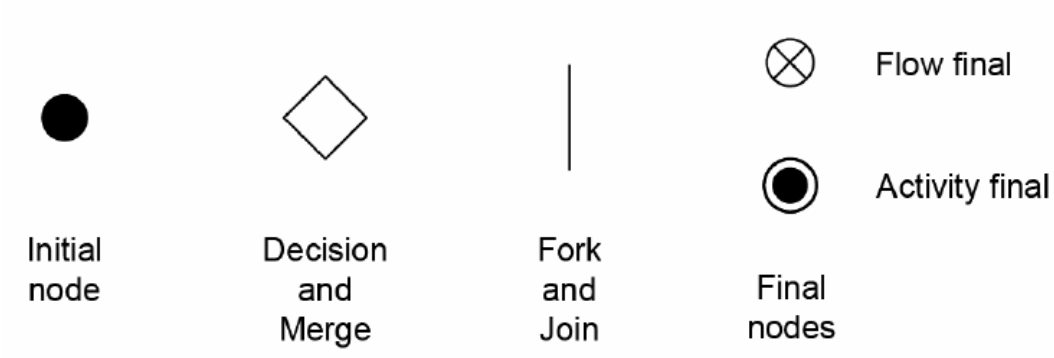


Fig. 3 Nodos de Control de los Diagramas de Actividad.

Initial Node: Nodo que recibe el control cuando comienza la ejecución de una actividad y que pasa inmediatamente dicho control a las acciones sucesivas. Es importante destacar que una actividad puede tener más de un nodo inicial y que si dicho nodo posee más de una transición de salida el control se pasa unicamente a una de dichas transiciones.

Decision Node: Guían el flujo en una u otra dirección. Esta dirección se decide en tiempo de ejecución al comprobar las condiciones de cada uno de los flujos salientes. Poseen un único flujo de entrada y varios flujos de salida que llevan condiciones asociadas.

Merge Node: Tiene la misma representación que el nodo anterior pero a diferencia de este tienen múltiples flujos entrada pero un único flujo de salida. Pasa de manera inmediata cualquier tipo de flujo que le llegue, ya sea de control o de datos, a su único flujo de salida, es decir, sirve para juntar varios flujos. Podemos combinar nodos Decision/Merge tal y como podemos apreciar en la

Fig. 3.

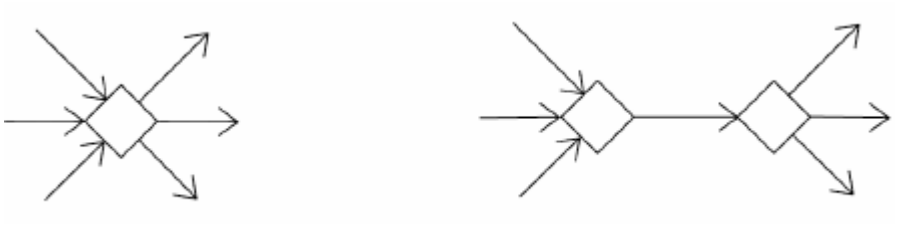


Fig. 4 Combinación Merge/Decisión.

Fork Node: Divide un único flujo de entrada en varios flujos de salida que se ejecutaran de manera concurrente. El control y los datos que llegan a este nodo son duplicados para cada uno de los flujos de salida.

Join Node: Para sincronizar múltiples flujos. Tienen la misma notación que los nodos Fork pero con la diferencia de tener varios flujos de entrada y único flujo de salida que unicamente se dispara cuando están disponibles todos los flujos de entrada. Podemos combinar nodos Fork/Join tal y como podemos apreciar en la Fig. 4.

Fig. 5 Combinación Fork/Join.

Flow Final: Recibe cualquier tipo de flujo, de control o de datos, y no hace nada, es decir destruye todo los tokens que le llegan. No tiene flujos de salida.

Activity Final: Al recibir un token acaba con todos los flujos de la actividad.

Nodos Objeto: Contienen datos de manera temporal a la espera de mover estos datos a lo largo del diagrama. Existen distintos tipos de nodos objetos que ilustra la Fig. 6

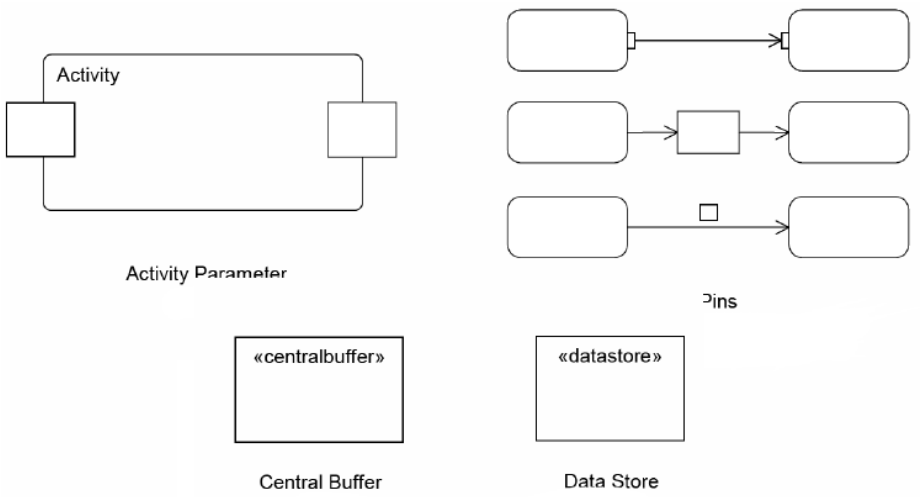


Fig. 6 Nodos Objeto.

Activity Parameter: Para representar datos de entrada o salida de una actividad completa. La actividad comienza cuando tiene disponibles todos sus parámetros de entrada y acaba cuando ha producido todos sus parámetros de salida.

Pins: Existen tres formas posibles de representarlos. Sirve para representar el paso de tokens de datos de una acción a otra.

Central Buffer: Sirven para evitar situaciones de carrera cuando los tokens vienen de diferentes fuentes. Acepta los tokens de los flujos de entrada y los pone disponibles a los flujos de salida. No se conecta directamente a las acciones si no que lo hace a través de pins.

Data Store: Son los nodos que nos proporciona la nueva especificación para dar soporte al comportamiento push de los diagramas de actividad anteriores. Almacenan tokens de datos de tal manera que estos no pueden ser borrados y están disponibles para que las acciones posteriores puedan comenzar a usarlos cuando estimen necesario. Si el objeto ya se encuentra en el data store es reemplazado.

Además de estos nodos en los Diagramas de Actividad disponemos de otro tipo de elementos y conceptos como:

Flujos: En los Diagramas de Actividad tenemos dos tipos de flujos:

El flujo de control que nos sirve para modelar el paso de una acción a otra. Tiene la misma notación que una transición en UML 1.X.

El flujo de datos que nos sirve para modelar el paso de información de una acción a otra.

Y para cada uno de estos flujos podemos, mediante etiquetas, especificar aspectos como: La multiplicidad (indica el número de tokens objeto que se consumen cada vez (como máximo o mínimo). El peso: Para indicar cuantos tokens de los disponibles consumo. La figura 6 ilustra estos flujos.

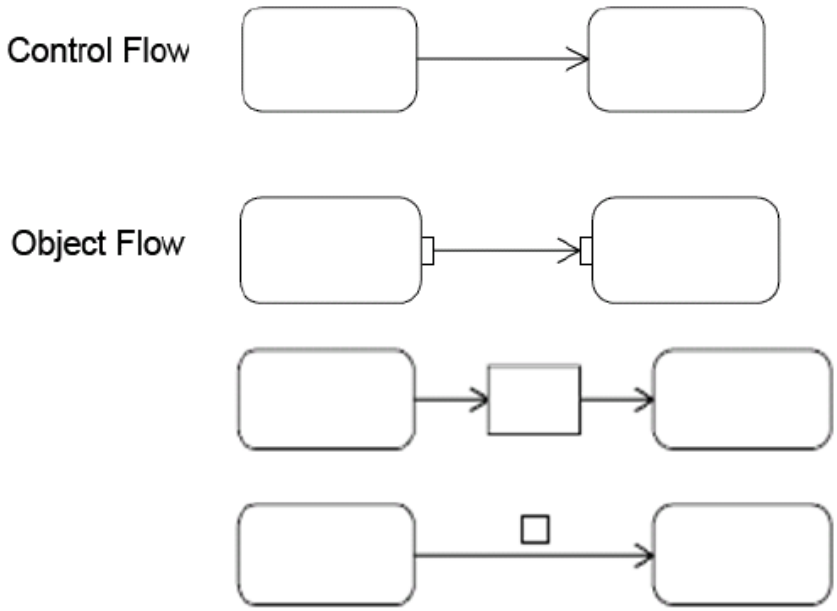


Fig. 7 Tokens Disponibles.

Particiones: Aunque ya existen en versiones anteriores de la especificación, las particiones a partir de la versión de UML 2.0 tienen más expresividad. No tienen semántica de ejecución y nos van a permitir agrupar acciones de acuerdo a una serie de criterios. Esta agrupación puede hacerse en una o en dos dimensiones, tal y como podemos apreciar en la figura 7 en la que las acciones se han agrupado

verticalmente en relación a la responsabilidad y horizontalmente en relación al lugar de ejecución de la acción.

Regiones de Expansión: Las regiones de expansión nos van a permitir realizar una misma acción y/o actividad sobre un conjunto de datos. La actividad o acción se ejecutará una vez por cada uno de los datos de entrada. Esta ejecución puede ser o bien paralela, iterativa o en stream y se debe tener en cuenta que el conjunto de los objetos de entrada tiene que ser igual que el conjunto de los objetos de salida, en el número y en el tipo de cada uno de ellos. Podemos ver un ejemplo de representación de una región de expansión en la Fig. 8.

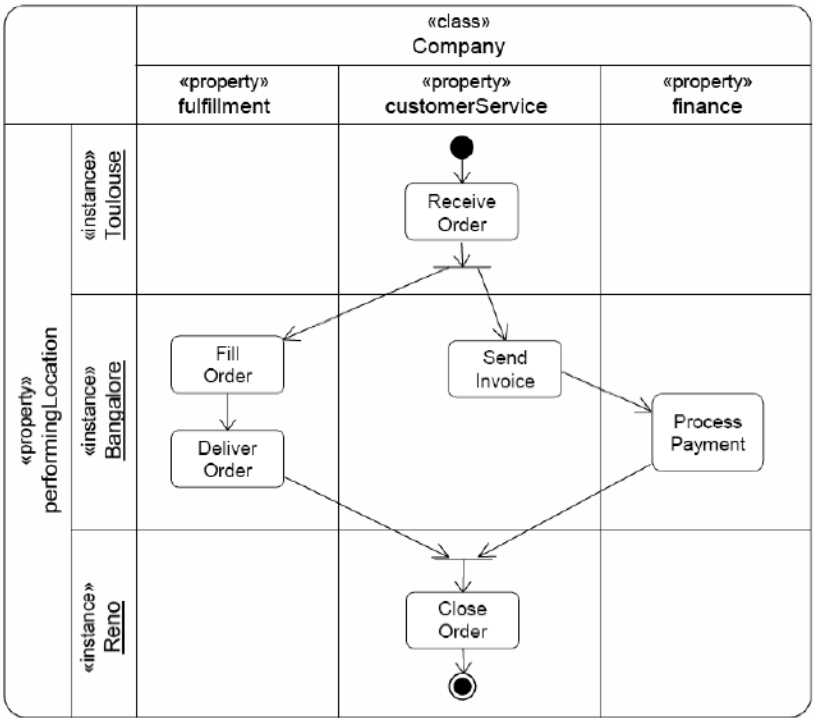


Fig. 8 Ejemplo de Partición Bi-dimensional.

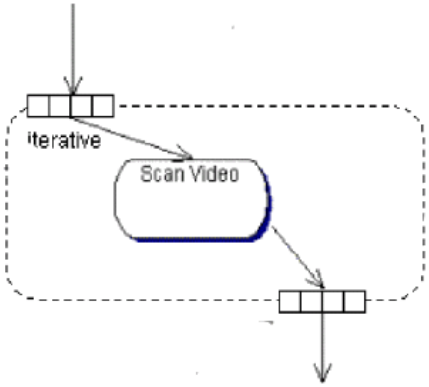


Fig. 9 : Ejemplo de Región de Expansión Iterativa.

Excepciones: Las excepciones en los Diagramas de Actividad sirven para modelar que tipo de acciones hay que llevar a cabo -en caso de que una excepción especificada ocurra durante la ejecución de un proceso protegido, que se interrumpe al llegar la excepción. Podemos ver un ejemplo de dos formas alternativas de describir este tipo de construcciones en la Fig. 9.

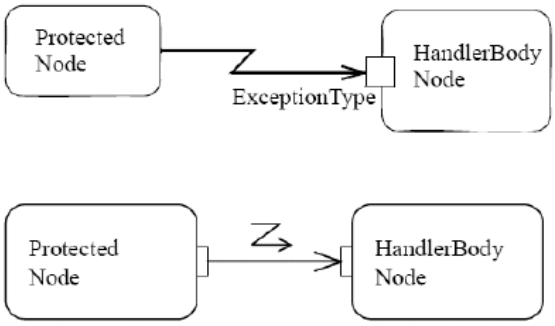


Fig. 10: Ejemplo de Utilizacion de Excepciones.

Regiones de Actividad Interrumpibles*:* Es una región dentro de la actividad que interrumpe todos sus flujos cuando un token atraviesa sus límites (delimitados por línea discontinua) a través de uno de sus flujos de salida. Para denotar una región de actividad interrumpible podemos usar una de las dos maneras especificadas en la Fig. 10.

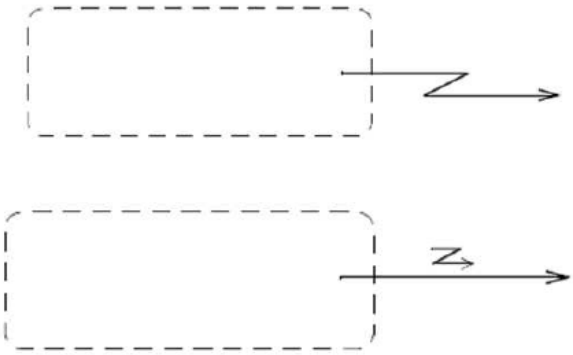


Fig. 11: Regiones de Actividad Interrumpibles.

Streaming: Con los Diagramas de Actividad de UML 2.X también se puede modelar un comportamiento de streaming. Una acción se dice que tiene una ejecución de streaming cuando puede producir su salida mientras procesa sus entradas. Para indicar esto en UML se tiene varias notaciones alternativas que muestra la Fig. 11.

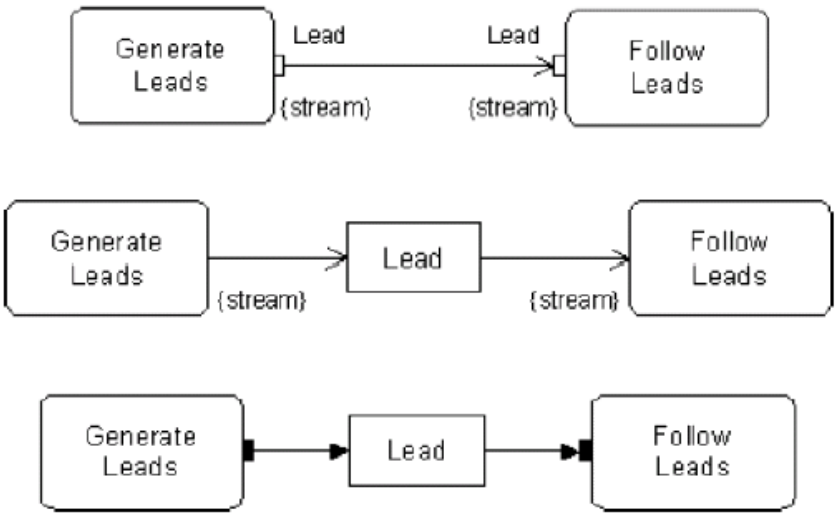


Fig. 12: Ejemplo de Acciones que se Ejecutan en Streaming.

2.3.BPMN:

EL Objet Management Group (OMG) tiene desarrollado un estandar de Business Process Model Notation(BPMN). El principal objetivo de BPMN proporciona una notación que realmente no es entendible por todos los usuarios de negocios, desde los analistas de negocios que crea la parte inicial de los procesos, la técnica de desarrollo resposable para la implementación de la tecnología que realizara estos procesos y finalmente, la gente de negocios quien manejara y controlar estos procesos.

2.3.1. Elementos básicos de los diagramas BPMN:

Los diagramas BPMN, también llamados BPD están formados por una serie de elementos fundamentales. Estos se pueden clasificar en cuatro categorías fundamentales:

1. Objetos de Flujo (Flow objects)
2. Conectores (Connecting Objects)
3. Calles (Swinlanes)
4. Artefactos (Artifacts)

Objetos de flujo (Flow objects).

Tipo	Descripcion	Imagen
Eventos(events)	Algo que ocurre durante el transcurso de un proceso de negocio. Pueden ser de tres tipos, de Inicio, Intermedio y de Finalización	
Actividades(Activity)	El término genérico para denominar cualquier trabajo que realiza la compañía. Pueden ser atómicas o compuestas Los tipos de actividades que son una parte de un modelo de proceso son: subprocesos y tareas, los cuales son rectángulos redondeados.	
Pasarelas (Gateway	Para controlar el flujo, puede ser una decisión tradicional, un join, un merge y un fork.	

Tabla 1: Objetos de Flujo en BPMN.

Conectores: Son los elementos que servirán para conectar los diferentes Flow Objects con el objeto de crear el esqueleto estructural básico del procesos de negocio.

Existen tres tipos de conectores cuyas descripciones y símbolos podemos ver en la Tabla 2.

Tipo	Descripción	Imagen
Flujo de secuencia (Secuence Flow)	Para indicar el orden en el cu´al son ejecutadas las actividades del proceso de negocio	
Flujo de mensaje (Message Flow)	Para mostrar el intercambio de mensajes entre dos participantes (entidades de negocio o roles).	
Asociación (Association)	Para asociar artifacts con flow objects	

Tabla 2: Conectores en BPMN.

Calles (Swinlanes): Las calles o swinlanes son un mecanismo que nos va a permitir clasificar las actividades de manera visual para ilustrar las distintas categorías o responsabilidades. Las distintas clases de este tipo de objetos se pude apreciar en la Tabla 3.

Tipo	Descripción	Imagen
Pool	Para indicar los participantes en el Proceso	
Lane	Es una particionn de POOL, ya sea vertical u horizontal que nos va a permitir clasificar las actividades	

Tabla 3: Objetos Swinlanes en BPMN.

Artifacts (Artefactos o Productos): Existen tres tipo de artifacts predefinidos, aunque para un determinado dominio BPMN permite añadir artifacts adicionales. Los tres tipos predefinidos se pueden apreciar en la Tabla 4.((OMG), 2006)

Tipo	Descripción	Imagen
Data(Data Object)	Para mostrar los datos que son producidos o requeridos por las actividades	

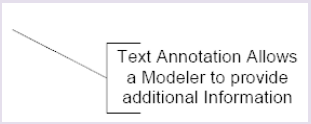
Grupo (Group)	Para agrupar distintos elementos del Diagrama	
Anotaciones (Annotations)	Para proporcionar información adicional.	

Tabla 4: Artefactos en BPMN.

2.3.2. Variaciones de los elementos básicos

En la sección anterior vimos los elementos básicos que componen los diagramas en BPMN. Además de estos elementos básicos existen distintas variaciones de los mismos.

2.3.2.1. Tipos de eventos

Los eventos, tal y como se definieron previamente son algo que ocurre en el transcurso de un proceso de negocio. Además de los tres tipos básicos (Inicio, Intermedio y Final) existen especializaciones de los mismos. Estas especializaciones las podemos ver en la Fig. 13.

Fig. 13: Tipos de Eventos en BPMN.

Message: Al recibir un mensaje de un participante (Inicio, intermedio) o que envía un mensaje a un participante al acabar el proceso.

Timer: Evento que se dispara al llegar un momento previamente determinado.

Error: Al producirse un error (Inicio o intermedio) o que genera un error que debe ser capturado.

Cancel: Evento que se dispara al cancelarse una transacción (Intermedio) o que permite generar una cancelación de una transacción.

Compensation: Para realizar acciones de compensación en caso de que se deba cancelar una actividad o para generar esta actividad de cancelación de una actividad en curso.

Rule: Evento que se dispara cuando se cumple una regla determinada. Va asociado a las excepciones.

Link: Para conectar eventos de distintos tipos.

Multiple: Cuando existen varias formas de que se dispare el evento (Inicio, intermedio) o cuando existen diversas consecuencias al producirse el mismo.

Terminate: Finaliza todas las actividades del proceso.

2.3.2.2. Tipos de Gateway.

Los gateways son los elementos que nos van a permitir realizar el control de flujo dentro de un diagrama BPMN. Además del tipo básico descrito anteriormente existen diversas variaciones. Estas variaciones las podemos ver en la Fig. 14.

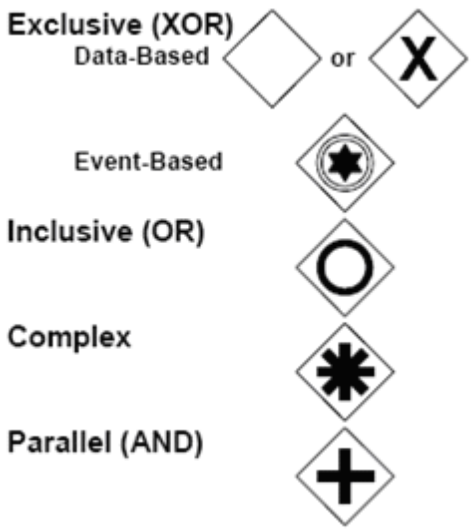


Fig. 14 Tpos de Gateways en BPMN.

Exclusive (Event o Data Based): Para consumir tokens únicamente de una de las ramas de entrada (Exclusive Merge) o para propagar tokens en sólo una de las ramas de salida (Exclusive Decision).

Inclusive: Para consumir tokens de una o m´as ramas de entrada (Inclusive Merge) o para propagar tokens a, al menos, una de las ramas de salida (Inclusive Decisión).

Complex: Para describir Merge/Join o decisiones que requieran condiciones complejas para consumir o producir tokens a través del gateway.

Parallel: Consume todos los tokens de entrada (Parallel Merge) y dispara todos los tokens de salida (Parallel Joining).

2.4. BPMN 2.0.

BPMN 2.0 es extendido hacia un modelo y una notación que incluye un meta modelo (cambia de nombre).

Nuevos modelos y Diagramas: Conversación y Coreografía.

Extensión del modelo de Colaboración: Múltiples participantes y nuevo objeto de mensajes.

Extensión de la tipología de actividades, eventos y gateways: muchos nuevos tipos.

Definición de un metamodlo de intercambio: basado en diagramas de clases de UML.

Reglas para la ejecución de diagramas y mapeo hacia BPEL.

2.4.1. Elementos BPMN 2.0

Tarea (atómica): Una tarea es una actividad atómica que esta incluido entre un proceso.

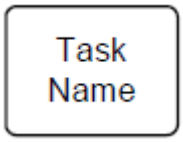


Fig. 15: Tarea

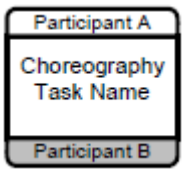


Fig. 16: Tarea Coreografia

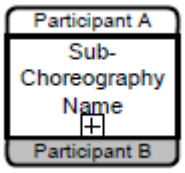


Fig. 17: Tarea Sub-Coreografia

Tarea coreografía: Una tarea de coreografía es una actividad atómica en una coreografía. Este representa un conjunto de uno o más intercambios de mensajes.

Cada tarea de coreografía involucra dos participantes. El nombre de tarea de coreografia y cada participante es visualizada en las diferentes bandas en la parte superior e inferior de la gráfica.

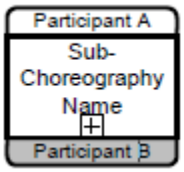


Fig. 18: Collapsed Sub-coreografia

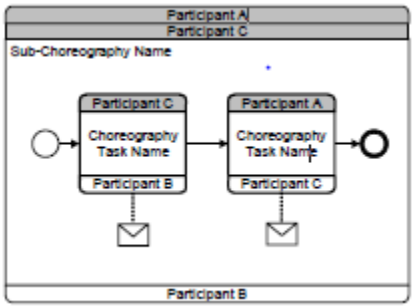


Fig. 19: Expanded Sub-coreografia

Collapsed Sub-coreografia: Los detalles de la sub-coreografía no son visibles en el diagrama. Un signo "plus" en el centro derecho del nombre de la tarea, forma que indica que la actividad es un sub-proceso.

Expanded Sub-coreografia: El límite de sub-coreografía es expandido y los detalles son visibles. Note que la secuencia de flujos común.

Fig. 20: Data Objet

Fig. 21: Data Objet(colección)

Fig.21: Data input Data Ouput

Data Objet: Proporciona informacion acerca de que actividad requiere ser representado y/o que los produce. Data objets pueden representar un objeto singular o una colección de objetos. La entrada de datos y salida de datos proporciona alguna información de los procesos.

2.5. Diagramas de Colaboración:

Diagramas de colaboración BPMN que contiene todos los objetos de los diagramas de procesos y adiciona el flujo de mensaje entre pools (participantes). Por tanto ellos son utilizados para mostrar un alto nivel de interacción entre participantes y el cambio de control de flujo entre pools. Una colaboración representa las interacciones entre dos o más entradas de negocios. Una colaboración usualmente contiene dos o más pools, representando los participantes en la colaboración. El intercambio de mensaje entre los participantes es visto por un flujo de mensaje que conecta dos pools(o los objetos entre pools) los mensajes asociados con los flujos de mensaje pueden también ser vistos.

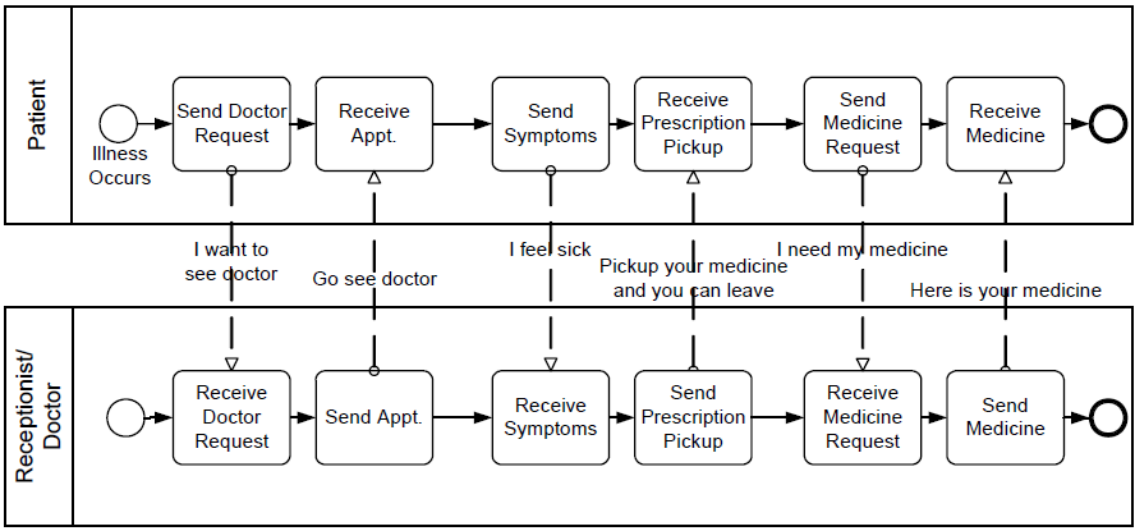


Fig. 22: Ejemplo de Procesos de Colaboración.

2.6. Diagramas de Coreografía.

Se ofrece dos nuevas formas de modelar entre los diferentes participantes en forma mas compacta y expresiva. Se trata de los diagramas de coreografía y de conversación que se concentran en forma exclusiva de representar el intercambio de información entre los participantes independientes. Los diagramas de coreografía pueden ser de gran utilidad en la coordinación de flujos complejos.

En este tipo de diagrama, solo se describen las comunicaciones entre los participantes del proceso ("Quien con Quien y Que"), ocultando todas las actividades internas. La comunicación es descrita mediante un conjunto de intercambios de mensajes los cuales están relacionados lógicamente y están vinculados a través de grupos de enlaces-conversación.

La coreografía representa la interacción entre dos participantes del proceso, distinguiéndose si un participante está iniciando la comunicación (parte activa) o si la está recibiendo (parte pasiva). El participante que inicia se especifica por encima o por debajo de la tarea, la tarea en blanco es quien la inicia y la gris quien recibe.

Participante: El participante que envía información en el cuadro de coreografía de fondo blanco y el participante que recibe la información con un fondo gris, no existe una regla en el objeto de coreografía con respecto al orden arriba o abajo del participante que envía o recibe; pueden proporcionarse como mejor le parezca al modelador.

Tarea coreografía: Una tarea de coreografía es una unidad atómica, que representa un conjunto de uno o más un intercambio de mensajes, cada tarea coreografía consta de dos mas participantes, el nombre de la tarea de la coreografía y cada uno de los participantes se visualizan en las distintas bandas que componen la notación gráfica de la forma:

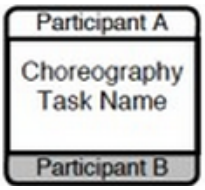


Fig. 23 Tarea de coreografía.

2.6.1.1. Diagrama de Coreografía

La coreografía de un proceso representa las interacciones entre dos o mas entidades de negocios. También representa una secuencia de los tipos de interacción de las actividades.

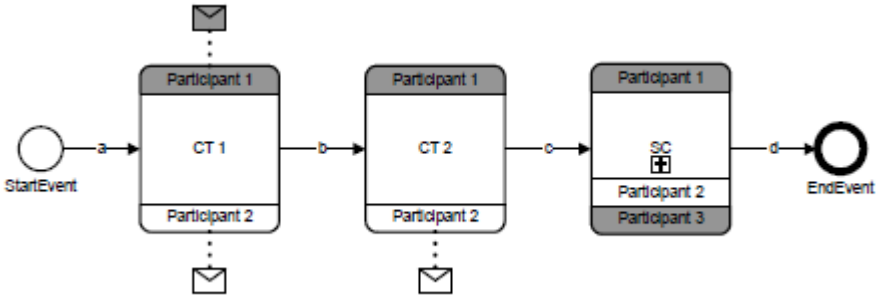


Fig. 24 Diagrama de coreografía.

2.7. Diagrama de Conversación:

Un diagrama de conversación es el uso particular y una descripción informal de un diagrama de colaboración. En general, este es una versión simplificada de colaboración, pero los diagramas de conversación mantienen todas las características de una Colaboracion. En particular procesos pueden aplicar mientras los participantes (Pools) de los diagramas de conversación, para mostrar como Conversacion y actividades relacionadas.

Una conversación es un grupo lógico de intercambio de mensajes. La relación lógica, en la práctica ofrece conocer unos objetos de negocio.

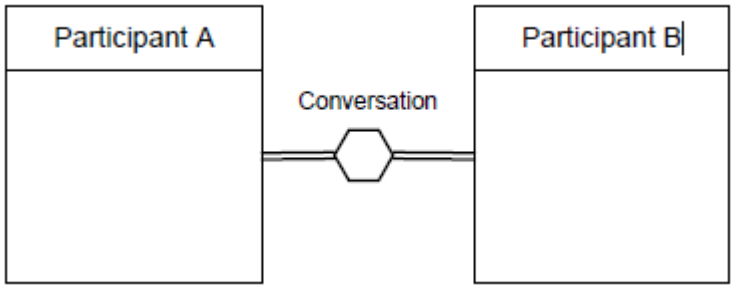


Fig. 25: Diagrama de Conversación.

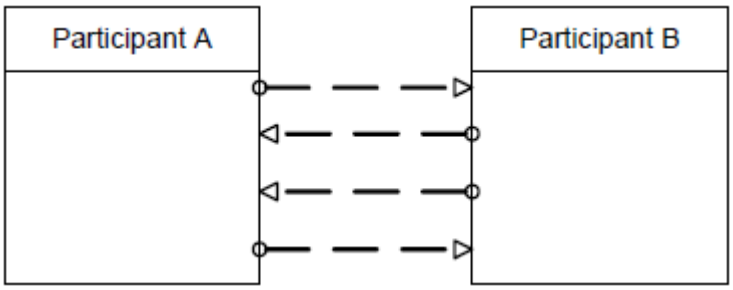


Fig. 26: Diagrama de Conversación donde la conversación es expandido entre flujos de mensaje.

Los requisitos asociados a un proceso están constituidos fundamentalmente por mensajes de entrada y mensajes de salida. Estos mensajes pueden ser formularios de entrada de datos están asociados a la adquisición de información. Son el punto de partida del análisis de comunicaciones.

2.8. Análisis de Comunicaciones.

El Análisis de Comunicaciones es un método de ingeniería de requisitos es que propone afrontar el análisis de SI desde una perspectiva comunicacional. El objetivo es descubrir y describir las interacciones comunicativas entre el SI y su entorno. Para luego especificar los procesos de negocio mediante *Diagramas de Sucesos Comunicativos*, una técnica de modelado gráfico de notación semejante a los Diagramas de Actividad. Las *Plantillas de Especificación de Sucesos* son una técnica de especificación textual que prescribe una estructura para los requisitos asociados a un suceso comunicativo. [España, González et al. 2009].

Se define *interacción comunicativa* como la interacción entre actores con el objetivo de intercambiar información. Dependiendo de la dirección preponderante de la comunicación, se distinguen dos tipos de interacción comunicativa:

Interacción comunicativa entrante: Su objetivo fundamental es aportar al sistema nueva información significativa; es decir, alimenta la memoria del SI con datos relevantes para la organización, previamente desconocidos.

Interacción comunicativa saliente: Su objetivo fundamental es distribuir datos conocidos a los usuarios; es decir, consulta la memoria del sistema para recuperar datos y presentarlos a los usuarios.

Suceso comunicativo: Un suceso comunicativo es un conjunto de acciones relacionadas con la información (adquisición, almacenamiento, proceso, recuperación y/o distribución), que se llevan a cabo de manera completa y sin interrupciones, con motivo de un estímulo externo. Para el Análisis de Comunicaciones, un suceso comunicativo es una interacción comunicativa entrante que cumple ciertos criterios de unidad (guías metodológicas que facilitan la creación de modelos modulares) (España, 2009).

Diagrama de Sucesos: A través de él se define el comportamiento comunicativo que requiere el proceso de negocio.

Cuando en un diagrama de sucesos relacionamos dos sucesos A y B estamos expresando que una instancia u ocurrencia de la clase B necesita la existencia previa de al menos una ocurrencia de la clase A.

Actor: Un actor es un usuario del sistema. Incluye usuarios humanos y otros sistemas computarizados. Los actores de un sistema de información describen sus percepciones de manera diversa.

Disparos o evento: El disparo se representa siempre mediante una entidad externa o actor primario que comunica al sistema algo que ha ocurrido. Un disparo es una relación entre una entidad externa que comunica un mensaje al sistema y la reacción que desencadena.

Fig. 27 El disparo como relación comunicativa.

Mensajes. Los mensajes intercambiados entre los actores y el sistema los representaremos mediante flechas de trazo grueso. A ser posible distinguiremos claramente el color de los mensajes de entrada y de los mensajes de salida.

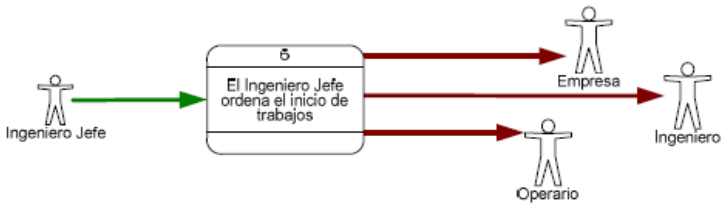


Fig. 28 Mensajes de entrada y salida.

2.9. Estructuras De Mensaje.

Estructuras de Mensaje es una técnica de especificación que permite describir, mediante texto estructurado, el mensaje asociado a una interacción comunicativa.

Constructores gramaticales: La sintaxis de las estructuras de mensaje puede ser escrita en términos de los siguientes constructores gramaticales:

Primitivas
Aggregation A = < a + b + c > A esta compuesto de los archives a y b y c.
Alternative A = [a | b | c] A is either composed of field a or b or c, (only one of them).
Iteration A = { B } A is composed of several substructures of type B.
Identification a(id) Field a identifies an object that is already known by the IS.

Tabla 5: Primitivas

Gramatica EBNF
message structure **= structure name, '=', initial substructure;**
initial substructure **= aggregation substructure | iteration substructure;**
aggregation substructure **= '<', substructure list, '>';**
iteration substructure **= '{', substructure list, '}';**
specialisation substructure **= '[', substructure list,{ '|', substructure list },']';**
substructure list **= substructure, { '+', substructure };**
complex substructure **= aggregation substructure | iteration substructure** **| specialisation substructure;**
Substructure **= substructure name, '=', complex substructure | field;**

Tabla 6: Gramatica EBNF de las Estructuras de Mensajes.

2.9.1. Especificación de campos

Para caracterizar un campo, se pueden especificar las siguientes propiedades.

Nombre. Cada campo debe tener un nombre significativo.

Operación de adquisición. Especifica la procedencia de la información que representa el campo.

Introducción (**i**). La información del campo la provee el actor primario.

Generación (**g**). La información del campo puede ser automáticamente generada por el SI.

Derivación (d). La información del campo puede ser derivada de la memoria del SI porque ya se conoce; es decir, fue comunicada en un suceso comunicativo anterior. La derivación lleva asociada una fórmula de derivación.

Dominio: Especifica el tipo de información que contiene el campo.

Ejemplo*:* Un ejemplo de valor para el campo, aportado por la organización.

Descripción*:* Una explicación que facilite comprender el significado del campo.

Etiqueta: Un texto breve que describe el campo cuando se presenta en un formulario de interfaz gráfica de usuario.

Vinculación con memoria: Describe la correspondencia del campo con una columna de tabla en una base de datos o con un atributo en un diagrama de clases.

Obligatoriedad: Indica si el campo debe necesariamente tomar valor o no. Es posible especificar esto usando una especialización con una sola variante.(Sergio España, 2011)

Inicialización. El valor por defecto asociado a un campo se puede especificar mediante una función o una fórmula de derivación.

Visibilidad. Indica si el campo es visible en un formulario de interfaz.

2.10. Herramientas de Soporte a las Estructuras de Mensaje.

Xtext es un marco de código abierto para el desarrollo de lenguajes de dominio específicos, es parte de la infraestructura Eclipse Modeling. A diferencia de generadores de analizadores sintácticos estándar, XText no sólo genera un analizador sintáctico, sino también un modelo para el árbol de sintaxis abstracta y con todas las funciones, editor de Eclipse personalizable.((MOF)2.0, 2008)

EMF: Es un marco de modelado y las instalaciones de generación de código para las herramientas de construcción y otras aplicaciones basadas en un modelo de datos estructurado. Desde una especificación del modelo se describe en XMI, EMF proporciona herramientas y el soporte de ejecución para producir un conjunto de clases Java para el modelo, un conjunto de clases de adaptadores que permiten la visualización y edición de comandos basados en el modelo, y un editor básico. Los modelos pueden ser especificados usando anotaciones de Java, los documentos XML, o herramientas de modelado como Rational Rose, luego

importados a EMF. Lo más importante de todo, EMF proporciona la base para la interoperabilidad con otros basados en EMF-herramientas y aplicaciones.

EMF extiende la herramienta CASE para Análisis de Comunicaciones y su metamodelo Ecore ofrece la posibilidad de definir transformaciones modelo a modelo mediante lenguajes como ATL Transformation Language (Kurtev, 2005).

2.11. Herramientas de Soporte BPMN2.0

Hoy en día, ya se puede adquirir la versión finalizada del BPMN 2.0 en diferentes editores en este documento notaremos algunos como: Enterprise Architect, Visio y Oryx, en su versión Open Source (Signavio en su versión de pago).

Enterprise Architect: puede ayudarlo a identificar y documentar los procesos dentro de un negocio, e identificar que procesos de negocio se pueden administrar más efectivamente. Además, tienen la importante función de situar sistemas de software nuevo y existente dentro del contexto de negocio. Se ajusta idealmente para capturar y documentar sus modelos del proceso de negocio. Aún mejor, los modelos que crea en EA se pueden luego usar para dirigir los requisitos, casos de uso, análisis y fases del diseño de nuevos proyectos de desarrollo de software, todo con una trazabilidad completa a su BPM original.

La versión 8.0 es una versión principal de la familia de productos IBM®Rational®Software Architect. Rational Software Architect (RSA) se ha reempaquetado para ofrecer un producto de nivel base que pueda ampliarse en la medida de lo necesario con capacidades específicas de dominio. Asimismo, hay una nueva y significativa funcionalidad central que permite ampliar las tecnologías que soporta y mejorar la productividad y la facilidad de uso.(Systems, 2000).

Fig. 29: Enterprice Architec

Visio: El Modelador BPMN 2.0 para Visio es una extensión independiente de Visio para dibujar y modelar procesos de negocio. Se trata de un paquete fácil de usar integral. La BPMN 2.0 Modeler para Visio admite el conjunto completo de la propuesta de los elementos BPMN 2.0 (objetos de flujo, la conexión de los objetos, Swimlanes, artefactos y datos). El modelador BPMN 2.0 para Visio permite a los usuarios para producir diagramas de proceso, diagramas de colaboración, diagramas y esquemas de Coreografía de conversación como se describe en el documento de especificación de OMG. Marcadores de BPMN se pueden seleccionar fácilmente para aprovechar al máximo el poder de representación de BPMN 2.0. BPMN 2.0 atributos gráficos no se pueden establecer, o de aquellos con un fácil de usar "Propiedades personalizadas" Panel disponibles para cada elemento de BPMN.

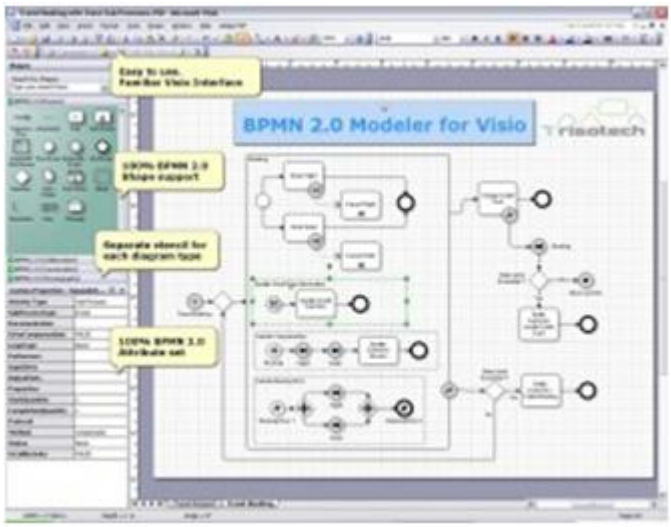


Fig. 30: Visio.

Oryx: Es una plataforma de modelado basada en la web, ampliable, licenciado bajo los términos de código abierto. Ademas de las características:

Soporte completo para BPMN 1.1 y extendida EPC .

Simulación de procesos a paso a través de la semántica.

Transformación de modelos, por ejemplo, inventario de BPMN a BPEL para la composición de servicios web (próximamente).

Exportación de modelos de procesos a PDF, PNG, RDF para su posterior procesamiento, análisis y presentación de informes.

Soporte de idiomas.

Oryx es una plataforma abierta para los acontecimientos relacionados con modelado de procesos de negocio. El proyecto se hospeda en un proyecto de Código de Google bajo licencia MIT. (Gloogle.com, 2000)

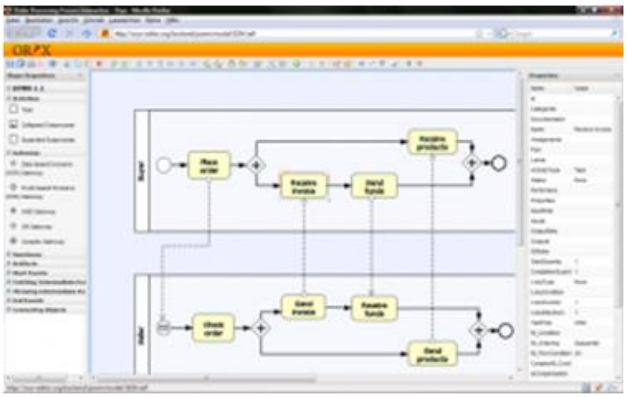


Fig. 31: Oryx.

CAPITULO III

3. Análisis de los Diagramas de Coreografía y Estructuras de Mensaje.

3.1. Actividades en el Análisis de Comunicaciones.

El Análisis de Comunicaciones es un método de ingeniería de requisitos que propone afrontar el análisis de SI desde una perspectiva comunicacional [12]; es decir, la captura, análisis, especificación y verificación de requisitos se centran en las necesidades de comunicación de la organización. El Análisis de Comunicaciones propone una estructura de requisitos que permite una aproximación por refinamientos sucesivos.

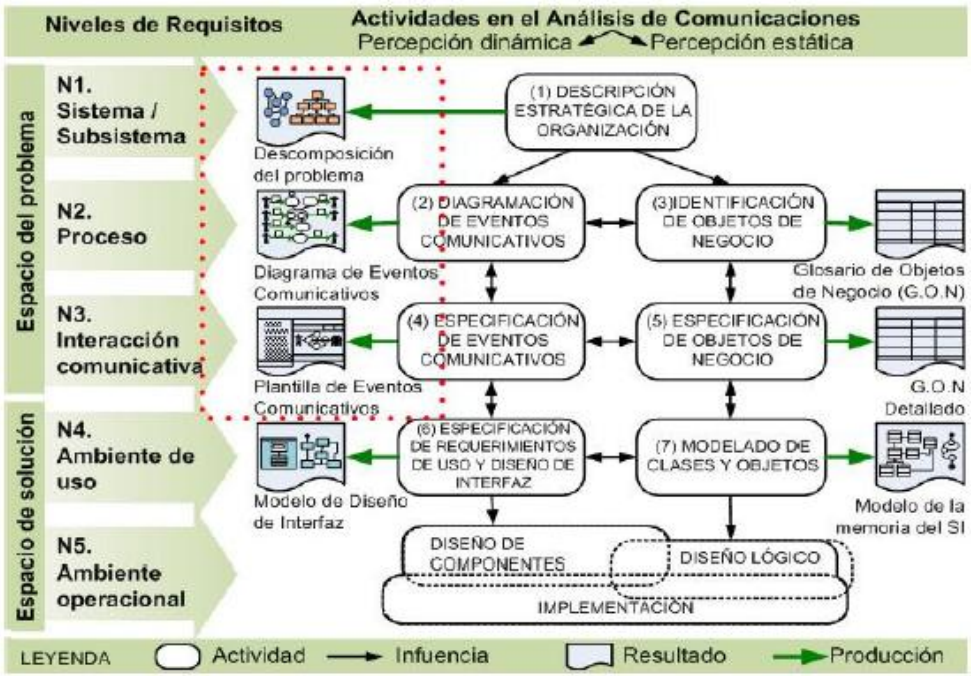


Fig. 32: Estructura básica y flujo de actividades del Analisis de Comunicaciones.

La Fig. 31 muestra los cinco niveles de requisitos propuestos por el Análisis de comunicaciones y las actividades asociadas a cada nivel. Los niveles L1, L2 y L3 corresponden al espacio de problema, ya que no presuponen la informatización de la IS y su objetivo es descubrir y describir las necesidades de comunicación internacionales de los usuarios. Niveles de L4 y L5 corresponden al espacio de solución, ya que especifica cómo las necesidades de comunicación van a ser apoyados. Solo incluye actividades relacionadas con la especificación de

requisitos; aquellas relacionadas con la captura, análisis, verificación y gestión se omiten para simplificar. La justificación de la aproximación y la estructura propuesta se presenta en (Dietz, 1998). En proyectos complejos, el sistema completo se refina en subsistemas (Austin, 1962). Cada proceso se modela mediante un diagrama de sucesos comunicativos (Ballmer, Springer (1981)). Se identifican los principales objetos de negocio (Baniassad, 2004). Cada suceso comunicativo se describe mediante una plantilla de especificación (Brown, 2006). Los objetos de negocio se especifican con más detalle (Bubenko, 1998). A continuación, la interfaz se diseña para dar soporte a la comunicación asociada a los sucesos (Castro, 2002). Se procede al modelado de la memoria del sistema (Chang, 1994). [España, González et al. 2009].

En este trabajo se centra en el espacio del problema, con particularidad en el nivel 2 y nivel 3.

En este nivel 2 de requisitos, análisis Comunicación propone describir los procesos de negocio desde una perspectiva comunicacional. El objetivo es descubrir las interacciones comunicativas entre el IS y su entorno. Se refiere a la interacción comunicativa a una interacción entre los actores con el objetivo de intercambiar información, especificando el flujo de eventos de comunicación por medio del diagrama de evento comunicativo.

En el nivel 3, los eventos comunicativos que aparecen en el Diagrama de Eventos comunicativos, necesitan ser descritos en detalle. Requisitos asociados a un evento, se estructuran a través de una plantilla de especificación de eventos.

Para una mejor comprensión en este documento se tiene el siguiente caso de estudio:

Un gabinete de servicios tiene una sección para realizar proyectos de instalaciones eléctricas industriales. A tal fin el gabinete tiene contratados operarios e ingenieros.

Las empresas contactan con el departamento de atención de clientes que abre una ficha de proyecto asignándole un identificador secuencial, en la que registra los datos de la empresa. A la última hora del día las fichas se depositan en la bandeja

de nuevos proyectos en el despacho del ingeniero jefe. Todas las mañanas, el ingeniero jefe revisa los nuevos proyectos, asignando a cada uno el ingeniero que le parece más conveniente. Para ello revisa las fichas de ingeniero en las que consta el número de proyectos activos en los que están involucrados. Le pasa a su secretario técnico una relación de los números de proyecto y el ingeniero asignado a cada uno.

Su secretario actualiza las fichas de proyecto anotando los datos de cada ingeniero asignado. Y envía a los ingenieros correspondientes una copia de las fichas de proyecto que se les ha asignado. Actualiza también la ficha de cada ingeniero incrementando el número de trabajos activos que constan en ella.

Cuando un ingeniero recibe la información de su asignación a nuevos proyectos se pone en contacto con las empresas correspondientes acordando fecha y hora para ir a visitarlas. En el día acordado el ingeniero asignado visita a la empresa y toma nota de las necesidades del cliente y de las características del local. En función de las necesidades del cliente, elabora un presupuesto que entrega al departamento técnico. En el presupuesto incluye el número de operarios que se necesitarán y el tiempo estimado de los trabajos.

El departamento técnico actualiza la ficha de proyecto correspondiente y envía una copia de este presupuesto a la empresa.

La empresa puede contestar rechazando el presupuesto en cuyo caso se archiva la ficha de proyecto en el fichero de proyectos rechazados. Si la empresa acepta el presupuesto el ingeniero jefe decide la fecha de inicio, según la disponibilidad de recursos, comunicándolo a la empresa y al ingeniero asignado. Al mismo tiempo asigna los operarios necesarios al proyecto comprobando que no estén ocupados en otro proyecto y actualizando sus fichas respectivas así como la ficha de proyecto que se guarda en el archivo de proyectos activos.

Una vez finalizados los trabajos el ingeniero asignado al proyecto se lo comunica al ingeniero jefe que procede a anotarlo en la ficha de proyecto que envía a la sección de contabilidad para que proceda a la gestión de cobro.

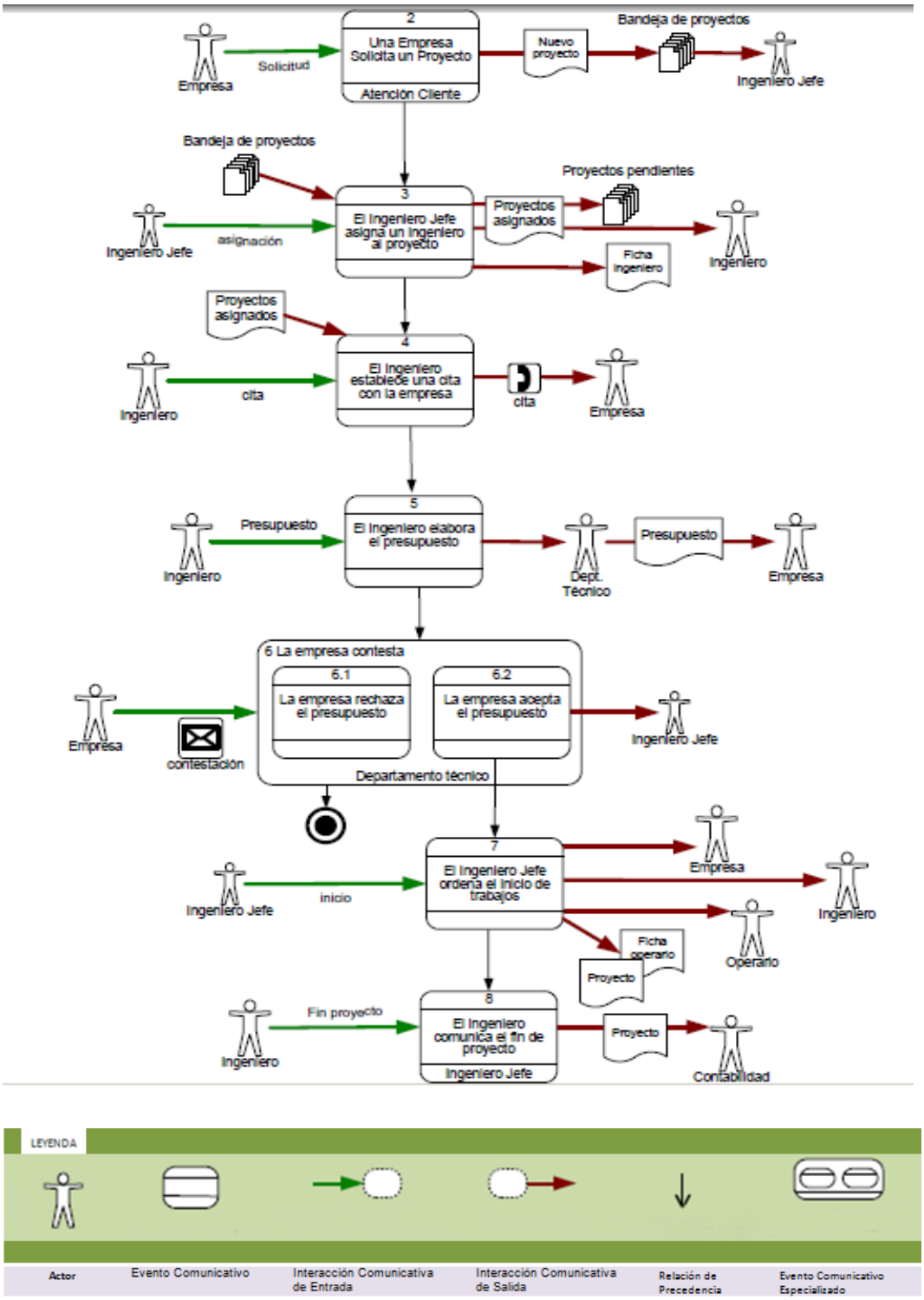


Fig. 33: Diagrama de Sucesos Comunicativos (Gabinete de Proyectos).

Los mensajes asociados a los eventos de comunicación se transmiten a través de entrante comunicativo interacciones salientes y las interacciones comunicativas. En el diagrama de eventos comunicativos, a los mensajes se les da un nombre (mediante el etiquetado de las interacciones comunicativas). Las interacciones de comunicación se modelan como flechas colocadas en el eje horizontal. El eje vertical es reservado para las relaciones de precedencia entre los eventos comunicativos, que son también modelados como flechas. En el caso de estudio un evento o suceso 2 por ejemplo: Las empresas contactan con el departamento de atención de clientes que se abre una ficha de proyecto. Registra en ella los datos de la empresa y le asigna un código secuencial. La ficha se deposita en la bandeja de nuevos proyectos del ingeniero jefe. Todas las mañanas, el ingeniero jefe revisa los nuevos proyectos, asignando a cada uno el ingeniero que le parece más conveniente al tiempo que se los comunica. También lo anota en la ficha del proyecto que guarda en el fichero de proyectos pendientes e incremente el número de incrementa el número de trabajos activos que constan en la ficha del ingeniero.

Respecto a los mensajes el Análisis de Comunicación propone una técnica de especificación de mensajes. Estructura de la Comunicación es una técnica de modelado que se basa en un texto estructurado y permite especificar el mensaje asociado a un evento comunicativo. La estructura de los campos del mensaje se encuentra en posición vertical y otros muchos detalles de los campos pueden ser dispuestas de forma horizontal, por ejemplo, la operación de adquisición de información, el dominio de campo, el vínculo con el objeto de negocio, un valor de ejemplo proporcionado por uno mismo.(España, 2009). Como se muestra en la Fig. 32.

Actualmente esta técnica se puede aplicar en tiempo de análisis, diseño o a lo largo del todo el ciclo de vida. La descripción refleja la estructura de la información asociada a la adquisición de información. Todas las propiedades que comunique deberá asociarlas a los objetos de negocio. Tiene un uso sintáctico y permite reflejar la estructura compositiva de cualquier formulario identificando simplemente las tres formas de composición: secuencia, repetición y alternativa. Estas técnicas se describen con más detalle en trabajos previos (España, 2009).

La estructura que se asocia al evento 2 del caso de estudio en la siguiente tabla:

Una Empresa Solicita un Proyecto

"Las empresas contactan" nos indica el establecimiento de una comunicación. En esa comunicación el emisor, el informador de lo que ocurre, es la empresa. El departamento de atención de clientes actúa como agente de adquisición. Está disponible para adquirir los datos de la empresa.

Deberá considerarse qué información se complementa de la ficha de proyecto. Está claro que los datos del ingeniero asignado no se conocen en el instante que la empresa solicita el proyecto. La estructura de este suceso está caracterizada por las siguientes componentes:

FIELD	OP	DOMAIN	EXAMPLE VALUE
Solicitud Proyecto=			
<Código Proyecto+	g	Tex	19.345.631-D
Fecha Petición+	i	Date	Enero, 10, 2012.
EMPRESA=			
< CIF+	g	Tex	253.45879
Razón social+	i	Tex	
Dirección+	i	Tex	Av. B./ Alonso 29
Persona de contacto+	i	Tex	Carlos Perez.
Teléfono >	i	Tex	9638700021
>			

Fig. 34: Estructura de Mensaje Solicitud Proyecto.

Un evento comunicativo totalmente no puede ser entendido hasta que su estructura de comunicación sea definida en detalle. Este hecho ayudara a los analistas y usuarios. Y así como se asocia dentro de los diagramas de eventos comunicativos, una estructura de comunicación o estructura de mensaje, también se pueden integrar a otros diagramas como es el caso de los diagramas de coreografía (BPMN2.0).

3.2. Analogía de Diagrama de Sucesos Comunicativos y los Diagramas de Corografía (BPMN2.0).

Asi como el Análisis de Comunicaciones es un método de Ingeniería de Requisitos que propone desde una perspectiva comunicacional describir los procesos de negocio. Siendo el objetivo describir y descubrir las interacciones comunicativas entre el Sistema de Información y su entorno. Es decir los procesos de negocio mediante Diagramas de Eventos Comunicativos, una técnica de modelado gráfico de notación semejante a los Diagramas de Actividad.

Tambien al mismo tiempo existen para especificar procesos de negocios los diagramas de coreografía, describen las comunicaciones entre los participantes del proceso. La comunicación es descrita mediante un conjunto de intercambios de mensajes los cuales están relacionados lógicamente representa la interacción entre dos participantes del proceso, distinguiéndose si un participante está iniciando la comunicación (parte activa) o si la está recibiendo (parte pasiva). El participante que inicia se especifica por encima o por debajo de la tarea, la tarea en blanco es quien la inicia y la gris quien recibe.

La siguiente figura muestra el contraste de los diagramas de sucesos comunicativos frente a los diagramas de coreografía del estándar BPMN 2.0.

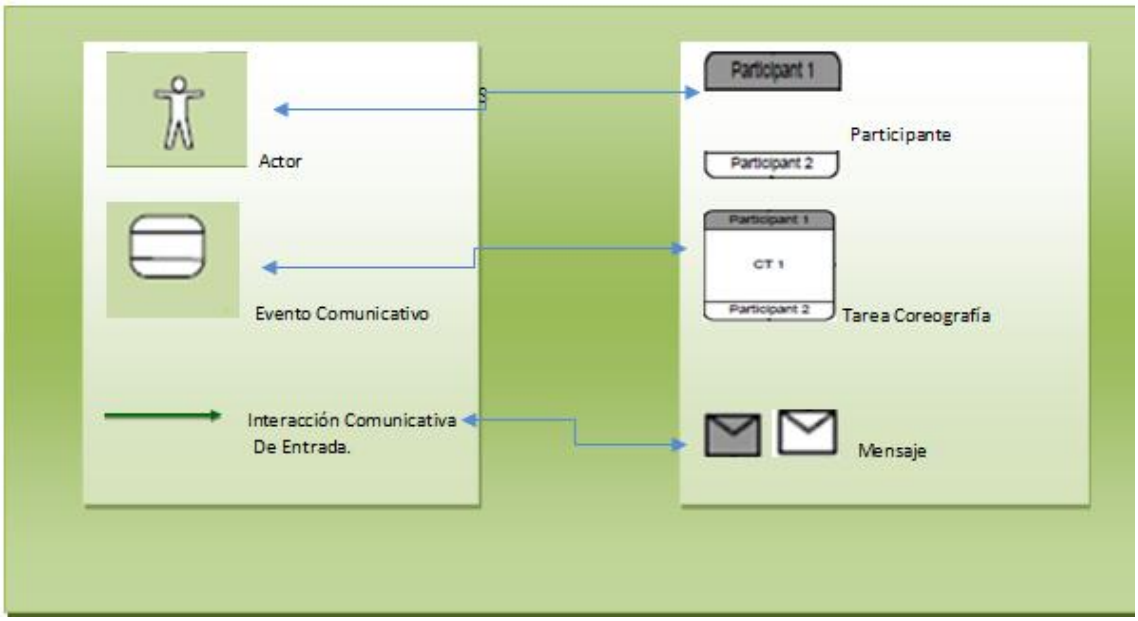


Fig. 35 Analogia Diagrama de Sucesos Comunicativos vs. Diagramas de Coreografia (BPMN2.0).

La Fig. 34 muestra la similitud de poder representar los procesos de negocios con los Diagramas de Eventos Comunicativos, y sus correspondientes en los Diagramas de Coreografía del estándar BPMN2.0.

Cada paso en la coreografía involucra dos o más participantes. En una Coreografía, la comunicación se representa mediante un elemento mensaje entre dos parcipantes. Un mensaje es un rectángulo con líneas diagonales convergentes en la mitad superior del rectángulo para dar la apariencia de un sobre (ver Figura 34). Tiene que estar dibujados con una línea delgada sola. Además, cuando se utiliza en un diagrama de Coreografía más de un mensaje puede ser utilizado para una sola de tarea de coreografia.

Para más claridad de los diagramas de coreografía se ilustra el caso de estudio de Gabinete de Proyectos de la Fig. 35 ahora representado con los diagramas de coreografia:

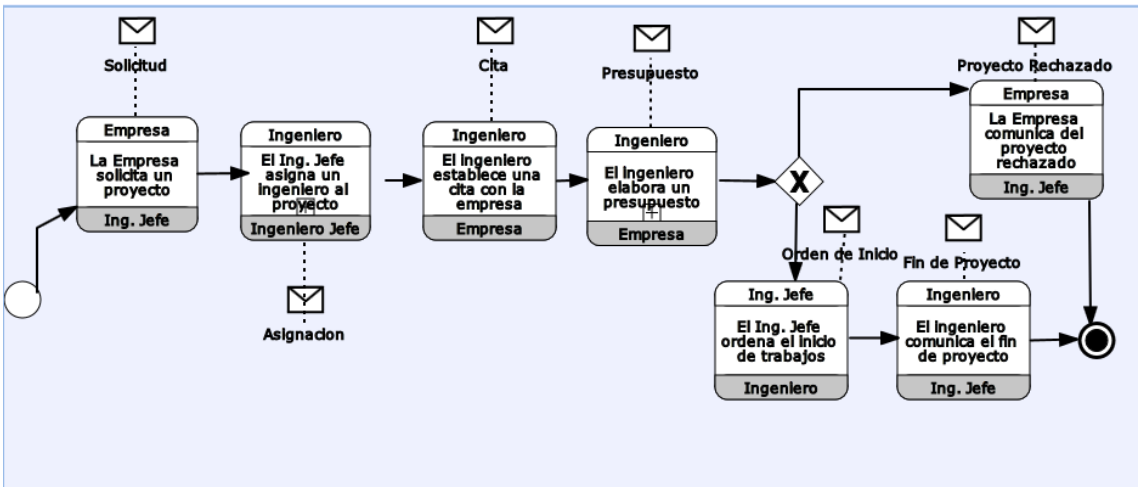


Fig. 36: Diagrama de Coreogratre ellos Text Anotation que pertenece al conjunto de Artefactos.

Text Anotation es un objeto en el cual se puede llenar datos en lenguaje natural, es decir acepta un texto común introducido por el usuario, permite cualquier tipo de digito.

De otra manera en la parte derecha un conjunto de propiedades asociado a cada uno de los elementos; para nuestro objetivo se toma en cuenta el elemento message de Data Objet dentro de este existe la propiedad Documentation con su respectiva representacion que permite al usuario digitar cualquier tipo de texto.

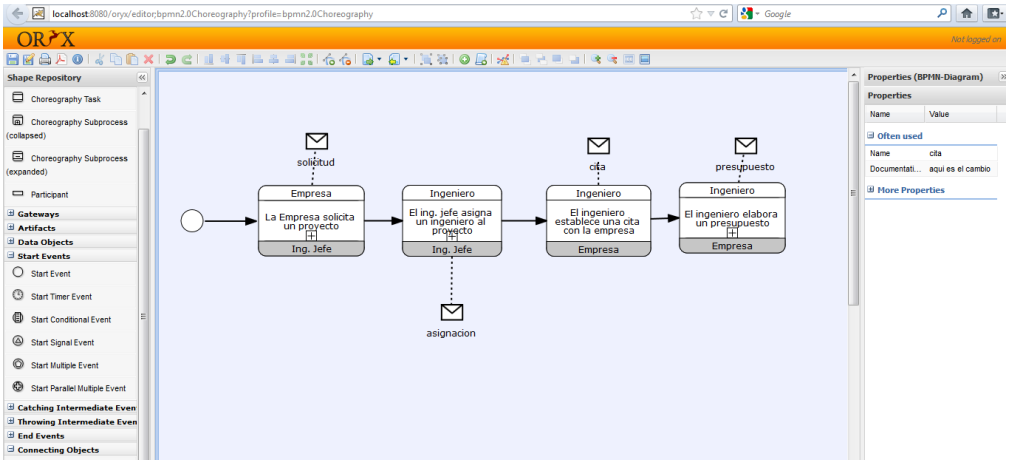


Fig. 37: Editor Oryx.

En cuanto al código referente a Oryx, propiamente los diagramas de coreografia y más específicamente para su objeto message, se tiene una función que escucha el evento generado al pulsar un doble click en un objeto message y el usuario puede digitar el nombre que asignara a dicho objeto.

Por tanto se observa actualmente que Oryx a pesar de ser una de las herramientas más completas y bastaste representativa, requiere de una extensión para dar soporte a las Estructuras de Mensaje.

4. Extension de BPMN 2.0

4.1. Integracion de Estructuras de Mensaje en Diagramas de Coreografia BPMN2.0

En esta parte del documento, se describe el proceso para integrar estructura de mensaje en los Diagramas de Coreografia, se elige el elemento mensaje. Una estrategia será hacer un doble click en el objeto de asociar al objeto mensaje de una ventana (formulario) que soporte la gramática para la estructura de mensaje, del tal manera que pueda el usuario visualizar y llenar dicho formulario con la posibilidad de editar, guardar e imprimir el mismo, figura .Consiguiendo de esta manera una representación completa de los procesos de negocio en BPMN 2.0.

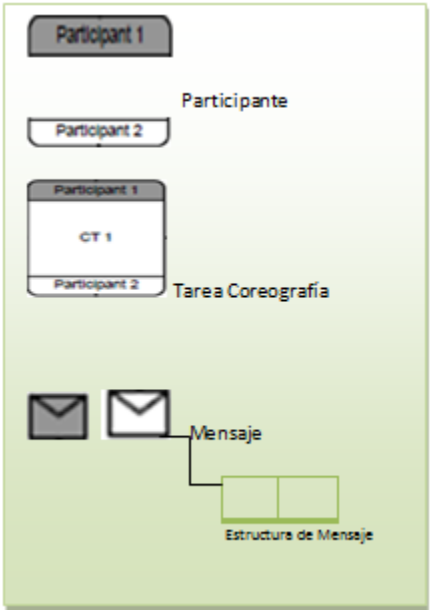


Fig. 38: Estructuras de Mensaje asociado a objeto Mensaje (BPMN2.0)

La siguiente figura ilustra el primer prototipo de la estructura de mensaje ral caso de estudio, en editor Oryx.

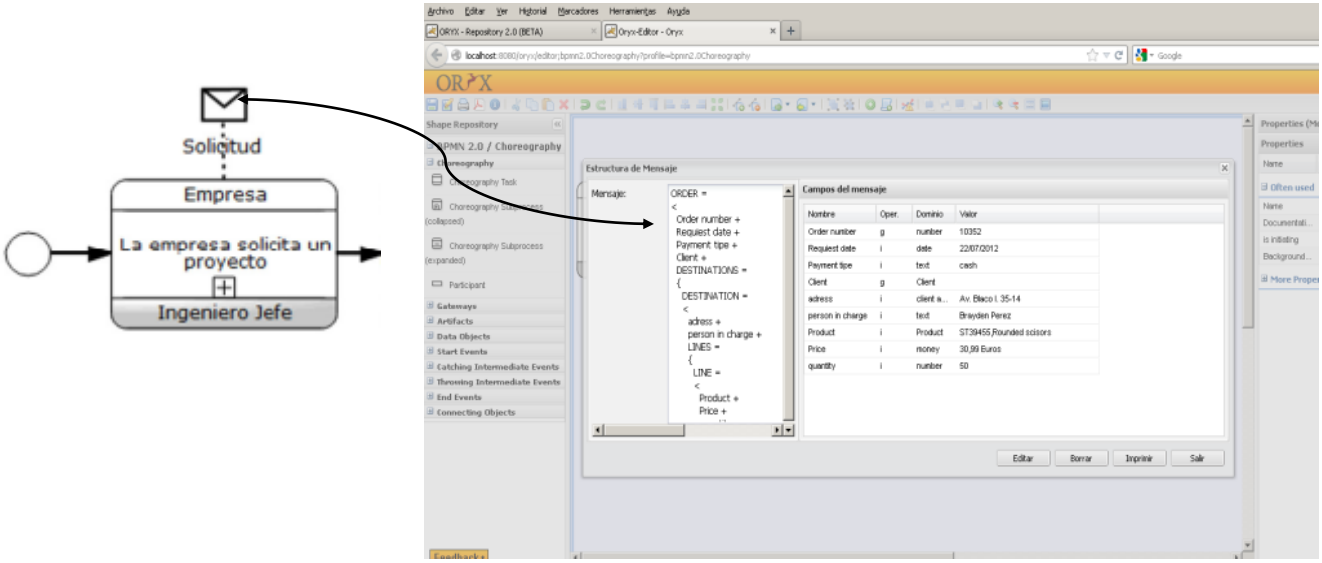


Fig. 39: Prototipo 1 en Oryx

4.2. Extensión de la Herramienta Oryx.

Oryx, es un marco de modelo extensible en el que hace uso de las tecnologías Web 2.0., soporte para múltiples idiomas Modelado toma un papel central en las diversas disciplinas de las ciencias de la computación, incluyendo el diseño del modelo de desarrollo impulsado por la base de datos, arquitecturas de software y gestión de procesos de negocio.

Los modelos son dominios específicos, abstracciones utilizadas para la documentación y el intercambio de ideas, decisiones y pautas de funcionamiento, sino también como modelo para el diseño y desarrollo del sistema. Incluso dentro de un dominio individual hay una amplia gama de notaciones en uso, es decir no hay metamodelo común, son notaciones de dominios específicos.

Meta-información y extensiones de funciones: Se dedica mucho esfuerzo en el uso de modelos como constructor de nuevos sistemas.

Arquitectura Oryx:

La versión actual requiere de un Oryx backend específico, en teoría, cualquier lugar en la Web va a hacer. Oryx es sólo un conjunto de rutinas de Javascript cargada en un navegador web. Los modelos están representados en formato RDF. El núcleo Oryx proporciona un manejo genérico de nodos y aristas, la forma de crear, leer, y actualizarlos, así como una infraestructura para el conjunto de plantillas y plugins.

Soporte de idiomas a través de conjuntos de conjuntos de plantillas conducir el núcleo de Oryx, ya que ofrece tipos explícitos, las reglas de conexión, apariencia visual, y otras características que diferencia un editor de modelo a partir de una herramienta de dibujo genérico.

Por lo tanto, la primera estructura es el modelo Oryx almacenado, directamente sobre la base de los conjuntos de plantilla, segundo el diagrama visual.

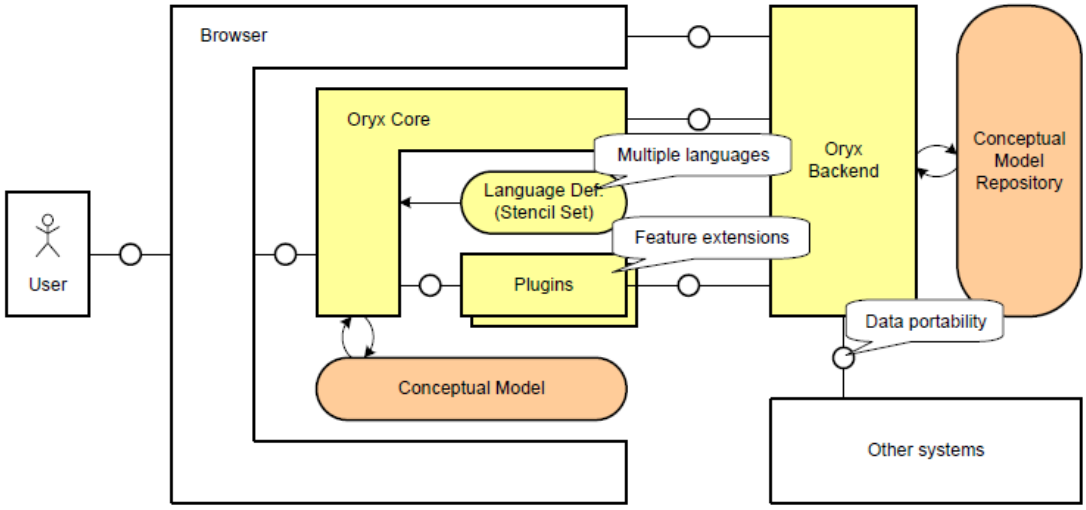


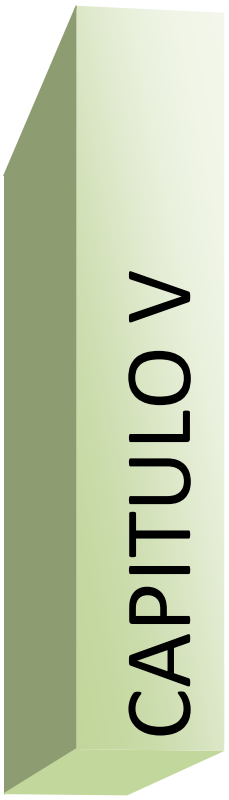
Fig. 40: Arquitectura Oryx

Las extensiones de funciones a través de Plugins: Los plugins permiten tanto genéricos, así como la notaciónextensiones específicas.

Los datos de portabilidad de Oryx: Con la ayuda de los conjuntos de la plantillas y plugins, permite a los usuarios crear, editar y ver los modelos visuales dentro. Sin embargo, cualquier elemento del modelo creado por Oryx es direccionable a través de una URI.

Oryx para BPMN: Un fuerte enfoque inicial fue puesto en el apoyo a Business Process Modeling Notation (BPMN). BPMN se adapte bien como un ejemplo por varias razones: Primero, es una notación importante en el proceso de negocio de la comunidad que los autores pertenecen. En segundo lugar, BPMN está diseñado para ser fácil de entender, incluso por personas fuera del dominio de modelado de procesos de negocio.

BPMN requiere muchas funciones más allá de una simple herramienta de dibujo, por ejemplo, escribir elemento, reglas de contención, y el tipo específico Layouting.



5. Implementación.

5.1. Herramientas a ser utilizadas en la Implementación.

JavaScript: Es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas.

Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.

SVG (Scalable Vector Graphics): define un lenguaje basado en XML para la construcción de gráficos vectoriales 2D con multitud de efectos y características avanzadas. SVG define la representación de gráficos dentro de cualquier documento.

Ext JS: Es una biblioteca de Java Script para el desarrollo de aplicaciones web interactivas usando tecnologías como AJAX, DHTML y DOM. Fue desarrollada por Sencha.

Originalmente construida como una extensión de la biblioteca YUI por Jack Slocum, en la actualidad puede usarse como extensión para la biblioteca jQuery y Prototype. Desde la versión 1.1 puede ejecutarse como una aplicación independiente. Para la implementación de oryx se utilizo Ext JS 2.2. Los mensaje, ventanas, grid, feed viewer, escritorios web siguientes realizadas con Ext JS.

Toncat: Apache Tomcat es una implementación de código abierto del software de Java Servlet y JavaServer Pages tecnologías.

Firefox Browser: Este navegador de Internet corresponde al proyecto, con interfaz gráfica, desarrollado por la corporación Mozilla.

Postgresql: PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales.

PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.

Python: Es un lenguaje de programación multiparadigma. Soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional.Es un lenguaje interpretado, usa tipado dinámico, es fuertemente tipado y es multiplataforma.

Proceso de instalación Editor Oryx.

Un aspecto importante para el desarrollo de un objetivo como el que se plantea en esta tesina, es tener el entorno de programación sea de la manera correcta, el orden del instalación requiere seguir la figura siguiente, con mas detalle en apéndice 1.

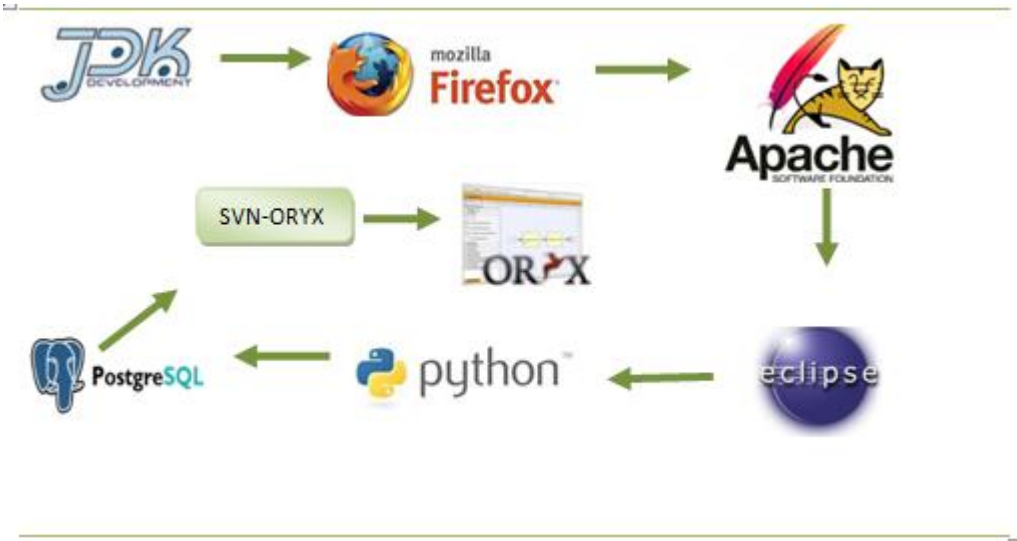


Fig. 41 Proceso de Instalación Oryx

5.2. Intervension y Modificación del Editor Oryx.

Para la implementación de la integración de las estructuras de mensaje en el entorno de los diagramas de coreografia (BPMN2.0) del Editor Oryx.

Se extendió el código interviniendo los siguientes archivos de repositorio que son "renameShapes.js" y "edit.js", ubicados en la dirección: oryx\editor\client\scripts\plugins\. A continuación se tiene los procesos mas importantes que se integra:

- FormPanel para texarea para edición.
- FormPanel para Grid editable.
- Definicion de window.
- Crear window de consulta
- Validacion de la estructura de mensaje.
- Funcion execute para borrar objetos oryx.

A continuación se detalla el código integrado para la extensión de los diagramas de coreografia (BPMN 2.0) en Oryx.

- **Adicion de Formulario al objeto message**.

Ahora se pregunta, si el objeto clickeado es un mensaje y en caso afirmativo ejecutamos la función "formularioMsg (evt, shape)" pasándole los parámetros con el objeto mensaje (shape), en caso de pulsar la tecla <esc> en el formulario ejecutamos el código original llamando a una nueva función (actOnDBLClickSin). El Nuevo código que sustituye a la función "actOnDBLClick" es el siguiente:

```
actOnDBLClick: function actOnDBLClick(evt, shape){
    if( !(shape instanceof ORYX.Core.Shape) ){ return }

    if(shape.getStencil().id()=="http://b3mn.org/stencilset/bpmn2.0#Message"){
        this.formularioMsg(evt, shape);

    } else {
        // Destroys the old input, if there is one
        this.destroy();

        var props = this.getEditableProperties(shape);

        // Get all ref ids
        var allRefToViews   = props.collect(function(prop){ return prop.refToView()
        // Get all labels from the shape with the ref ids
        var labels          = shape.getLabels().findAll(function(label){ return allR

        // If there are no referenced labels --> return
        if( labels.length == 0 ){ return }

        // Define the nearest label
        var nearestLabel    = labels.length == 1 ? labels[0] : null;
        if( !nearestLabel ){
            nearestLabel = labels.find(function(label){ return label.node == evt.tar
            if( !nearestLabel ){
                var evtCoord    = this.facade.eventCoordinates(evt);

                var trans       = this.facade.getCanvas().rootNode.lastChild.getScre
                evtCoord.x      *= trans.a;
                evtCoord.y      *= trans.d;
                if (!shape instanceof ORYX.Core.Node) {

                    var diff = labels.collect(function(label){

                                var center  = this.getCenterPosition( label.node );
                                var len     = Math.sqrt( Math.pow(center.x - evtCoor
                                return {diff: len, label: label}
```

```
                            }.bind(this));

                        diff.sort(function(a, b){ return a.diff > b.diff })

                        nearestLabel =  diff[0].label;
                    } else {

                        var diff = labels.collect(function(label){

                                    var center  = this.getDifferenceCenterForNode( label
                                    var len     = Math.sqrt( Math.pow(center.x - evtCoor
                                    return {diff: len, label: label}
                                }.bind(this));

                        diff.sort(function(a, b){ return a.diff > b.diff })

                        nearestLabel =  diff[0].label;
                    }
                }
            }

            // Get the particular property for the label
            var prop = this.getPropertyForLabel(props, shape, nearestLabel);

            this.showTextField(shape, prop, nearestLabel);
        }
    },

    // Maritza
    actOnDBLClickSin: function actOnDBLClickSin(evt, shape){
        if( !(shape instanceof ORYX.Core.Shape) ){ return }

        // Destroys the old input, if there is one
        this.destroy();

        var props = this.getEditableProperties(shape);


                            }.bind(this));

                    diff.sort(function(a, b){ return a.diff > b.diff })

                    nearestLabel =  diff[0].label;
                }
            }
        }

        // Get the particular property for the label
        var prop = this.getPropertyForLabel(props, shape, nearestLabel);

        this.showTextField(shape, prop, nearestLabel);
    },
```

Siempre que se pulse doble click en un mensaje oryx se ejecutará la función "formularioMsg" que hace lo siguiente:

En primer lugar se mira si el mensaje oryx tiene algún texto tecleado en su textarea (lógicamente la primera vez que se hace doble click no tiene texto asociado), si se ha tecleado algún texto lo recuperamos y lo guardamos en la variable "textosal" que por defecto tiene el valor "Estructura de Mensaje" (el texto tecleado en un mensaje oryx se guarda en una etiqueta <tspan> distinta por cada línea del mensaje, todas ellas hijas de la etiqueta <text> que contiene todo el mensaje oryx y cuyo id es el id del objeto mensaje mas el literal "text_name".

```
var sw_export = 0;
var import_obj="";
var my_text = d.id + "text_name";
var hhhtext = Ext.get(my_text);
var textosal = "Estructura de Mensaje";

if (hhhtext!=null) {
    var textoobj = document.getElementById(my_text).getElementsByTagName("tspan");
    if (textoobj!=null) {
        if (textoobj.length>0) {
            var textosalida = "";
            for (var i = 0;i < textoobj.length; i++) {
                var textosal = textoobj[i].firstChild.data + " ";
                if (textosal.length>=19) {
                    var mensaj=textosal.substring(0, 18);
                    var otrms = "message_estructure";
                    if (mensaj==otrms) {
                        sw_export = 1;
                        import_obj = textosal.substring(19);
                        import_obj = this.trim(import_obj);
                    } else {
                        textosalida = textosalida + textosal;
                    }
                } else {
                    textosalida = textosalida + textosal;
                }
            }
            if (this.trim(textosalida)!="") {
                textosalida = textosalida.replace(/</gi, "&lt;");
                textosalida = textosalida.replace(/>/gi, "&gt;");
                textosal = this.trim(textosalida);
            }
        }
    }
}
```

Este objeto Window de edición se compone de dos objetos, un panel (formPanels) que contiene un campo tipo textArea en el que se tecleará la estructura del mensaje y de un botón para validar el contenido de dicho mensaje, al pulsar este

botón se llama a la función "`verTexto(mensaje)`" pasándole la estructura de mensaje tecleada para que sea validada y formateada si no tiene errores. El código que define este panel es:

```
var formPanels =  new Ext.FormPanel({
    region      : 'center',
    height      : 275,
    width       : 400,
    autoScroll  : true,
    id          : 'formpanel',
    defaultType : 'field',
    frame       : true,
    items       : [
       {
           xtype: 'textarea',
           id:'msgid',
           fieldLabel:'Mensaje',
           width:390,
           height:290,
           allowBlank:false,
           blankText:"Teclee el mensaje y pulsa Ok para validar",
           maskRe:/^([a-zA-Z<>=+_\[\]\{\}\- |().:º,áéíóúñüÑ\d]+ ?[a-zA-Z<>=+_\[\]\{\}
       }
    ]
});
```

Este código añade el botón "Validar" al panel que al pulsarse llama a la función "verTexto" que valida y formatea la estructura de mensaje:

```
var submit = formPanels.addButton({
    text: 'Validar',
    disabled:true,
    handler: function(){
            var sw_error = 0;
            if (formPanels.getForm().isValid()) {
                var mensaje = Ext.getCmp('msgid').getValue();
                var tex = this.verTexto(mensaje);
                tex = tex.replace(/</gi, "&lt;");
                tex = tex.replace(/>/gi, "&gt;");
                if (tex.length>1) {
                    if (tex.substring(0, 2)!="OK") {
                        sw_error = 1;
                        Ext.Msg.show({
                           title:"Aviso",
                           msg: tex,
                           buttons: Ext.Msg.OK,
                           minWidth: 200,
                           fn: function(button){
                              Ext.getCmp('msgid').focus();
                              this.setCaretTo(Ext.getCmp('msgid').getId(), pos_error);
                           }.bind(this)
                        });
                    }
                }

                if (sw_error==0) {
                    // Borro el contenido del grid
                    store.removeAll();
                    grid.stopEditing();
                    for (var i = (nombres.length - 1);i >= 0;i--) {
                        var p = new Plant({
                            nombre: nombres[i],
                            oper: 'i',
                            dominio: 'Texto',
                            comentario: '',

                              valor: ''
                          });
                          store.insert(0, p);
                     }
                     grid.startEditing(0, 0);
                     grid.render();
                  }
             } else {
                    Ext.Msg.alert(ORYX.I18N.MessageForm.error)
                 Ext.Msg.show({
                    title:"Aviso",
                    msg: "Debe teclear algo para validar",
                    buttons: Ext.Msg.OK,
                    minWidth: 200
                 });
                     formularioMsg(h,d)
             }
    }.bind(this)
});

submit.enable();
```

Donde los objetos “store” y “cm” contienen información sobre los datos que contiene el panel y la forma de editarlos. Uno de esos campos, el dominio, contiene un combobox con los tipos de datos más usados en los campos normales de la estructura de mensaje y además se buscan todas las estructuras de mensaje en la página de coreografía (menos la tratada) y se incluyen sus nombres en el combo para que puedan ser usados como campos de referencia. Todo esto se define así:

```
var storeprot = new Ext.data.SimpleStore({
    fields: [
             {name: 'nombre', type: 'string'},
             {name: 'oper', type: 'string'},
             {name: 'dominio', type: 'string'},
             {name: 'comentario', type: 'string'},
             {name: 'valor', type: 'string'}
    ]
});

var Opera = Ext.data.Record.create([
             {name: 'idop', type: 'string'},
             {name: 'description', type: 'string'}
     ]);

var storeOp = new Ext.data.SimpleStore({
    fields: ['idop', 'description'],
});
var op = new Opera({
    idop: 'i',
    description: 'Introduccion'
});
storeOp.insert(0, op);
op = new Opera({
    idop: 'g',
    description: 'Generacion'
});
storeOp.insert(0, op);
op = new Opera({
    idop: 'd',
    description: 'Derivacion'
});
storeOp.insert(0, op);

var comboGenre = new Ext.form.ComboBox({
    triggerAction : 'all',
```

```
        displayField : 'description',
        valueField : 'idop',
        store : storeOp
    });
    var Domino = Ext.data.Record.create([
                {name: 'iddom', type: 'string'},
                {name: 'descripdom', type: 'string'}
           ]);

    var storeDom = new Ext.data.SimpleStore({
        fields: ['iddom', 'descripdom'],
    });
    var dom = new Domino({
        iddom: 'Otra referencia',
        descripdom: 'Otra referencia'
    });
    storeDom.insert(0, dom);

    // Se obtienen todas las estructuras de mensaje definidas en la página
    // y se pasan al combo de dominio (menos la tratada)
    var divobj = document.getElementsByTagName("div");
    if (divobj!=null) {
        if (divobj.length>0) {
            for (var i = 0;i < divobj.length; i++) {
                var divosal = divobj[i].id;
                if (divosal.indexOf("message_estructure")>0 && divosal!=my_div) {
                    divosal = divobj[i].firstChild.data;
                    divosal = this.decodeFromHex(divosal);
                    if (divosal.indexOf("=")>0) {
                        var nombreref = divosal.substring(0, divosal.indexOf('='));
                        dom = new Domino({
                            iddom: nombreref,
                            descripdom: nombreref
                        });
                        storeDom.insert(0, dom);

                    }
```

```
var cm = new Ext.grid.ColumnModel([
     {
        id:'nombre',
        header: "Nombre",
        dataIndex: 'nombre',
        width: 100
    },{
                id:'oper',
        header: "Oper.",
        dataIndex: 'oper',
        width: 30,
        editor: new Ext.form.ComboBox({
            allowBlank: false,
            triggerAction: 'all',
                store:storeOp,
            displayField: 'description',
            valueField: 'idop',
            mode: 'local'
         })

    },{
         header: "Dominio",
         dataIndex: 'dominio',
         width: 100,
         editor: new Ext.form.ComboBox({
             allowBlank: false,
             triggerAction: 'all',
             store:storeDom,
             displayField: 'descripdom',
             valueField: 'iddom',
             mode: 'local'
         })
    },{
         id:'comentario',
         header: "Comentario",
         dataIndex: 'comentario',
         width: 150,
```

```
                }
            }
        }
    }

    dom = new Domino({
        iddom: 'Otro campo',
        descripdom: 'Otro campo'
    });
    storeDom.insert(0, dom);
    dom = new Domino({
        iddom: 'Number',
        descripdom: 'Number'
    });
    storeDom.insert(0, dom);
    dom = new Domino({
        iddom: 'Date',
        descripdom: 'Date'
    });
    storeDom.insert(0, dom);
    dom = new Domino({
        iddom: 'Money',
        descripdom: 'Money'
    });
    storeDom.insert(0, dom);
    dom = new Domino({
        iddom: 'Texto',
        descripdom: 'Texto'
    });
    storeDom.insert(0, dom);

    var comboDomin = new Ext.form.ComboBox({
        triggerAction : 'all',
        displayField : 'descripdom',
        valueField : 'iddom',
        store : storeDom
    });
          editor: new tm.TextField({
              allowBlank: false
          })
      },{
          id:'valor',
          header: "Valor",
          dataIndex: 'valor',
          width: 200,
          editor: new fm.TextField({
              allowBlank: false
          })
      }
]);

// by default columns are sortable
cm.defaultSortable = true;
```

La función que se usa para validar la estructura del mensaje tecleada en el formulario de edición es la función “verTexto(mensaje)”. Esta función valida la sintaxis de la estructura y si es correcta extrae los campos (y los pasa al panel de edición) y formatea toda la estructura (y la guarda en el textarea del panel editable). Si todo es correcto devuelve un mensaje que comienza con “OK”, en caso contrario muestra el mensaje de error producido y no extrae los campos ni formatea el mensaje. El código de esta función es:

```
verTexto:function verTexto(mensaje){
    var msg_sal = "";
    // Sustituyo los saltos de linea por blancos
    // exceptuando los que se consideran separadores de campos
    var msg_sin = mensaje;
    // Sustituyo los tabuladores por blancos
    msg_sin = msg_sin.replace(/\t/gi, " ");

    var sw_campo_nb = 0;
    var sw_salto_linea = 0;
    var ult_plus = -1;
    var ult_char_campo = -1;
    for (var i = 0;i < msg_sin.length; i++) {
        if (this.esCampo(msg_sin.charAt(i))) {
            if (msg_sin.charAt(i)!=' ') {
                sw_campo_nb = 1;
                sw_salto_linea = 0;
                ult_char_campo = i;
            }
        } else {
            if (msg_sin.charAt(i)!='\n') {
                if (ult_plus>=0 && ult_char_campo>=0 && ult_char_campo<ult_plus) {
                    msg_sin[ult_plus] = ' ';
                    if (msg_sin.charAt(i)=='<' || msg_sin.charAt(i)=='{' || msg_sin
                        msg_sin = msg_sin.substr(0, ult_plus) + "=" + msg_sin.subst
                    } else {
                        msg_sin = msg_sin.substr(0, ult_plus) + " " + msg_sin.subst
                    }
                }
                sw_salto_linea = 0;
                sw_campo_nb = 0;
                ult_plus = -1;
                ult_char_campo = -1;
            }
        }
        if (msg_sin.charAt(i)=='\n') {
            if (sw_campo_nb==1 && sw_salto_linea==0) {
```

```
                msg_sin = msg_sin.substr(0, i) +  +  + msg_sin.substr(i + 1);
                ult_plus = i;
                sw_salto_linea = 1;
                sw_campo_nb = 0;
            } else {
                msg_sin[i] = ' ';
                msg_sin = msg_sin.substr(0, i) + " " + msg_sin.substr(i + 1);
            }
        }
    }
    if (ult_plus>=0 && ult_char_campo>=0 && ult_char_campo<ult_plus) {
        msg_sin[ult_plus] = ' ';
        msg_sin = msg_sin.substr(0, ult_plus) + " " + msg_sin.substr(ult_plus +
    }

    // Compruebo que no haya caracteres inválidos con una expresión regular - Fi
    var patron = /[a-zA-Z<>=+_\[\]\{\}\- |().:º,áéíóúñüÑ\d]/;
    var sw_error = false;
    for (var i = 0;i < msg_sin.length; i++) {
        if (msg_sin.charAt(i).match(patron)) {
            sw_error = false;
        } else {
            sw_error = true;
            break;
        }
    }
    if (sw_error) {
        pos_error = i;
        return "El mensaje contiene caracteres no permitidos.";
    }

    msg_sal = msg_sin;

    if (msg_sal.length<=0) {
        pos_error = 1;
    return "El mensaje no tiene contenido";
}

if (!this.esCampo(msg_sal.charAt(0))) {
    pos_error = 1;
    return "El mensaje debe comenzar por el nombre de estructura";
}

nombres = [];
estructuras = [];
var tokens = [];
var pos_tokens = [];

// Guardo los campos del mensaje en nombres[],
// Guardo las estructuras del mensaje en estructuras[],
// la secuencia de constructores gramaticales en tokens[] y
// la posicion de cada token en el mensaje en pos_tokens[]
var sw_campo = 0;
var nombre = "";
var pos_campo = 0;
var sw_pos_campo = 0;
var sw_parentesis = 0;
sw_error = false;
for (var i = 0;i < msg_sal.length; i++) {
    if (this.esCampo(msg_sal.charAt(i)) || (msg_sal.charAt(i)=="+" && sw_parentes
        if (msg_sal.charAt(i)=="(") {
            sw_parentesis = 1;
        }
        if (msg_sal.charAt(i)==")") {
            sw_parentesis = 0;
        }
        if (sw_pos_campo==0) {
            sw_pos_campo = 1;
            pos_campo = i;
        }
        nombre = nombre + msg_sal.charAt(i);
```

```
        if (this.trim(nombre).length>0) {
            sw_campo = 0;
        }
    } else {
        sw_parentesis = 0;
        if (sw_campo==0) {
            sw_campo = 1;
            if (msg_sal.charAt(i)=="+" || msg_sal.charAt(i)==">" || msg_sal.charA
                || msg_sal.charAt(i)=="|") {
                tokens.push("c");
                pos_tokens.push(pos_campo);
            } else {
                if (msg_sal.charAt(i)=="=") {
                    tokens.push("s");
                    pos_tokens.push(pos_campo);
                } else {
                    pos_error = i;
                    sw_error = true;
                    break;
                }
            }
        }
        if (msg_sal.charAt(i)=="+" || msg_sal.charAt(i)==">" || msg_sal.charAt(i)
            || msg_sal.charAt(i)=="|") {
            if (this.trim(nombre).length>0) {
                nombres.push(this.trim(nombre));
            }
        } else {
            if (msg_sal.charAt(i)=="=") {
                if (this.trim(nombre).length>0) {
                    estructuras.push(this.trim(nombre).toUpperCase());
                }
            }
        }
        sw_pos_campo = 0;
        nombre = "";
        tokens.push(msg_sal.charAt(i));
              pos_tokens.push(i);
          }
          pos_error = i;
    }

    if (sw_error) {
        return "Error, un campo o estructura va seguido de un caracter distinto de '
    } else {
        if (this.trim(nombre).length>0 && sw_campo==0) {
            nombres.push(this.trim(nombre));
            tokens.push("c");
            pos_tokens.push(pos_campo);
        }
    }

    if (tokens.length<2) {
        pos_error = 1;
        return "El mensaje tiene contenido insuficiente";
    }

    if (tokens[0]!='s' || tokens[1]!='=') {
        pos_error = 1;
        return "El mensaje debe comenzar por 'nombre de estructura ='";
    }
    // Validamos la colocación de los campos
    sw_error = false;
    for (var i = 0;i < tokens.length; i++) {
        if (tokens[i]=='c') {
            if (i<(tokens.length - 1)) {
                if (tokens[i+1]!='+' && tokens[i+1]!='>' && tokens[i+1]!='}' && toke
                    pos_error = pos_tokens[i];
                    sw_error = true;
                    break;
                }
            }
        }
    }
```

```
    if (sw_error) {
        return "Los campos del mensaje deben finalizar con '+', '}', ']', '|' o '>'";
    }

    sw_error = false;
    for (var i = 0;i < tokens.length; i++) {
        if (tokens[i]=='c') {
            if (i==0) {
                pos_error = pos_tokens[i];
                sw_error = true;
                break;
            } else {
                if (tokens[i-1]!='+' && tokens[i-1]!='<' && tokens[i-1]!='{' && toker
                    pos_error = pos_tokens[i];
                    sw_error = true;
                    break;
                }
            }
        }
    }
    if (sw_error) {
        return "Los campos del mensaje deben estar precedidos con '+', '{', '[', '|'
    }

    sw_error = false;

    // Se comprueba la precedencia y la continuación del elemento '+'
    var sw_err_plus = 0;
    for (var i = 0;i < tokens.length; i++) {
        if (tokens[i]=='+') {
            if (i==0 || i==(tokens.length - 1)) {
                pos_error = pos_tokens[i];
                sw_error = true;
                break;
            } else {
                if ((tokens[i-1]!='c' && tokens[i-1]!='>' && tokens[i-1]!='}' && toke
                    sw_err_plus = 1;

                pos_error = pos_tokens[i];
                sw_error = true;
                break;
            }
        }
    }
    if (sw_error) {
        if (sw_err_plus==0) {
            return "El elemento '+' debe unir dos campos y/o estructuras ";
        } else {
            return "El elemento '+' debe unir un campo con otro campo o una estructura
        }
    }

    // Se comprueba la precedencia y la continuación del elemento '='
    var mens_elem = "";
    for (var i = 0;i < tokens.length; i++) {
        if (tokens[i]=='=') {
            if (i==0) {
                pos_error = pos_tokens[i];
                sw_error = true;
                mens_elem = "El elemento '=' debe ir precedido de un nombre de estruct
                break;
            } else {
                if (i==(tokens.length - 1)) {
                    pos_error = pos_tokens[i];
                    sw_error = true;
                    mens_elem = "La estructura debe tener un contenido";
                    break;
                } else {
                    if (tokens[i-1]!='s') {
                        pos_error = pos_tokens[i];
                        sw_error = true;
                        mens_elem = "El elemento '=' debe ir precedido de un nombre de
                        break;
```

```
                } else {
                    if (tokens[i-1]!='s') {
                        pos_error = pos_tokens[i];
                        sw_error = true;
                        mens_elem = "El elemento '=' debe ir precedido de un nombre de
                        break;
                    }
                    if (i<(tokens.length - 1)) {
                        if (tokens[i+1]!='{' && tokens[i+1]!='<' && tokens[i+1]!='[')
                            pos_error = pos_tokens[i];
                            sw_error = true;
                            mens_elem = "El elemento '=' debe ir seguido de un element
                            break;
                        }
                    }
                }
            }
        }
    }
    if (sw_error) {
        return mens_elem;
    }

    // Se comprueba la precedencia y la continuación del elemento '<'
    mens_elem = "";
    for (var i = 0;i < tokens.length; i++) {
        if (tokens[i]=='<') {
            if (i==0 || i==(tokens.length - 1)) {
                pos_error = pos_tokens[i];
                sw_error = true;
                mens_elem = "El elemento '<' debe ir entre campos y/o estructuras";
                break;
            } else {
                if (tokens[i-1]!='=' && tokens[i-1]!='[' && tokens[i-1]!='{') {
                    pos_error = pos_tokens[i];
                    sw_error = true;
                    mens_elem = "El elemento '<' debe ir precedido de un elemento '='.
                    break;
                }
                if (i<(tokens.length - 1)) {
                    if (tokens[i+1]!='c' && tokens[i+1]!='s'&& tokens[i+1]!='{'&& toke
                        pos_error = pos_tokens[i];
                        sw_error = true;
                        mens_elem = "El elemento '<' debe ir seguido de un nombre de c
                        break;
                    }
                }
            }
        }
    }
    if (sw_error) {
        return mens_elem;
    }

    // Se comprueba la precedencia y la continuación del elemento '{'
    mens_elem = "";
    for (var i = 0;i < tokens.length; i++) {
        if (tokens[i]=='{') {
            if (i==0 || i==(tokens.length - 1)) {
                pos_error = pos_tokens[i];
                sw_error = true;
                mens_elem = "El elemento '{' debe ir precedido de un elemento '=', '+'
                break;
            } else {
                if (tokens[i-1]!='=' && tokens[i-1]!='+' && tokens[i-1]!='[' && tokens
                    pos_error = pos_tokens[i];
                    sw_error = true;
                    mens_elem = "El elemento '{' debe ir precedido  , '[' , '|' o '<'
                    break;
                }
                if (i<(tokens.length - 1)) {
                    if (tokens[i+1]!='c' && tokens[i+1]!='s' && tokens[i+1]!='<' && tc
                        pos_error = pos_tokens[i];
```

```
                    sw_error = true;
                    mens_elem = "El elemento '{' debe ir seguido de un nombre de ca
                    break;
                }
            }
        }
    }
}
if (sw_error) {
    return mens_elem;
}

// Se comprueba la precedencia y la continuación del elemento '['
mens_elem = "";
for (var i = 0;i < tokens.length; i++) {
    if (tokens[i]=='[') {
        if (i==0 || i==(tokens.length - 1)) {
            pos_error = pos_tokens[i];
            sw_error = true;
            mens_elem = "El elemento '[' debe ir precedido de un elemento '=', '+',
            break;
        } else {
            if (tokens[i-1]!='=' && tokens[i-1]!='+' && tokens[i-1]!='{' && tokens[
                pos_error = pos_tokens[i];
                sw_error = true;
                mens_elem = "El elemento '[' debe ir precedido de un elemento '=',
                break;
            }
            if (i<(tokens.length - 1)) {
                if (tokens[i+1]!='c' && tokens[i+1]!='s' && tokens[i+1]!='{') {
                    pos_error = pos_tokens[i];
                    sw_error = true;
                    mens_elem = "El elemento '[' debe ir seguido de un nombre de ca
                    break;
                }
            }
        }
    }
    if (sw_error) {
        return mens_elem;
    }

    // Se comprueba la precedencia y la continuación del elemento '|'
    mens_elem = "";
    for (var i = 0;i < tokens.length; i++) {
        if (tokens[i]=='|') {
            if (i==0 || i==(tokens.length - 1)) {
                pos_error = pos_tokens[i];
                sw_error = true;
                mens_elem = "El elemento '|' debe ir precedido de un nombre de ca
                break;
            } else {
                if (tokens[i-1]!='c' && tokens[i-1]!='>' && tokens[i-1]!='}') {
                    pos_error = pos_tokens[i];
                    sw_error = true;
                    mens_elem = "El elemento '|' debe ir precedido de un nombre
                    break;
                }
                if (i<(tokens.length - 1)) {
                    if (tokens[i+1]!='c' && tokens[i+1]!='s' && tokens[i+1]!='<'
                        pos_error = pos_tokens[i];
                        sw_error = true;
                        mens_elem = "El elemento '|' debe ir seguido de un nombre
                        break;
                    }
                }
            }
        }
    }
    if (sw_error) {
        return mens_elem;
    }
```

```
}


// Se comprueba la correspondencia de los simbolos de apertura con los
// de cierre correspondientes
var wtokens = [];
for (var i = 0;i < tokens.length; i++) {
    wtokens[i] = tokens[i];
}

var sw_elem = 0;
sw_error = false;
var ult_open = "";
var ult_ind = -1;
var sw_corchete = 0;
var num_barras = 0;
while (sw_elem==0 && !sw_error) {
    ult_open = "";
    ult_ind = -1;
    sw_elem = 1;
    num_barras = 0;
    for (var i = 0;i < wtokens.length; i++) {
        sw_corchete = 0;
        if (wtokens[i]=='<' || wtokens[i]=='[' || wtokens[i]=='{') {
            num_barras = 0;
            ult_open = wtokens[i];
            ult_ind = i;
        } else {
            if (wtokens[i]=='|') {
                num_barras = num_barras + 1;
            }
            if (wtokens[i]=='>') {
                if (ult_open!='<') {
                    pos_error = pos_tokens[i];
                    sw_error = true;
                    break;
                } else {
                    sw_elem = 0;
                    wtokens[i] = " ";
                    wtokens[ult_ind] = " ";
                    break;
                }
            }
            if (wtokens[i]==']') {
                if (ult_open!='[') {
                    pos_error = pos_tokens[i];
                    sw_error = true;
                    break;
                } else {
                    sw_elem = 0;
                    wtokens[i] = " ";
                    wtokens[ult_ind] = " ";
                    break;
                }
            }
            if (wtokens[i]=='}') {
                if (ult_open!='{') {
                    pos_error = pos_tokens[i];
                    sw_error = true;
                    break;
                } else {
                    sw_elem = 0;
                    wtokens[i] = " ";
                    wtokens[ult_ind] = " ";
                    break;
                }
            }
        }
    }
    if (sw_error) {
        break;
    }
}
```

```
if (sw_error) {
    if (sw_corchete==1) {
        return "Una subestructura de especializacion debe contener al menos un el
    } else {
        if (ult_ind<0) {
            return "Un elemento de cierre no tiene el correspondiente elemento de
        } else {
            pos_error = pos_tokens[ult_ind];
            return "Un elemento '"+ult_open+"' no tiene el correspondiente elemen
        }
    }
}

if (sw_elem==1 && ult_ind>=0) {
    pos_error = pos_tokens[ult_ind];
    return "Un elemento '"+ult_open+"' no tiene el correspondiente elemento de ci
}

// Formateo del mensaje
var cont_blank = 0;
var new_msg = "";
var ind_nom = -1;
var ind_est = -1;
var sw_salto = 0;
for (var i = 0;i < tokens.length; i++) {
    if (tokens[i]=='s') {
        if (cont_blank>0 && i<(tokens.length - 1)) {
            for (var x = 0;x < cont_blank; x++) {
                new_msg = new_msg + " ";
            }
        }
        ind_est++;
        new_msg = new_msg + estructuras[ind_est] + " ";
        sw_salto = 0;
    }
    if (tokens[i]=='c') {
      if (cont_blank>0 && i<(tokens.length - 1)) {
          for (var x = 0;x < cont_blank; x++) {
              new_msg = new_msg + " ";
          }
      }
      ind_nom++;
      new_msg = new_msg + nombres[ind_nom] + " ";
      sw_salto = 0;
  }
  if (tokens[i]=='=' || tokens[i]=='+') {
      if (tokens[i-1]=='s' || tokens[i-1]=='c') {
          new_msg = new_msg + tokens[i] + "\n";
      } else {
          if (cont_blank>0 && i<(tokens.length - 1)) {
              for (var x = 0;x < cont_blank; x++) {
                  new_msg = new_msg + " ";
              }
          }
          new_msg = new_msg + tokens[i] + "\n";
      }
      sw_salto = 1;
  }
  if (tokens[i]=='{' || tokens[i]=='[' || tokens[i]=='<') {
      if (cont_blank>0 && i<(tokens.length - 1)) {
          for (var x = 0;x < cont_blank; x++) {
              new_msg = new_msg + " ";
          }
      }
      new_msg = new_msg + tokens[i] + "\n";
      sw_salto = 1;
      cont_blank = cont_blank + 2;
  }
  if (tokens[i]=='}' || tokens[i]==']' || tokens[i]=='>' || tokens[i]=='|') {
      if (sw_salto==0) {
          new_msg = new_msg + "\n";
          sw_salto = 1;
```

```
        }
    }

    Ext.getCmp('msgid').setValue(new_msg);

    return("OKRecibido: "+mensaje);
},


// Valida que la cadena este compuesta por caracteres validos
esCampo:function esCampo(caracter){
    var patron = /[a-zA-Z_\- ().:º,áéíóúñüÑ\d]/;
    if (caracter.match(patron)) {
        return true;
    }
    return false;
},

// Quita blancos a la izda. y a la dcha. de una cadena
trim:function trim(str){
    if (str=="") {
        return str;
    }
    var str_ret = "";
    var sw_blanco = 0;
    for (var i = 0;i < str.length; i++) {
        if (str.charAt(i)!=' ') {
            str_ret = str_ret + str.charAt(i);
            sw_blanco = 1;
        }  else {
            if (sw_blanco>0) {
                str_ret = str_ret + str.charAt(i);
                sw_blanco = 0;
            }
        }
    }
    if (str_ret=="") {
        return str_ret;
    }
    sw_blanco = -1;
    for(var i = (str_ret.length - 1); i >= 0 ; i--){
        if (str_ret.charAt(i)==' ') {
            sw_blanco = i;
        }  else {
            break;
        }
    }
    if (sw_blanco>=0) {
        str_ret = str_ret.substring(0, sw_blanco);
    }
    return str_ret;
},

// Codifica de string a hex
encodeToHex:function encodeToHex(strxx){
    var rxx="";
    var exx=strxx.length;
    var cxx=0;
    var hxx;
    while(cxx<exx){
        hxx=strxx.charCodeAt(cxx++).toString(16);
        while(hxx.length<3) hxx="0"+hxx;
        rxx+=hxx;
    }
    return rxx;
},
```

Las funciones para pasar de texto a hexadecimal y viceversa, se usan al guardar el mensaje y recuperarlo del mismo, respectivamente.

```
// Codifica de string a hex
encodeToHex:function encodeToHex(strxx){
    var rxx="";
    var exx=strxx.length;
    var cxx=0;
    var hxx;
    while(cxx<exx){
        hxx=strxx.charCodeAt(cxx++).toString(16);
        while(hxx.length<3) hxx="0"+hxx;
        rxx+=hxx;
    }
    return rxx;
},

// Decodifica de hex a string
decodeFromHex:function decodeFromHex(strxx){
    var rxx="";
    var exx=strxx.length;
    var sxx;
    while(exx>0){
        sxx=exx-3;
        rxx=String.fromCharCode("0x"+strxx.substring(sxx,exx))+rxx;
        exx=sxx;
    }
    return rxx;
},
```

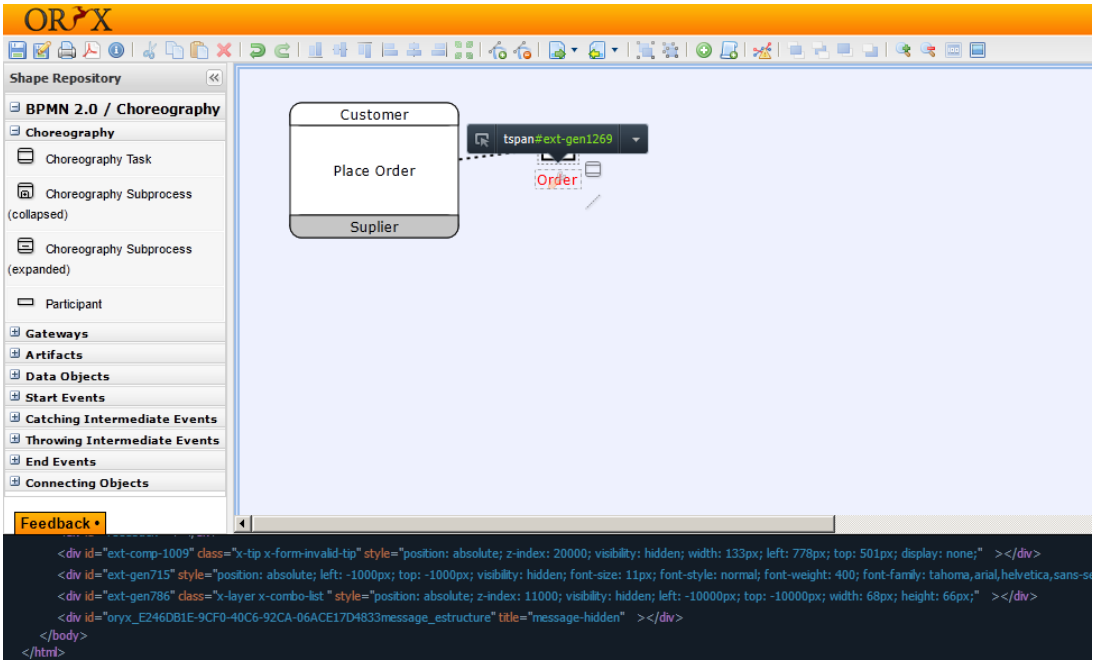
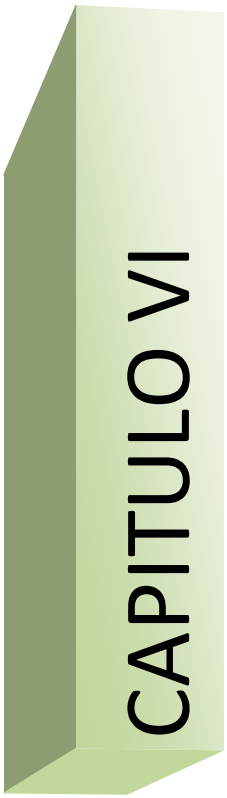


Fig. 42: Elemento para guardar Estructuras de Mensaje.



6. Conclusiones y Recomendaciones

6.1. Conclusiones.

Análisis de la Comunicación se aplica actualmente a los grandes proyectos industriales, una visión general del mismo se puede encontrar.

Asi mismo BPMN (Business Process Modeling Notation) es creado por los miembros de la OMG (Object Management Group), que se utiliza para asignar los procesos de las organizaciones. La optimización de procesos se aplica hoy en día a más empresas y diversas ramas de la competencia de mercado que fomenta una constante actualización y mejora de los procesos con el fin de obtener mejores resultados en el análisis final de los resultados.

Por tanto en esta tesina, el análisis de la comunicación adopta a BPMN en su versión BPMN2.0 mas concretamente los diagramas de coreografia con su objeto messaje para asociar con la técnica de estructuras de mensaje, esta notación basada en texto estructurado; de esta manera tambien apoyar la comunicación a BPMN2.0.

Para hacer una integración mas completa aum, se eligio una herramienta CASE de soporte que es ORYX ya que es una plataforma de código abierto para los acontecimientos relacionados con el modelado de procesos de negocio, definición declarativa de nuevos lenguajes de modelado de procesos (juegos de plantilla), la ampliación de la funcionalidad del editor a través de un mecanismo de plugin. Para tener un framework más profesional utiliza la librería EXTJS. Esta librería hizo posible obtener una interfaz sencilla, realizando entre los más importantes funcionalidades fueron como piedra angular de la gestión de estructuras de mensaje añadida se ha modificado la función que escucha el evento generado al pulsar un doble click en un objeto message de Oryx, funciones para windows donde se digite la estructura de mensaje, otro para la edición, restricciones para validar la gramatica y finalmente distinguir los objetos messge que fueron asociados a una estructura de mensaje.

Esta tesis de Master fue desarrollado en el marco de los siguientes proyectos del Centro **PROS:**

Project title: ORCA: Métodos de desarrollo orientados a la calidad de las tecnologías de la información (Methods for quality-oriented development of information technology)

Funding organisation: Generalitat Valenciana, cofinanced by ERDF.

Project type: Proyectos de investigación competitiva

Project id: PROMETEO/2009/015

Partners: Organismo: Universidad Politécnica de Valencia. PROS Research Centre

Duration, from 2009 **to** 2012

Project title: ProsREQ - Producción de Software Orientado a Servicios basada en Requisitos: La parte Funcional.

Funding organisation: Ministerio de Ciencia e Innovación

Project type: Proyectos de investigación competitiva

Project id: TIN2010-19130-C02-02, clave específica 20110040, financiación Generalitat Valenciana ACOMP/2011/186

Partners: Universidad Politécnica de Valencia.

Duration, from 01-01-2011 **to** 01-01-2014

4.Bibliografía

(MOF)2.0, M. O. (2008). *Xtex User Guide*. Retrieved from http://www.eclipse.org/Xtext/documentation/1_0_1/xtext.pdf

(OMG), O. M. (2006, Enero). *Business Process Modeling Notation (BPMN) Specification, Final Adopted Specification.* Retrieved from http://www.omg.org/bpmn/Documents/BPMN_1-1_Specification.pdf

Austin, J. (1962). *How to do things with words.* Oxford University Press.

Ballmer, T. B. (Springer (1981)). *Speech act classification: A study of the lexical analysis of English speech activity verbs.* Berlin.

Baniassad, E. C. (2004). Theme: an approach for aspect-oriented analysis and design. 26th International Conference on Software Engineering (ICSE 2004). *IEEE Computer Society*, pp. 158--167 .

Brown, J. C. (2006). A taxing problem: the complementary use of hard and soft OR in the public sector. *Eur J Oper Res 172(2)*, pp. 666--679.

Bubenko, J. B. (1998). *EKD User Guide. Dept. of Computer and Systems Sci-ence tech. report.* Stockholm University.

Castro, J. K. (2002). Towards requirements-driven information systems en-gineering: the Tropos project. *Information Systems 27*, pp. 365--389.

Chang, M. W. (1994). A speech-act-based negotiation protocol: design, implementation, and test use. *ACM Trans. Inf. Syst. 12(4)*, pp. 360--382.

Dietz, J. G. (1998). The Communicative Action Paradigm for business modelling - a research agenda. 3rd International Workshop on the Language Action Perspective on Communication Modelling (LAP 1998). *Jönköping Inter-national Business School (1998)*.

España, S. ,. (2009). Communication Analysis: a Requirements Engineering. *The Netherlands*, LNCS: 530-545.

Gloogle.com. (2000). *Oryx-Editor Web-based Graphical Business Process Editor*. Retrieved from http://code.google.com/p/oryx-editor

Kurtev, F. a. (2005). Transforming models with ATL. . *Satellite Events at the MoDELS 2005 Montego Bay, ,* (pp. 128-138.). Jamaica.

Sergio España, A. G. (2011). A practical guide to Message Structures: a modelling technique for information systems analysis and design. *14th Workshop on Requirements Engineering*, (p. 30). Brazil.

Systems, S. (2000). *Sparxsystems*. Retrieved from http://www.sparxsystems.com/products/ea/index.html

ANEXOS

APENDICE 1.

Oryx install and setup on Windows

1. Install Tools in following order (also described at: http://code.google.com/p/oryx-editor/wiki/SetupDevelopmentEnvironment)

1.1) Java JDK 5 or higher (http://java.sun.com/javase/downloads/index.jsp)

- Ensure that JAVA_HOME Enviroment variable is set to java installation dir (e.g. "C:\program files\java\jdk1.6")
    - Ensure that PATH Enviroment variable is extended with "\bin"-folder of java installation dir (e.g. "C:\program files\java\jdk1.6\bin")

1. 2) Firefox Browser (http://www.mozilla.com/de/firefox/).

- For debugging it is helpful to also install the Firebug Addon (https://addons.mozilla.org/de/firefox/addon/1843)

1.3) Tomcat 6 (unpack to arbitrary folder)

1.4) Eclipse IDE for Java EE Developers (http://www.eclipse.org/downloads/)

- You also need a subversion plugin for eclipse, e.g. subclipse (http://subclipse.tigris.org/) or any other subversion client.

1.5) Python version 2.5.2

- IMPORTANT: Use exactly this version, available here: http://www.python.org/download/releases/2.5.2
- Just follow the install wizard. No special setup is needed here.
- Install 32bit-Pyhton also on 64bit Windows!

1.6) PostgreSQL 8.3.x (IMPORTANT: Use exactly this version)

- Download the pgInstaller version to enable advanced installation options
- Use included wizard for installation
    - Within the install wizard select "install postgreSQL as service"
- ATTENTION: The page after "Initialize database cluster" (called "Enabled procedural languages") has to show PL/python (otherwise python installation failed. (Re-)Install PL/python.)
    - Ensure that PATH Enviroment variable is extended with "\bin"-folder of postgresql installation dir (e.g. "C:\program files\PostgreSQL\8.3\bin")
    - IMPORTANT: remember the password for windows user postgres and database. You will need them later...

2) In eclipse, checkout Oryx from SVN: "http://oryx-editor.googlecode.com/svn/trunk/"

- SVN-Link and information is also available here: http://code.google.com/p/oryx-editor/source/checkout

2.1) How to Checkout SVN in eclipse:

- Select "File -> Import..." and choose "SVN -> Project from SVN"
    - Follow the wizard. Create a new repository location using the Oryx SVN link above. Then Checkout the "trunk"-folder.

3) Setup Database Environment:

- Run comandline as user postgres: type "`runas /user:postgres cmd`" in run-prompt in Windows. On the Keyboard press the Windows-key and R at the same time to get the run-prompt. To logon use windows user (postgres) password from postgrSQL installation.
- change to schema file directory which is located in "oryx-workspace/poem-jvm/data/database/db_schema.sql"
- Run the following commands to import schema:
    - `createuser -U postgres --echo --pwprompt --encrypted poem`
        - the best is to use password `poem`
    - `createdb -U postgres --echo --encoding utf8 --owner poem poem`
    - `psql poem < db_schema.sql postgres`

4) Change Build Properties:

- open the file `build.properties` in root dir
- edit "deploymentdir" and set it to your apache tomcat wepapps folder (e.g. C:/apache-tomcat-6.0.32/webapps)
- If you used a password different than `poem` for the poem user (see 3, bullet point 3.1), edit line "`<property name="connection.password">poem</property>`" in `/poem-jvm/etc/hibernate.cfg.xml`

5) Build Oryx to produce deployable war-files:

- Right-Click on "build.xml" in root dir and select "Run As -> External Tool Configuration"
- Set buildfile to "build.xml", e.g. ${workspace_loc:/oryx/build.xml}
- Set base directory to oryx root directory, e.g. ${workspace_loc:/oryx}
- Choose tab "Targets" and check the following targets: "build-all" and "deploy-all"
    - Start build by clicking on run-button.
    - If the build was succsessfull, the two files "oryx.war" and "backend.war" should have been copied to your Apache Tomcat "\webapps" folder

6) Start Apache Tomcat with "apache-install-dir/bin/startup.bat".

- If Tomcat was already running, it automatically re-deploys the war files.

7) Start your browser and open "http://localhost:8080/backend/poem/repository".

7.1) Opening the backend enables users to create new process models or to browse/manage the process models that are already stored in oryx. A double-click on a process model then opens the model in the frontend, where models can be modified.

7.2) To store models in oryx you need an open-id (use google for more information)