Notaciones y lenguajes de procesos. Una visión global.

Juan Diego Pérez, 25179398X
jdperez.averroes@juntadeandalucia.es

Supervised by Prof. Dr. Amador Durán Toro

Departamento de
Lenguajes y Sistemas Informáticos
Universidad de Sevilla

Research Report submitted to the Department of Computer Languages
and Systems of the University of Sevilla in partial fulfilment
of the requirements for the degree of Ph.D. in Computer Engineering.
(Research Period)

Índice general

Índice de figuras

Índice de cuadros

Reconocimientos

A todo el departamento de Lenguajes y Sistemas Informáticos de la Universidad de Sevilla por su inestimable ayuda, en especial a mi supervisor, Dr. Amador Durán por su paciencia, orientación e ideas, al Dr. Antonio Ruiz por ser fuente inagotable de información y al profesor Carlos Müller por estar siempre que se le necesitó. Y por supuesto, a Zahira, que no se pone celosa de una máquina.

Resumen

Las notaciones y lenguajes de procesos llevan ya varias décadas usándose en numerosos campos de la industria. Su objetivo principal siempre ha sido la búsqueda de costes y tiempos óptimos. Sin embargo, el uso de esta técnica es relativamente reciente dentro del ámbito de la Ingeniería del Software. Su aplicación dentro de este campo, el de la Ingeniería del Software, no sólo se limita al intento de optimizar el proceso de desarrollo software en sí mismo, además se está utilizando como elemento de comunicación durante la elicitación de requisitos y como elemento fundamental dentro de enfoques como MDD (Model Driven Engineering) que buscan una paulatina automatización del proceso de desarrollo. El objetivo de este trabajo es repasar las notaciones y lenguajes de procesos más difundidos, realizar una comparativa de éstos e introducir algunas herramientas que nos permitan trabajar con ellos.

Capítulo 1

Introducción

El presente documento es la memoria de investigación correspondiente al periodo investigador del programa de doctorado "Tecnología e Ingeniería de Software" de la Universidad de Sevilla. El punto de partida para la elaboración fue el trabajo de investigación "Definición de modelos de procesos de desarrollo para fábricas BDD/SOA"propuesto por el departamento de Lenguajes y Sistemas Informáticos.Este trabajo de investigación proponía en un principio el siguiente programa:

- Estudio de las diferentes notaciones para la definición de procesos, en especial SPEM.
- Estudio de herramientas de software libre y/o propietario para el diseño de modelos de procesos.
- Estudio del modelo de desarrollo de Fábricas BDD/SOA que se quiere especificar.
- Aplicación de la notación SPEM al modelo estudiado en el apartado anterior.

Al abordar ya el primero de los puntos del programa se pudo comprobar la gran cantidad de lenguajes y notaciones existentes, la gran cantidad de nuevos conceptos que se debían asimilar y la gran cantidad de literatura a revisar y por ello se optó por la modalidad de *Análisis de Bibliografía* que está pensada para aquellos, que como el autor de esta memoria, carecen de resultados investigadores pero que sin embargo han realizado un importante esfuerzo para comenzar a encauzar su investigación mediante un profundo análisis de la bibliografía.

En este capítulo presentamos esta memoria de investigación. En su primer apartado expondremos cuáles han sido los motivos y objetivos que la han guiado, en el segundo daremos una visión global sobre el contexto de esta investigación y sobre la situación actual del modelado y definición de procesos de negocio dentro del mundo del desarrollo del software y en el último apartado describiremos la estructura del resto de este documento.

1.1. Motivaciones y objetivos

En la actualidad la situación económica mundial esta dominada por aspectos como la globalización, que conlleva la deslocalización de las empresas, la búsqueda constante de una reducción de costes para maximizar los beneficios, por continuas fusiones y adquisiciones de empresas y por una búsqueda constante de la mejora y optimización de todos los procesos.

Las tecnologías de la información van de la mano (no se sabe muy bien quien tira de quien) de las tendencias económicas globales y nos encontramos con un panorama donde la situación está determinada por los siguientes aspectos:

- Lucha entre las tendencias centralizadoras y descentralizadoras.
- Éxito de tecnologías como los servicios web y SOA.
- La importancia de la integración de aplicaciones, que en algunos casos se convierte en el aspecto más importante y complejo del desarrollo.
- La existencia de una gran cantidad de plataformas (estamos en la era del todo conectado), lo que, si no se toman medidas dispara enormemente los costes de los desarrollos.

Traducido a términos más técnicos y centrándonos en el marco de los desarrollos software nos damos cuenta de que la industria del software se encuentra en un estado de transición. Tecnológicamente estamos en el fin de la era web tal y como la conocemos ya que el navegador web no es un cliente viable para todo tipo de aplicaciones. Arquitectónicamente los sistemas han llegado a un grado de complejidad enorme en donde los bordes entre las diferentes partes son difusos y en donde los esfuerzos por integrar las aplicaciones constituyen muchas veces la parte más importante de los costes.

A la vez las necesidades de los clientes son cada día más complejas y piden aplicaciones mucho más generalistas (sin bordes), que aseguren el éxito comercial, con ciclos de desarrollo cada vez más cortos, con presupuestos cada vez más bajos, con la posibilidad de reingeniería para adaptarse a los continuos cambios de las empresas y que además permitan gestionar gran cantidad de datos y realizar actividades complejas.Todo esto nos conduce a una globalización de la información y los procesos. En este marco están tomando cada vez más importancia actividades como el modelado y el análisis de procesos de negocio.

Un **proceso de negocio** es un conjunto de actividades relacionadas dentro de una organización que tienen como objetivo conseguir un determinado resultado. La gestión de estos procesos, BPM (Business Process Management), conlleva una serie de actividades de las cuáles las más importantes son:

1. La definición de los procesos, normalmente mediante una notación formal, y la creación del correspondiente modelo.
2. La configuración de los procesos como paso previo a su ejecución.
3. La ejecución y/o simulación de los mismos.
4. El control y análisis de las distintas ejecuciones.

Los modelos de procesos de negocio se usan para mejorar la comunicación tanto entre el analista y el desarrollador como entre el analista y el ciente, se están utilizando para analizar el comportamiento de procesos de desarrollo de software con el objetivo de comprobar cuáles son las causas de el retraso en las fecha de entrega y que alternativas existen para reducir los costes y además, se están empezando a utilizar para integrar la definición, el modelado y el análisis de procesos dentro de las metodologías de desarrollo que tienen un enfoque MDD/MDA(Model Driven Development/Model Driven Architecture) [13] con el objetivo de ir automatizando, poco a poco, y en la medida de lo posible, el desarrollo de productos software dentro de un dominio determinado.

Los objetivos de esta memoria, una vez se optó por la modalidad de revisión bibliográfica, son:

- Adquirir una visión global sobre el estado del arte en el mundo del modelado y definición de procesos de negocio y conocer la situación de la industria en ese campo detectando las áreas en las que se está poniendo en práctica esta técnica.
- Profundizar en el conocimiento de las las notaciones y lenguajes más relevantes con el propósito de distinguir los diferentes enfoques existentes y las características diferenciadoras.
- Familiarizarse con la utilización de herramientas que soporten las notaciones descritas a lo largo de la memoria.
- Sentar las bases para afrontar con garantías futuros trabajos de investigación en este área.

1.2. Contexto de la investigación

La investigación sobre la definición formal y el modelado de procesos de negocio dentro del marco descrito en el apartado anterior constituye una tendencia actual de cuyo universo forman parte:

- Los grupos de trabajo sobre workflow.
- Los estándares y lenguajes de workflow.
- Las metodologías de modelado de procesos.
- Los patrones de workflow.

Existen numerosos grupos de investigación que desarrollan su trabajo en este área aunque debemos destacar los siguientes:

- OMG. Object Management Group.
- WfMC. Workflow Management Coalition.
- BPMI. Business Process Management Initiative.
- BPMG. Business Process Management Group.
- ebXML. UN/CEFACT and OASIS.

- Universidades, destacando Eindhoven University of Technology.
- WARIA. Workflow and Reengineering International Association.

En cuanto a los lenguajes y estándares que nos van a permitir realizar modelos de procesos de negocio destacamos:

- Diagrama de Actividad de UML.
- SPEM. Software Process Engineering Metamodel.
- BPMN. Businees Process Moedeling Notation.
- XPDL. XML Workflow Definition Language.
- jBPM-jPDL. jBOSS Process Definition Language.
- IDEF. ICAM Definition Language.
- ARIS-EPC. Event-Driven Process Chain.

Estos lenguajes y estándares no son por casualidad los mismos que los seleccionados para ser analizados posteriormente en esta memoria de investigación. Se han elegido por ser los más usados dentro del la industria, por ser los que más éxito están teniendo comercialmente o por ser los que están apoyados por organismos que tiene un gran peso dentro del ámbito de la ingeniería del software o del modelado y definición de procesos.

Dependiendo de las metodologías y estrategias empleadas pueden clasificarse en :

- Orientados a proceso: Se centran en las diferentes tareas a completar para llevar a cabo un proceso completo.
- Orientados a recurso: Se centran en la utilización y distribución de los recursos que son necesarios para llevar a cabo la realización del proceso.
- Orientados a datos: Se centran en la definición de los datos y en las transformaciones que sufren éstos a los largo del proceso.

Los patrones de workflow [32] surgen de la investigación de Wil van der Alst de la Eindhoven University of Technology y son la herramienta principal que hemos utilizado en esta memoria de investigación para el análisis de la expresividad de las diferentes notaciones presentadas. Se han elegido porque, tal y como se concluye en [29]:

- Están ampliamente difundidos.
- Han sido aceptados por la comunidad investigadora.
- Son comprensibles por los profesionales de la informática.
- Presentan el nivel de abstracción adecuado para comparar las características de los lenguajes y notaciones de modelado de procesos de negocio.

Estos patrones nos van a ayudar a describir los comportamientos de los procesos, la interacción entre los diferentes procesos y van a permitir a los desarrolladores construir modelos y documentarlos de forma eficiente. Son 20 y se dividen en 6 categorías:

- De control básico de flujo: Describen los aspectos elementales del control de flujo de los procesos.
- De ramificación avanzada y sincronización.
- Patrones estructurales: Permiten identificar limitaciones estructurales de los procesos, en especial aquellas relacionadas con bucles y terminaciones.
- Patrones con múltiples instancias: Comprenden aquellas situaciones en las que puede haber ejecutándose varias instancias de una misma actividad dentro de una misma instancia de un proceso.
- Patrones basados en estado: Permiten describir situaciones donde el siguiente paso de la ejecución de la instancia de un proceso viene determinado por el estado de la propia instancia.
- Patrones de cancelación: Para representar la terminación de actividades e instancias de procesos cuando concurren ciertas circunstancias.

Como resultado de todo este movimiento en el mundo de los procesos de negocio han surgido numerosas herramientas que pretender dar soporte a alguna de las cuatro actividades principales relacionadas con esta rama, ya sea la definición o el modelado, la configuración, la ejecución o simulación o el análisis, gestión y monitorización de los procesos. En esta memoria nos vamos a centrar en la fase de modelado y definición y analizaremos las siguientes herramientas:

- TIBCO Business Process Studio.
- Soyatec eBPMN.
- Borland Together Architect 2006.
- MagiDraw 12.
- jBOSS jBPM.

¿Por qué estas herramientas y no otras?. La respuesta es sencilla, principalmente porque son herramientas muy usadas, gratuitas y de fácil obtención. Como ya veremos a lo largo de toda esta memoria, hay muchísimas más herramientas pero muchas de ellas son difíciles de conseguir y además tienen unas licencias que son enormemente caras.

1.3. Estructura de este documento

El resto del trabajo se estructura de la siguiente forma. En los capítulos del segundo al octavo se hace una descripción de las notaciones y lenguajes que han sido incluidas en la memoria por su difusión e implantación dentro de la industria. El capítulo noveno hace una comparativa de estas notaciones de acuerdo a una serie de características. En el capítulo décimo se describen una serie de herramientas para el modelado de procesos de negocio y en el último capítulo se desarrollan una serie de conclusiones y se presentan las posibles líneas de investigación que podría abordar el autor a la hora de realizar su tesis doctoral.

Capítulo 2

Diagramas de Actividad

2.1. ¿Qué son los Diagramas de Actividad?

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. Estos diagramas utilizan clases del metamodelo de UML que se encuentran en los paquetes de la especificación dedicados a la descripción de comportamientos. Estos paquetes y las relaciones existentes entre ellos se pueden apreciar en la figura 2.1.

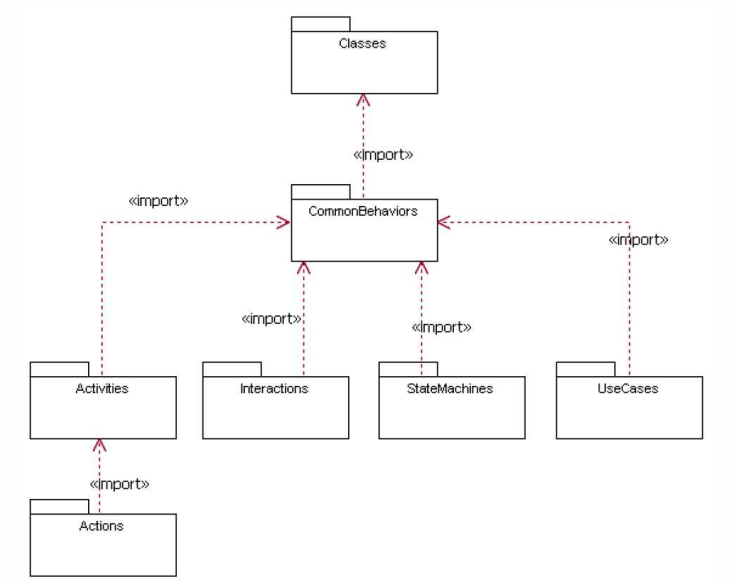


Figura 2.1: Paquetes de UML para la descripción de comportamientos

El objetivo de estos diagramas es *"describir lógica procedural, flujos de trabajo y procesos de negocio"*[10] lo que coincide perfectamente con la temática de esta memoria de investigación. Esto unido a la gran difusión de UML justifica

la inclusión de esta notación dentro del presente trabajo.

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]. Este gran cambio estuvo motivado por la falta de expresividad de los Diagramas de Actividad que en sus versiones anteriores dejaban sin cubrir muchos de los patrones de workflow [33]. Para dar solución a esta situación se modificó la semántica de los diagramas que pasaron de ser un caso especial de máquina de estados a tener una semántica de Redes de Petri basada en la producción y consumo de tokens por parte de los elementos que conforman el diagrama. Las principal diferencia entre estos dos enfoques es que:

- En los diagramas o máquina de estados las transiciones se producen automáticamente, pasando el control al estado siguiente una vez ha acabado el estado anterior. Tienen lo que se viene a llamar transiciones sin disparadores, o transiciones push.
- En las redes de Petri una acción comienza a ejecutare únicamente cuando tiene disponibles todas las transiciones o flujos precedente. Son transiciones pull.

2.2. 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.
- Nodos de Control: Distribuyen el control de la ejecución y los tokens a lo largo del diagrama.
- Nodos Objeto: Contienen datos de manera temporal a la espera de mover estos datos a lo largo del diagrama.

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

- Flujos
- Particiones
- Regiones de Expansión
- Excepciones
- Regiones de Actividad Interrumpibles
- Streaming

2.2.1. Nodos de Acción

A la hora de hablar de los nodos de acción tenemos que distinguir 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.

Podemos ver en ejemplo de una actividad con parámetros y condiciones y las acciones que la componen en la figura 2.2.

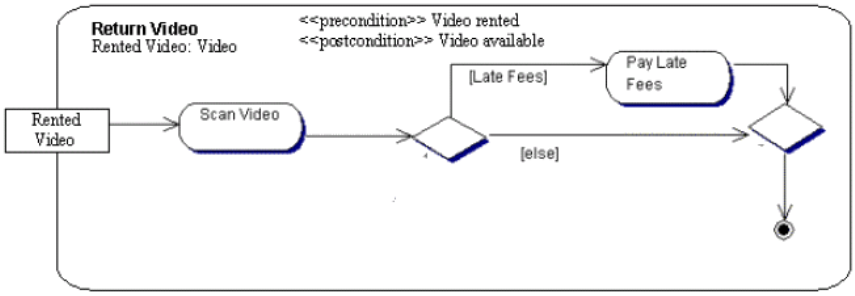


Figura 2.2: Ejemplo de Actividad y Acciones que la componen.

2.2.2. 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.

Podemos ver las distintas formas de modelar estos flujos en la figura 2.3.
Además para cada uno de estos flujos podemos, mediante etiquetas, especificar aspectos como:

- La multiplicidad: Para indicar el número de tokens objeto que se consumen cada vez (como máximo o mínimo).
- El límite superior: Para indicar el número de tokens que como máximo se pueden acumular en un flujo debido a que la acción en la que acaba el flujo no consume tokens.
- El peso: Para indicar cuántos tokens de los disponibles consumo.
- El orden en el que se van a consumir los tokens.

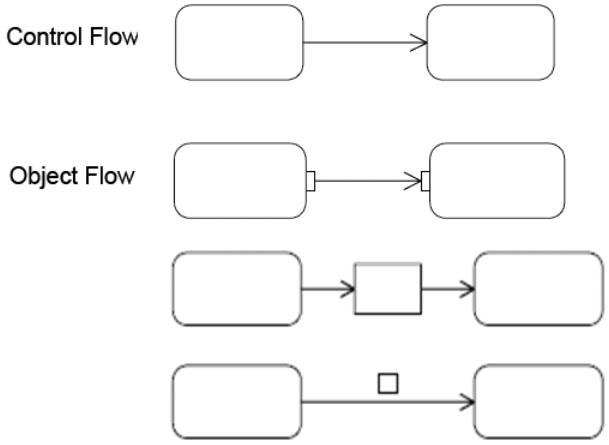


Figura 2.3: Tipos de flujos.

- Los posibles grupos de objetos que entran en una acción.
- Las transformaciones que van a sufrir los objetos que se pasan de una acción a otro.

2.2.3. Nodos de Control

Existen distintos tipos de nodos de control que podemos ver en la figura 2.4.

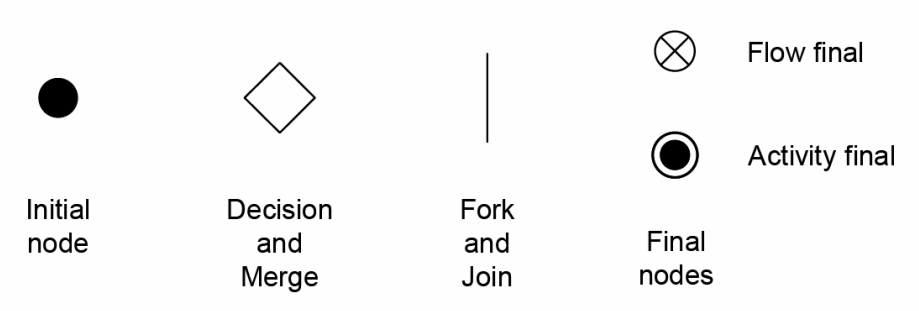


Figura 2.4: Nodos de Control de los Diagramas de Actividad

- **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 únicamente 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 figura 2.5.

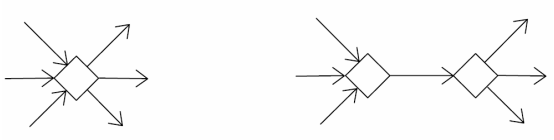


Figura 2.5: Combinación Merge/Decision

- **Fork Node:** Divide un único flujo de entrada en varios flujos de salida que se ejecutarán 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 únicamente se dispara cuando están disponibles todos los flujos de entrada. Podemos combinar nodos Fork/Join tal y como podemos apreciar en la figura 2.6

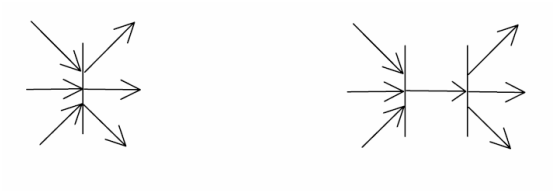


Figura 2.6: Combinación Fork/Join

- **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.

2.2.4. Nodos Objeto

Existen distintos tipos de nodos objetos que podemos ver en la figura 2.7.

- **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.

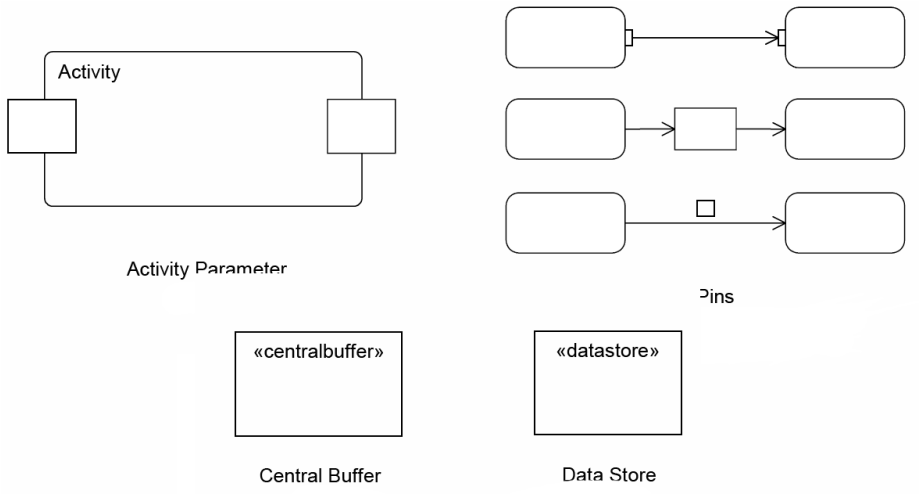


Figura 2.7: Nodos Objeto

- **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 éstos 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.

2.2.5. 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 2.8 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.

2.2.6. 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

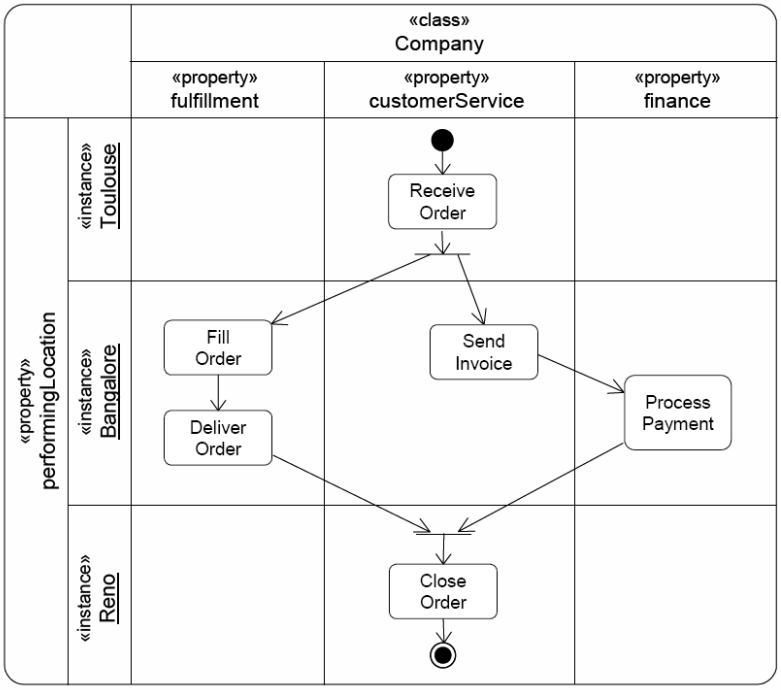


Figura 2.8: Ejemplo de Partición Bi-dimensional

y en el tipo de cada uno de ellos. Podemos ver un ejemplo de representación de una región de expansión en la figura 2.9.

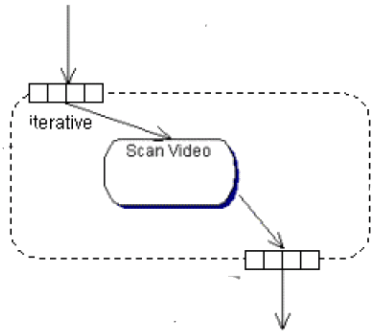


Figura 2.9: Ejemplo de Región de Expansión Iterativa

2.2.7. 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 figura 2.10

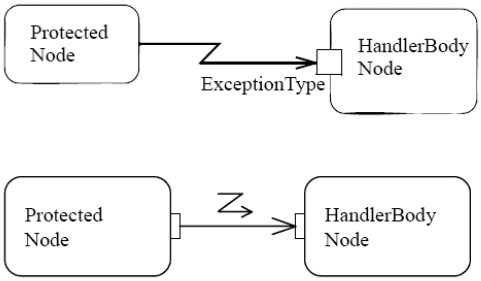


Figura 2.10: Ejemplo de utilización de excepciones

2.2.8. 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 figura 2.11

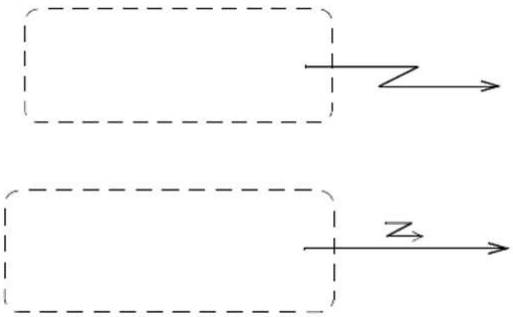


Figura 2.11: Regiones de actividad interrumpibles

2.2.9. Streaming

Con los Diagramas de Actividad de UML 2.X también podemos 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 tenemos varias notaciones alternativas que podemos ver en la figura 2.12.

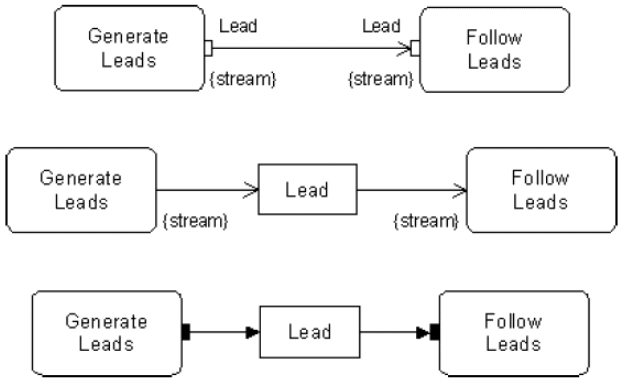


Figura 2.12: Ejemplo de acciones que se ejecutan en streaming.

2.3. Herramientas para Diagramas de Actividad

Desde la creación de UML han surgido innumerables herramientas. Las que según la propia OMG soportan los Diagramas de Actividad que nos ocupan quedan recogidas en [2] y algunas de ellas son:

- **Atportunity**
- **Altova UModel**
- **JUDE**
- **Concept Draw**
- **IBM Rational Software Architect**
- **EDGE UML Suite**
- **Mia-Studio**
- **Innovator**
- **Magic Draw**
- **Select Component Architect**
- **MDWorkbench**
- **Objecteering**
- **Enterprise Architect**
- **Power Designer**
- **Pattern Weaver**
- **Telelogic TAU**
- **Visual Paradigm**

2.4. Conclusiones sobre los Diagramas de Actividad

UML es la especificación de software más extendida de OMG, la organización más importante relacionada con la creación de estándares de software. Una de las partes de UML, los Diagramas de Actividad, se ha utilizado para el modelado de procesos de negocio aunque su uso para este tipo de propósito fue muy criticado dada la limitada expresividad de las versiones anteriores [33].

Para solucionar esta situación y dada la gran importancia que está tomando el modelado de procesos de negocio en el mundo del desarrollo del software la OMG tomó una serie de medidas. En primer lugar, tal y cómo se ha dicho anteriormente, reformó por completo los Diagramas de Actividad para conseguir que fueran una notación con la expresividad adecuada para modelar todo tipo de procesos de negocio [33]. En segundo lugar absorbió a la organización BPMI(Business Process Management Initiative)[12] y con ello BPMN, otra notación para el modelado de procesos de negocio que también es analizada en esta memoria. Por lo tanto nos encontramos con dos notaciones con el mismo propósito dentro de una misma organización. Sobre esta situación de conflicto y sobre cómo la está resolviendo la OMG se habla en el siguiente capítulo en el que analizaremos en profundidad BPMN.

Lo que si es cierto es que en relación a los Diagramas de Actividad debemos tener presente algunas cosas:

- Existen muchas herramientas para trabajar con ellos.
- Existe gran cantidad de procesos de negocio que están modelados usando esta notación.
- Los desarrolladores tienen gran experiencia utilizando tanto UML como sus Diagramas de Actividad.

2.5. Ejemplos de Diagramas de Actividad

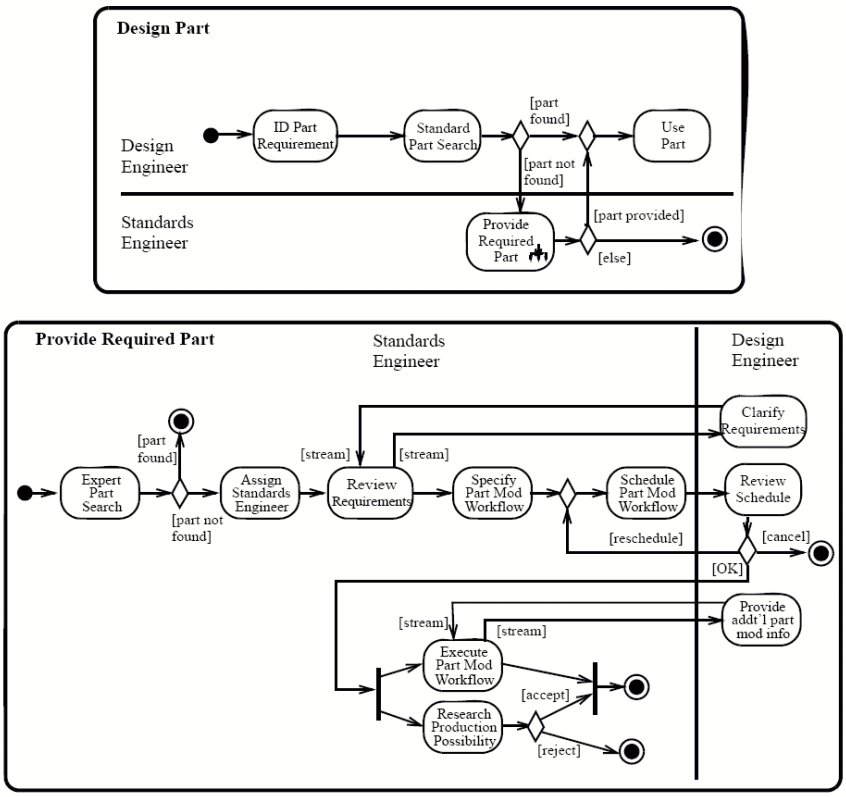


Figura 2.13: Ejemplos de Diagramas de Actividad

Capítulo 3

SPEM

3.1. ¿Qué es SPEM?

SPEM (Software Process Engineering Metamodel) es un estándar de la OMG [6] cuyo objetivo principal es proporcionar un marco formal para la definición de procesos de desarrollo de sistemas y de software así como para la definición y descripción de todos los elementos que los componen. Recientemente se ha publicado la versión draft apdoted (no definitiva) de SPEM 2.0 [16], una versión casi totalmente nueva que intenta solucionar las numerosas críticas que aparecieron tras la versión anterior SPEM 1.1 [15] que prácticamente no ha tenido ningún soporte por parte de la industria. La nueva versión, SPEM 2.0,se marcaba en sus inicios[14] los siguientes objetivos:

- Ser compatible con UML 2.0, una especificación con mucha más expresividad la hora de describir comportamientos [33].
- Definir un nuevo SPEM XML Schema que fuera compatible con MOF 2.0.
- Alinear SPEM con los estándares emergentes que no sean UML, como por ejemplo, BPMN que es también descrito en este mismo documento.
- Permitir el desarrollo de extensiones para SPEM con el objetivo de que sean usadas por herramientas para conseguir la automatización de procesos.

Tras el desarrollo de la especificación además de conseguir los propuesto en [14] se han añadido a SPEM nuevas capacidades como:

- Clara separación de la definición de los métodos de la aplicación de dichos métodos al desarrollo de un proceso concreto.
- Mantenimiento consistente de distintas alternativas de procesos.
- Soporte para diferentes modelos de ciclo de vida.
- Mecanismo flexible para dar soporte a la variabilidad y extensibilidad de los procesos.
- Ensamblado rápido de procesos mediante el uso de patrones.

- Utilización de los principios de la encapsulación para conseguir componentes de proceso reemplazables y reusables.

3.2. Metamodelo de SPEM

El metamodelo de SPEM 2.0 se estructura en 7 paquetes tal y como queda reflejado en la figura 3.1.

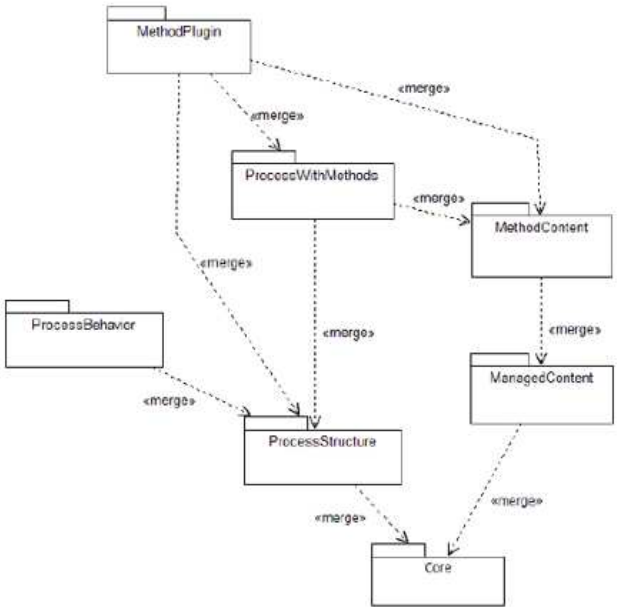


Figura 3.1: Paquetes del metamodelo de SPEM 2.0

Estos paquetes nos van a proporcionar las siguientes capacidades:

- **Core:** Contiene todas las clases y abstracciones que constituyen la base para el resto de los paquetes del metamodelo.
- **Process Structure:** Contiene las clases necesarias para la creación de modelos de procesos.
- **Process Behavior:** Para representar la parte dinámica de los procesos, su comportamiento.
- **Managed Content:** Nos va a permitir dotar a nuestros procesos o sistemas de anotaciones y descripciones que no pueden ser expresadas como modelos y que por lo tanto deben ser documentadas y gestionadas como descripciones en lenguaje natural.
- **Method Content:** Contiene los conceptos de SPEM 2.0 relacionados con los usuarios y la organización. Estos conceptos son necesarios para construir una base de conocimiento sobre desarrollo que pueda ser utilizada independientemente del proceso o proyecto específico.

- **Process With Methods:** Contiene los elementos necesarios para integrar los conceptos del paquete Process Structure con los conceptos y elementos del paquete Content Method.
- **Method Plug-In:** Introduce los conceptos para diseñar, gestionar y mantener repositorios y librerías de Methods Content y Procesos.

3.2.1. SPEM Core

El paquete SPEM Core constituye, tal y como se dijo en el apartado anterior, la base sobre la que se construyen el resto de los paquetes de SPEM 2.0. Sus superclases principales son Extensible Element, Kind, Parameter Direction Kind, Work Definition, Work Definition Parameter y Work Definition Performer Map y nos van a permitir dar soporte a dos de las capacidades de SPEM:

- Crear clasificaciones de las clases de SPEM 2.0. de acuerdo a las necesidades de los usuarios.
- Tener disponibles un conjuntos de clases abstractas para describir el trabajo mediante un proceso SPEM 2.0.

Son especialmente interesantes las tres últimas:

- **Work Definition:** Es una clase asbtracta que nos va a permitir generalizar cualquier tipo de trabajo dentro de una especificación en SPEM 2.0. Es uno de los elementos más importantes de la especificación y su relación con el resto de las clases la podemos apreciar en la figura 3.2
- **Work Definition Parameter:**: Es una generalizacíón de los elementos del proceso que representan parámetros de entrada o de salida para un Work Definition determinado.
- **Work Definition Performer Map:** Es clase abstracta que nos sirve para generalizar la relación entre aquel que realiza un trabajo y un Work Definition.

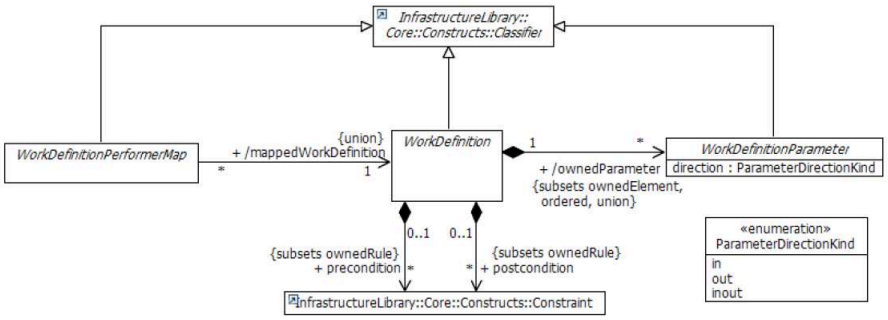


Figura 3.2: SPEM 2.0 Work Definition y sus relaciones

3.2.2. SPEM Process Structure

Contiene los elementos básicos para definir procesos de desarrollo mediante un mecanismo de descomposición. El elemento principal es la actividad (Activity) que a su vez se puede descomponer en otras actividades y contener otro tipo de elementos como MileStones, Role Uses etc. . . . Podemos ver los elementos del paquete y sus relaciones el figura 3.3

Las clases más relevantes de este paquete son:

- **Activity:** Elemento que nos sirve para representar la unidad básica de trabajo o para representar un proceso en sí mismo.
- **Activity Use Kind:** Para reusar una actividad relacionándola con otra.
- **Breakdown Element:** Clase abstracta que nos sirve para generalizar cualquier tipo de elemento que forma parte de una actividad de nivel superior.
- **MileStone:** Un evento significativo dentro del proceso de desarrollo.
- **Process Element:** Clase abstracta que generaliza cualquier elemento que forma parte de un proceso expresado en SPEM 2.0.
- **Process Parameter:** Especialización de Work Definition Parameter y Breakdown Element que sirve para representar entradas y salidas de un proceso.
- **Process Performer Map:** Para representar las relaciones entre una actividad y las instancias de Role Use que indican que el Role Use participa de alguna manera en el trabajo desempeñado en la Activity.
- **Process Responsibility Assignment Map:** Elemento que representa una relación entre instancias de Role Use y un único Work Product Use.
- **Role Use:** Para representar quién realiza o participa en una Activity.
- **Work BreakDown Element:** Especialización de Breakdown Element que se usa para representar alguna clase de trabajo.
- **Work Product Use:** Especialización de Breakdown Element que se usa para representar o bien cualquier entrada o salida de una actividad o bien cualquier participante en la actividad.
- **Work Product Use Relationship:** Para expresar relaciones (que pueden ser de distintos tipos) entre Work Products.
- **Work Sequence:** Relación entre dos Work Breakdown elements que sirve para expresar que el comienzo de uno depende del final del otro.
- **Work Sequence Kind:** Para expresar relaciones de orden entre dos Work BreakDown Element de manera más general que en el elemento anterior.

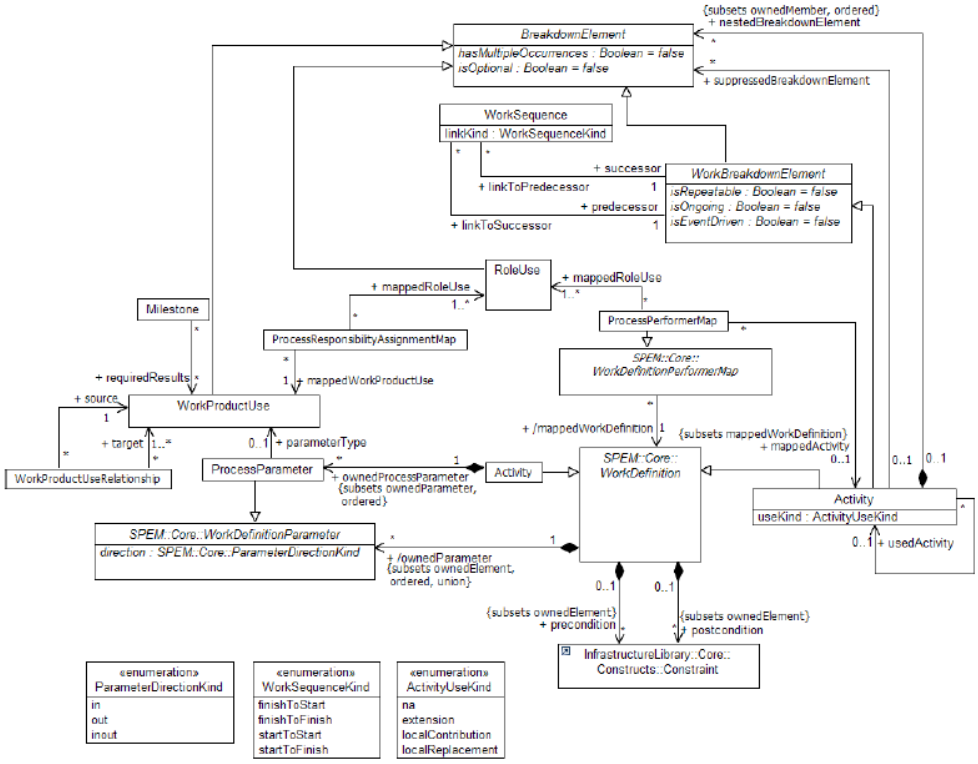


Figura 3.3: SPEM 2.0 Metamodelo del Paquete Process Structure

3.2.3. SPEM Process Behavior

SPEM 2.0 deja en manos del usuario la elección de la notación que crea más conveniente para modelar el comportamiento dinámico del sistema. No introduce por lo tanto ningún formalismo y únicamente define como enlazar los modelos creados con otras notaciones (BPMN, Diagramas de Actividad . . .) con los elementos del paquete Process Structure del metamodelo.

3.2.4. SPEM Managed Content

El paquete Managed Content nos proporciona todo lo necesario para gestionar descripciones textuales de procesos y elementos. Podemos apreciar los elementos del paquete y sus relaciones en la figura 3.4

El elemento pincipal de este paquete es la clase abstracta Describable Element que es la superclase de todos los elementos del sistema para los cuáles es posible tener una descripción textual. Además de esa clase, que nos va a permitir describir (de una u otra manera) todos los elementos de mi modelo de proceso, los otro componentes fundamentales del paquete son:

- **Category:** Para clasificar en categorías los distintos elementos.
- **Content Description:** Para almacenar descripciones textuales de un Describable Element.
- **Guidance:** Elemento que nos proporciona información adicional sobre cualquier Describable Element. Pueden ser de distintos tipos guías, plantillas, checklists etc. . . .
- **Metric:** Define medidas sobre Describable Elements.
- **Section:** Para estructurar en distintas partes las descripciones de los elementos

3.2.5. SPEM Method Content

El paquete Method Content contiene los elementos principales de los métodos como Roles, Tareas(Tasks), Pasos(Steps) y WorkProduct Definitions. Su objetivo principal es definir las tareas, organizarlas en distintos pasos, definir cuáles son los productos de entrada y salida de cada una de ellas y especificar quién ha sido el que ha realizado la tarea. Los elementos de este paquete y las relaciones entre ellos quedan descrito en el metamodelo que podemos apreciar en la figura 3.5

- **Default Responsibility Assignment Map:** Para espeficicar relaciones entre instancias de Role Definition y Work Product Definition.
- **Default Task Definition Parameter:** Especialización de Work Definition Parameter que nos permite usar Work Product Definitions como un atributo opcional.
- **Default Task Definition Performer Map:** Para relacionar instancias de Role Definition con instancias de Task Definition.

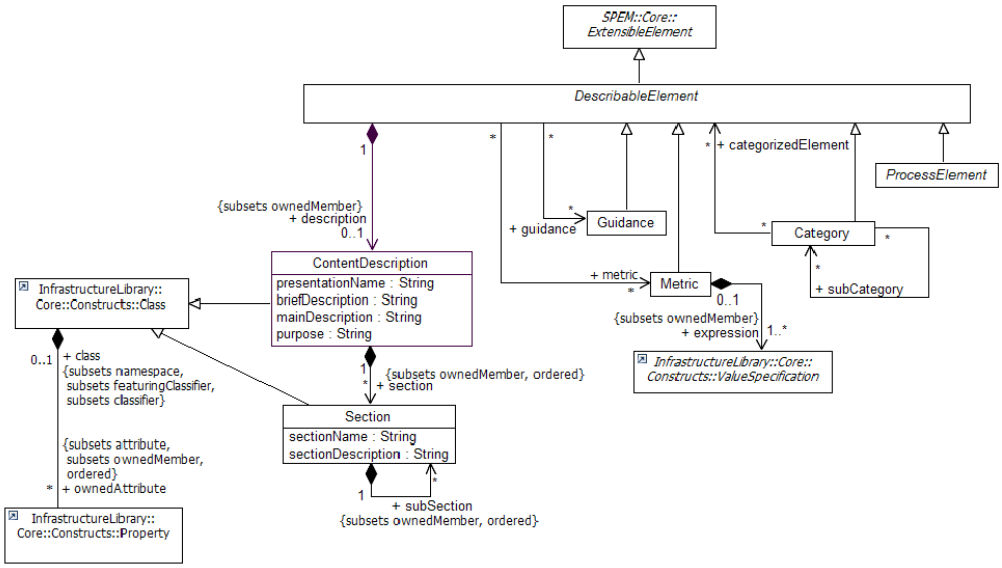


Figura 3.4: SPEM 2.0 Metamodelo del Paquete Managed Content

- **Method Content Element:** Generalización de cualquiera de los elementos que van a componer un método.
- **Optionality Kind:** Enumeración para describir si los Task Definition Parameters son opcionales u obligatorios.
- **Qualification:** Para documentar las habilidades o competencias que deben poseer los Role Definitios.
- **Role Definition:** Conjunto de habilidades, competencias y responsabilidades de un participante o de un conjunto de participantes.
- **Step:** Para organizar una Task Definition en partes o subunidades de trabajo.
- **Task Definition:** Para definir el trabajo llevado a cabo por las instancias de Role Definition, es decir una unidad asignable de trabajo.
- **Tool Definition:** Especificación de la participación de una aplicación (de cualquier tipo) para la realización de una tarea.
- **Work Product Definition:** Cualquier cosa que es consumida, producida o modificada por las tareas. Constituye la unidad básica para conseguir la reutilización.
- **Work Product Definition Relationship:** Prácticamente tiene la misma semántica que Work Product Use Relationship sin embargo se utiliza para representar relaciones por defecto entre Work Products que han sido definidas en las descripciones generales de los métodos.

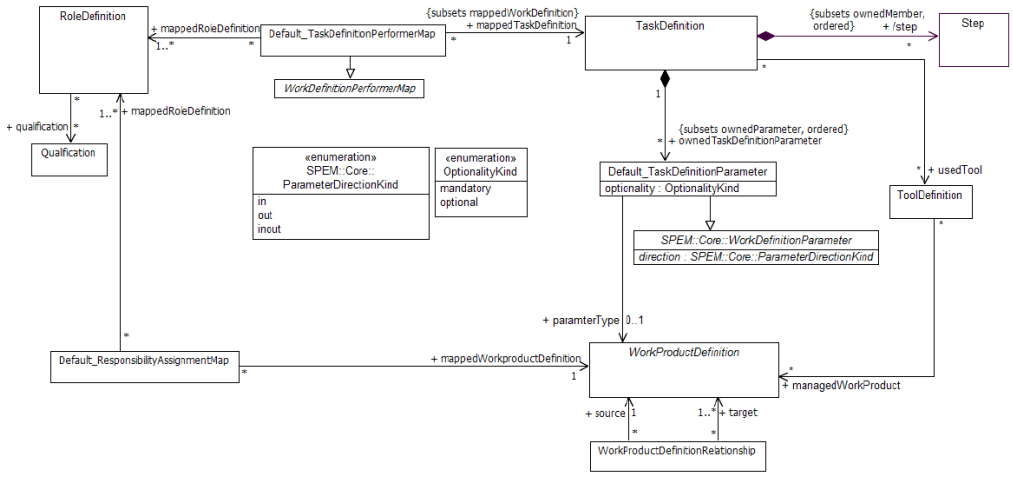


Figura 3.5: SPEM 2.0 Metamodelo del paquete Method Content

3.2.6. SPEM Process With Methods

El paquete Process With Methods proporciona las estructuras de datos necesarias para reflejar las buenas prácticas de la industria y permitir la reutilización entre distintas instancias de métodos y procesos. Los elementos del paquete y sus relaciones pueden verse en la imagen 3.6

- **Activity:** Grupo de distintos elementos (otras actividades, task uses, roles, milestone etc...) definidos en un espacio de nombres y para los que se han descrito una serie de relaciones según el método o proyecto en el que nos encontremos.
- **Breakdown Element:** Generalización de Process Element que es parte de una estructura compuesta por varios elementos.
- **Composite Role:** Role Use especial que referencia a más de un Role Definition con el objetivo de simplificar.
- **Method Content Kind:** Refinamiento de Kind (del paquete Core) para Method Content Element.
- **Method Content Package:** Para describir un paquete que se caracteriza por contener únicamente Method Content Elements.
- **Method Content Packageable Element:** Cualquier elemento que puede ser empaquetado en un Method Content Package.
- **Method Content Use:** Generalización para Breakdown Elements que hace referencia a un Method Content Element concreto.Es el concepto clave para entender la separación entre process y method content. Los Method Content poseen una serie de elementos y una serie de relaciones que pueden ser modificados para un Process particular para el cuál ha sido creado el Method Content.

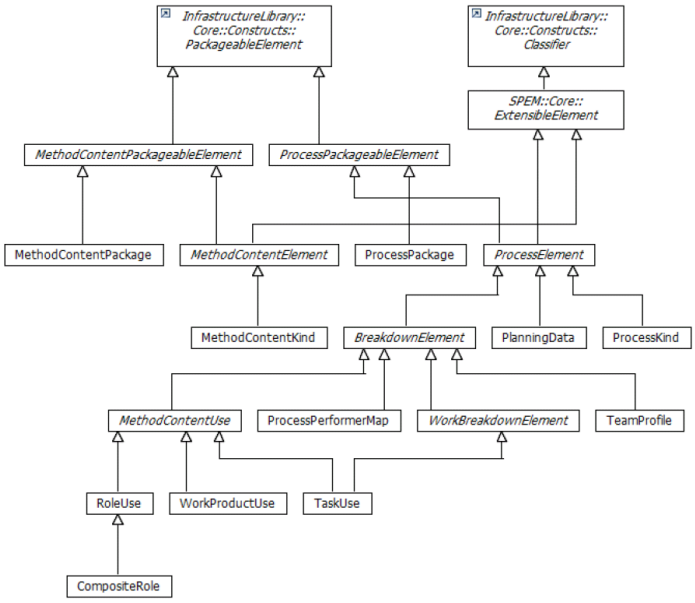


Figura 3.6: SPEM 2.0 Estructura del paquete Process With Methods

- **Planning Data:** Process Element que añade información sobre planificación.
- **Process Kind:** Refinamiento de Kind (paquete Core) para referirse a los Process Element.
- **Process Package:** Un paquete que únicamente puede contener Process Elements.
- **Process Packageable Element:** Cualquier elemento que puede ser empaquetado dentro de un Process Package.
- **Process Performer Map:** Extiende el Process Performer Map del paquete Process Structure añadiéndole una asociación adicional con Task Use.
- **Role Use:** Un role en el contexto de una actividad determinada.
- **Task Use:** Task Definition dentro del contexto de una actividad determinada.
- **Team Profile:** Agrupación con estructura jerárquica de Role Uses o Composite Roles.
- **Work Product Use:** Work Product Definition dentro del contexto de una actividad determinada.

3.2.7. SPEM Method Plug-In

El paquete Method Plug-In proporciona capacidades para gestionar librerías de Method Content y Process permitiendo mecanismos de extensibilidad y variabilidad mediante la reutilización de Method Content y Process determinados. La estructura de este paquete, sus elementos y las relaciones entre ellos se pueden apreciar en la figura 3.7.

- **Actividad:** Extensión de Activity para proporcionarle capacidades de variabilidad.
- **Method Configuration:** Subconjunto lógico dentro de una Method Library definido mediane una selección de Method Packages.
- **Method Library:** Contenedor de Method Plugins y definiciones de Method Configuration. Puede albergar cualquier elemento de SPEM 2.0.
- **Method Library Packageable Element:** Cualquier elemento que puede contener una Method Library.
- **Method Plugin:** Una unidad de almacenamiento de la configuración, modularización, extensión, empaquetado e implantamción de Process y Method Content.
- **Method PluginPackageable Element:** Cualquier elemento que puede ser empaquetado dentro de un Method Plugin.
- **Process Component:** Especialización de Process Package a la que se le pueden aplicar los conceptos de encapsulación y que contiene exactamente una Activity.
- **Process Component Use:** La aplicación de un Process Component dentro de cualquier otro Process.
- **Section:** Extiende Section (del paquete Managed Content) con capacidades de variabilidad.
- **Variability Element:** Para proporcionar capacidades de variación y extensión a los elementos de SPEM que derivan de él.
- **Variability Type:** Enumeración de valores (contributes, replaces,extends,extends-replace) para instancias de Variability Element.
- **WorkProduct Port:** Entradas y salidas de Work Products para un Process Component.
- **Work Product Connector:** Para conectar Work Product Ports con el objetivo de ensamblar Process Components.

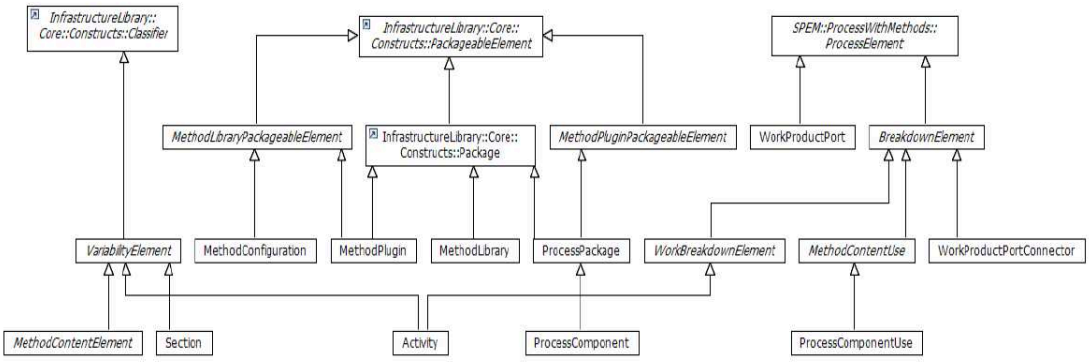


Figura 3.7: SPEM 2.0 Estructura del paquete Method Plugin

3.3. SPEM 2.0 como perfil.

Además de como metamodelo SPEM 2.0, al igual que sus versiones anteriores, se define como un perfil UML. Con la creación del perfil los creadores pretenden que se puedan seguir utilizando para SPEM la gran cantidad de herramientas que han sido creadas para trabajar con UML. De esta manera SPEM, el perfil de SPEM y UML se relacionan dentro de los niveles de abstracción de la OMG tal y como podemos apreciar en la figura 3.8. Los estereotipos más

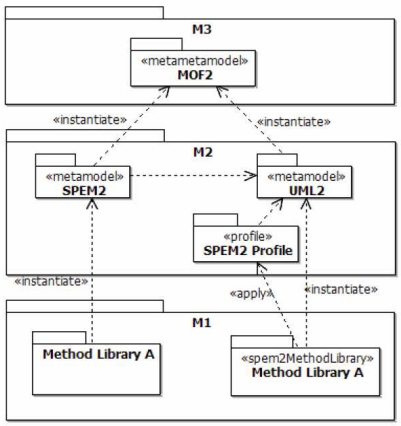


Figura 3.8: SPEM 2.0 Situación del perfil y del metamodelo dentro de los niveles de abstacción de la OMG.

importantes del perfil ,la Meta-/Superclase de cada uno de ellos y el icono correspondiente para su representación gráfica quedan recogidos en la tabla 3.1.El resto de estereotipos queda recogidos en [16].

Estereotipo	Meta-/Superclase	Icono

Activity	Work Definition, Planned Element /Action	
Category	Describable Element / Class	
CompositeRole	Role Use	
Guidance	Describable Element / Class	
Method Content Package	Package	
MethodLibrary	/Package	
MethodPlugin	/Package	
Metric	Guidance	
Milestone	WorkBreakdownElement / Classifier	
Process	Activity	
ProcessComponent	/Package	
ProcessPackage	Package	
Role Definition	MethodContentElement / Class	
RoleUse	BreakdownElement / Classifier	
Step	MethodContentElement, WorkDefinition	
TaskDefinition	MethodContentElement, WorkDefinition	

TaskUse	WorkBreakdownElement, PlannedELement / Classifier, Action	
TeamProfile	BreakDownElement / Classifier	
ToolDefinition	MethodContent Element / Class	
WorkProductDefinition	Method Content Element / Class	
WorkProductUse	BreakDownElement / Classifier	

Cuadro 3.1: Esteretipos del Perfil SPEM 2.0

3.4. Conclusiones sobre SPEM

La última versión de SPEM [16] ha sido publicada hace muy poco tiempo. Aún es pronto para ver si con esta nueva versión se consigue superar el fracaso de versión anterior, la 1.1, que fue adoptada por muy pocas empresas dentro de la industria. SPEM 1.1 [15] presentaba numerosos problemas entre ellos, según los propios desarrolladores de la OMG:

- Era una especificación de difícil comprensión.
- Poseía una semántica con algunas ambiguedades.

Además de estas afirmaciones hechas por los propios desarrolladores de la especificación, en SPEM 1.1 la descripción de la parte dinámica del proceso se hacía mediante la utilización de los diagramas de actividad de UML 1.4 que tenían una expresividad limitada [33]. SPEM 2.0 ha modificado drásticamente la versión anterior y ahora proporciona una serie de características que pueden hacer, una vez ya se haya publicado la versión totalmente definitiva, que las empresas se decidan por su utilización. Podemos destacar las siguientes:

- Se sigue definiendo como un perfil UML, lo que permite la utilización de la ingente cantidad de herramientas que dan soporte a esa especificación.
- Facilita la definición de procesos a partir del ensamblaje de partes ya existentes.
- Reconoce la necesidad de que puedan existir ciertas partes que no puedan formalizarse y que por lo tanto deben incluirse dentro del proceso mediante una descripción en lenguaje natural.
- Permite la creación de repositorios donde almacenar los procesos para posteriormente recuperarlos, editarlos y personalizarlos.
- Prevee la derivación de manera automática de planes de desarrollo y otro tipo de documentación.
- Establece una clara separación entre los elementos utilizados para la definición formal de un método (Method Content) y los elementos utilizados para describir la aplicación de dicho método a un proceso (Process) dentro de un proyecto en concreto. En la figura 3.9 se puede ver un esquema donde se distingue esta separación.
- Deja abierta a los desarrolladores la elección de la notación (ya sea de la OMG o de terceras partes) para describir la parte dinámica del proceso.Este hecho permite superar los problemas de expresivdad de la versión anterior ya que se elegirá la notación de acuerdo al dominio del problema. No obstante dentro de la propia especificación “sugiere“ especificaciones como los diagramas de actividad de UML 2.0 que han mejorado mucho la expresividad con respecto a las versiones anteriores [33] y BPMN que soporta la mayoría de los patrones de workflow [34]

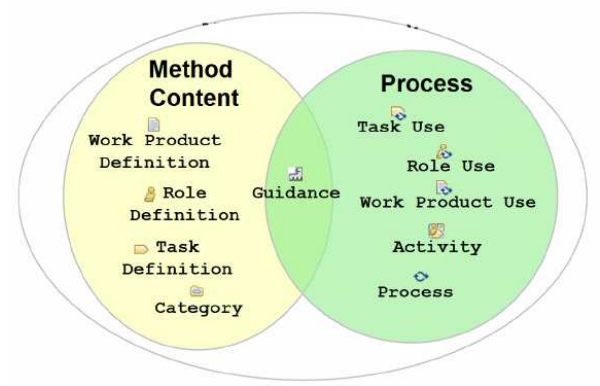


Figura 3.9: SPEM 2.0 Separación entre definición e implementación

3.5. Ejemplos en SPEM

```
Activity {kind: Iteration}: First Joint Requirements Planning (JRP) Workshop
    TaskUse: Define Owner Requirements
      ProcessPerformerMap {kind: primary}
        RoleUse: System Architect {kind : in}
      WorkDefinitionParameter {kind : in}
        WorkProductUse: EnterpriseArchitecture
      WorkDefinitionParameter {kind : out}
        WorkProductUse: Assessment of Current System {state: initial draft}
        WorkProductUse: Owner Requirements {state: initial draft }
      Steps
        Step : Define objectives based on stated needs
        Step : Define the key issues
        Step : Determine the relevant enterprise principles
```

Figura 3.10: SPEM 2.0. Fragmento de la descripción textual de la metodología DMR Macroscope de Fujitsu.

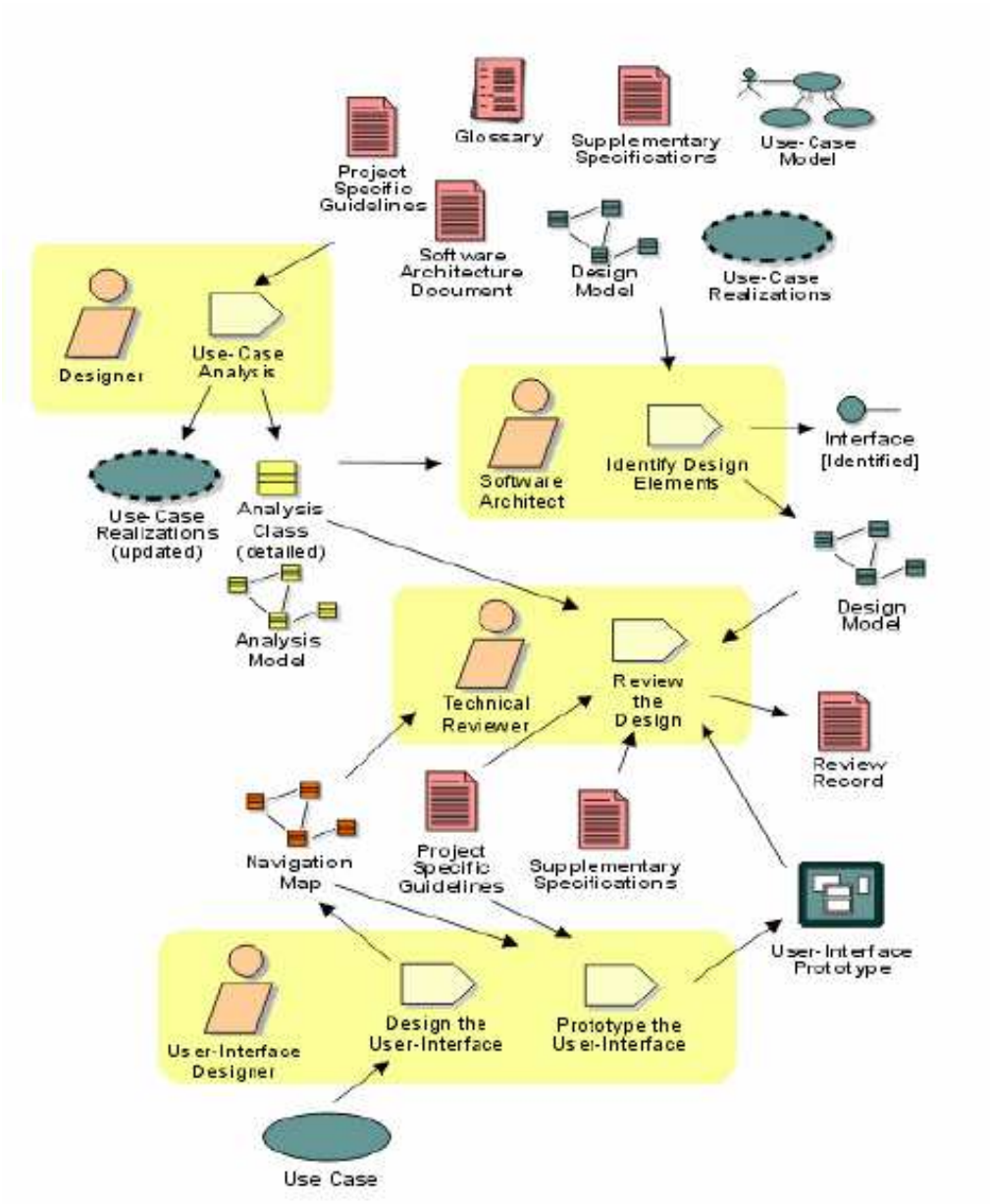


Figura 3.11: SPEM 2.0 Diagrama detallado de una actividad

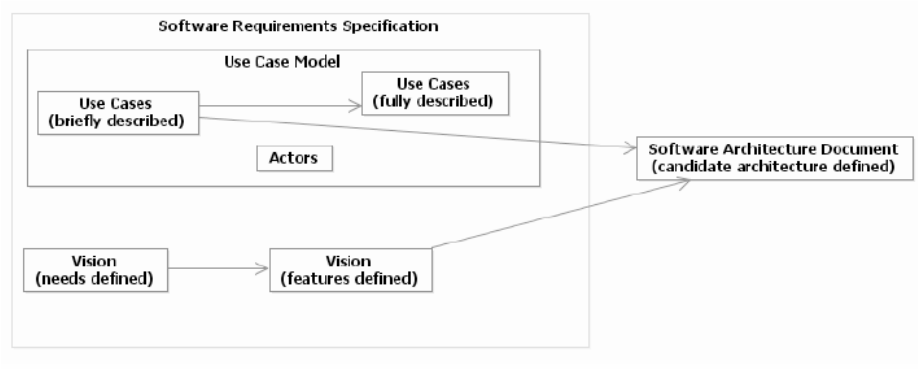


Figura 3.12: SPEM 2.0. Diagrama de dependencias entre Work Products

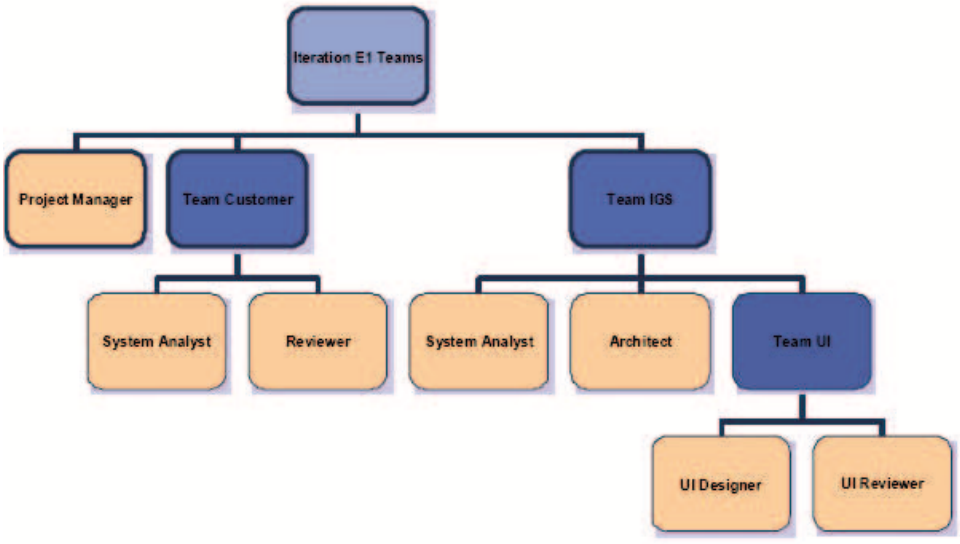


Figura 3.13: SPEM 2.0. Diagrama de componentes del equipo.

Capítulo 4

BPMN

4.1. ¿Qué es BPMN?

BPMN (Business Process Modelling Notation) es un estándar de la BPMI (Business Process Management Initiative)[12], organismo que ha sido absorbido recientemente por la OMG, cuyo principal objetivo es según [8] *"proporcionar una notación fácilmente comprensible por todos los usuarios del negocio, desde los analistas... los desarrolladores técnicos... hasta aquellos que monitorizarán y gestionarán los procesos"*.Otros objetivos importantes que se plantea esta especificación son:

- Crear puentes entre el diseño de los procesos de negocio y la implementación del proceso.
- Que los lenguajes basados en XML para describir procesos (como BPEL4WS) tengan una notación gráfica.

Los propios autores de BPMN por otro lado, reconocen haberse inspirado y haber recogido la experiencia de varios estándares:

- Diagramas de Actividad de UML.
- UML EDOC
- IDEF
- ebXML BPSS
- ADF Diagram
- RossetaNet
- LOVeM
- EPC

Es importante tener en cuenta que BPMN abarca únicamente los procesos de negocio, lo que significa que otro tipo de modelos relacionados (estructura de la organización, recursos, modelos de datos, estrategias, reglas de negocio etc...) quedan fuera de la especificación.

Todo este tipo de modelados y sus relaciones con BPMN serán definidos más formalmente conforme BPMN y otras especificaciones evolucionen, de hecho, *"aunque BPMN muestre el flujo de datos(mensajes) y las asociaciones de los artifacts con las actividades, no es un diagrama de flujo de datos"*.

4.2. Modelos en BPMN

Los modelos BPMN se expresan gráficamente mediante diagramas BPMN. Estos diagramas constan de una serie de elementos que nos van a permitir diferenciar claramente las tres secciones (o submodelos) básicos que existen en un modelo BPMN. Estas secciones son:

- Procesos de negocio privados (internos).
- Procesos abstractos (públicos).
- Procesos de colaboración (globales).

4.2.1. Procesos de negocio privados(internos)

Los procesos de negocio privados o internos son los que, dentro de una organización específica, han sido tradicionalmente llamados diagramas de flujo de trabajo o diagramas de workflow. Si usamos calles para representarlos este tipo de procesos únicamente ocuparán una calle aunque pueda interactuar, mediante el flujo de mensajes, con otros procesos de negocio de la misma clase.Podemos ver un ejemplo de este tipo de submodelo en la figura 4.1.

Figura 4.1: Proceso de negocio privado

4.2.2. Procesos de negocio abstractos (públicos)

Los procesos de negocio abstractos nos sirven para representar las interacciones existentes entre un proceso de negocio privado y, o bien otro proceso de negocio o bien un participante del proceso. En este tipo de procesos únicamente se incluyen aquellas actividades que se usan para comunicar un proceso privado con el exterior, así como las correspondientes estructuras de control de flujo.

Podemos ver un ejemplo de representación gráfica para este tipo de procesos en la figura 4.2.

4.2.3. Procesos de colaboración (globales)

.

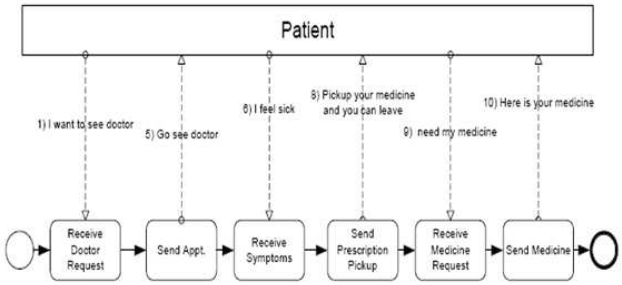


Figura 4.2: Proceso de negocio abstracto

Este tipo de procesos sirven para mostrar la interacción entre distintas entidades de negocio. Estas interacciones son definidas como secuencias de actividades que representan el intercambio de mensajes entre las distintas entidades. La colaboración se entiende como la comunicación entre dos o más procesos.Podemos ver un ejemplo de representación gráfica para este tipo de procesos en la figura 4.3.

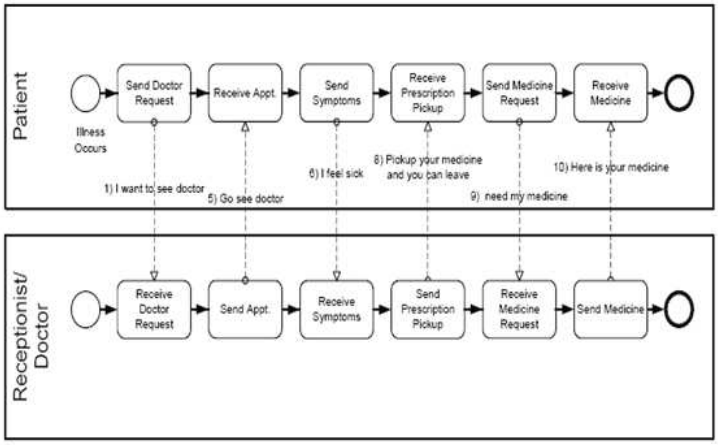


Figura 4.3: Proceso de colaboración

4.3. Diagramas BPMN

BPMN es una notación gráfica con la que podemos crear multitud de diagramas dentro de los tres tipos de submodelos (privado, público y de colaboración). Podremos además, crear diagramas con distintos tipos de modelos aunque siempre debemos tener en cuenta la advertencia de la propia especificación de BPMN *"debemos tener cuidado si combinamos demasiados tipos de submodelos... obtendremos un diagrama difícil de entender... por eso se recomiendo al modelador que se centre en un tipo de modelo para los diagramas"*.[8]

4.3.1. 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)

BPMN posee un conjunto reducido de elementos de este tipo. El objetivo de que sea un conjunto reducido es *"que los modeladores no tengan que aprender y memorizar gran cantidad de iconos"*[37]. Los tres objetos de flujo principales los podemos ver en la tabla 4.1.

Tipo	Descripción	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 compañia. Pueden ser atómicas o compuestas	
Pasarelas (Gateway)	Para controlar el flujo, puede ser una decisión tradicional, un join, un merge y un fork.	

Cuadro 4.1: 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 4.2

Tipo	*Descripción*	*Imagen*
Flujo de secuencia (Secuence Flow)	Para indicar el orden en el cuál 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	

Cuadro 4.2: Conectores en BPMN

Calles(Swinlanes)

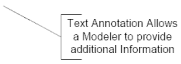
Las calles o swinlanes son un mecanismo que nos va a permitir clasificar las actividades de manera visual para ilustrar las distintas categorías o responsabilidades.Las distinas clases de ete tipo de objetos se puende apreciar en la tabla 4.3

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

Cuadro 4.3: Objetos Swinlane 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 4.4.

Tipo	Descripción	Imagen
Datos (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	

Cuadro 4.4: Artifacts en BPMN

4.3.2. Variaciones de los elementos básicos

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

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 imagen 4.4

Figura 4.4: BPMN. Tipos de eventos.

- **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.

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 4.5.

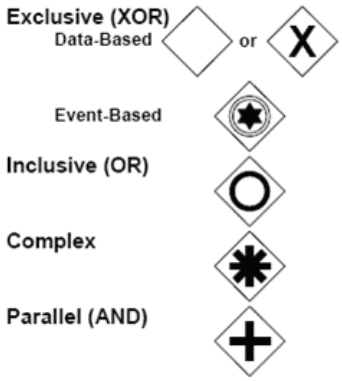


Figura 4.5: Tipos de Gateways en BPMN

- **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 más ramas de entrada(Inclusive Merge) o para propagar tokens a, al menos, una de las ramas de salida(Inclusive Desicion).
- **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).

Además de estas variaciones, BPMN posee otros símbolos y variaciones cuya descripción y semántica podemos ver en [8]

4.4. Herramientas BPMN

Desde la aparición de BPMN, y mucho más desde la absorción de BPMI por parte de la OMG, la notación BPMN ha tenido un éxito notable y como consecuencia de éste éxito han ido apareciendo gran cantidad de herramientas que dan soporte a esta especificación.

Las que según la propia OMG[20] implementan la especificación son las siguientes:

- **Appian Enterprise 5 Business Process Management Suite**
- **aXway: Process Manager**
- **BizAgi**
- **BOC Information Systems: ADONIS**
- **BOC Information Systems: ADONIS**
- **Borland® Together® Products: Together Architect® 2006 and Together Designer®**
- **Casewise: Corporate Modeler**
- **Cordys: Studio**
- **Fuego: Fuego 5 (BEA)**
- **Elixer Intelligent Software: eliXir BPMN-MDA Framework**
- **Fujitsu: Interstage Business Process Manager 7.1**
- **Graham Technology: GT-X**
- **Global 360: Business Optimization Server - Process Sketchpad**
- **IDS-Scheer: Aris**
- **Corel: iGrafx**
- **ILOG: JViews**
- **Intalio: n Designer**
- **Intellior AG: AENEIS**
- **ITpearls: Process Modeler for Visio**
- **Kaisha-Tec: ActiveModeler**
- **Lanner: Witness**
- **Lombardi Software: TeamWorks**
- **M1 Global: BPI Studio**
- **Mega International: Mega Suite**
- **No Magic: MagicDraw UML 10.0**

- **Orbus Software: iServer**
- **Pegasystems: BPMSuite**
- **Seagull Software: LegaSuite BPM**
- **Software AG: Enterprise Business Process Manager**
- **Popkin: System Architect**
- **Popkin: System Architect**
- **Proforma: ProVision**
- **Santeon: XIP BPM Platform**
- **Select Business Solutions: Select Component Factory**
- **Skelta: Skelta BPM.NET 2006**
- **Sparx Systems: Enterprise Architect 6.1**
- **Sun Microsystems: Studio Enterprise Edition**
- **Sybase: PowerDesigner® 12**
- **Troux: Metis 3.6 Enterprise Architecture Suite**

4.5. Conclusiones sobre BPMN

BPMN fue, ya se dijo anteriormente, una iniciativa de la BPMI. Esta institución fue absorbida por la OMG que es una de las organizaciones punteras en la creación de especificaciones para el desarrollo de software orientado a objetos.

A día de hoy es un hecho que cada día están teniendo más importancia los procesos de negocio y por extensión las herramientas que nos sirven para modelar, simular, supervisar y gestionar ese tipo de procesos.

El hecho de que la OMG haya absorbido BPMI hace preguntarse si los procesos de desarrollo de software pueden ser expresados mediante notaciones de procesos de negocio. Uno de los cambios más profundos al pasar de UML 1.5 a UML 2.0 (especificación estrella de la OMG) fue la nueva semántica con la que se dotó a los diagramas de actividad, que pasaron de tener semántica de máquinas de estados a redes de petri (pseudo redes de petri para los puristas). El motivo de éste cambio fue la falta de expresividad que para algunos investigadores de la rama del workflow tenían los citados diagramas de actividad. La mejora producida por este cambio fue notable, y podemos comprobar en [33] que la nueva versión de los diagramas de actividad cumple casi todos los patrones de workflow [32].

BPMN, según [34] cubre casi totalmente los patrones de workflow con lo cuál se le supone una gran expresividad a la hora de especificar procesos.La nueva especificación SPEM 2.0, [16] cuando se refiere a la descripción del comportamiento deja las puertas abiertas y nos da la libertad para elegir la especificación que creamos más conveniente para el dominio de nuestra aplicación, sugiriendo no obstante la misma BPMN o los Diagramas de Actividad. Nos encontramos por lo tanto con dos especificaciones dentro de la misma OMG, los Diagrama de

Actividad de UML y BPMN, cuyo objetivo es básicamente el mismo. Todo parece indicar que la OMG se está decantando por BPMN, no sólo como concesión a la organización absorbida sino también por otras razones expuestas en[28]:

- BPMN es capaz de expresar más patrones [34] que los diagramas de actividad [33], es decir, es más expresivo.
- BPMN es gráficamente más rico, con menos símbolos fundamentales, pero con más variaciones de éstos,lo que facilita su comprensión por parte de gente no experta.
- BPMN tiene el apoyo de la WfMC, una de las organizaciones más importantes en el campo del workflow que además de miembro de la propia OMG ha modificado una de sus especificaciones XPDL (que posteriormente es descrita en esta memoria) para dar cobertura total a BPMN.
- BPMN puede transformarse directamente en BPEL, un lenguaje de orquestación de servicios web que se está consolidando como un estándar.

4.6. Ejemplos de Diagramas en BPMN

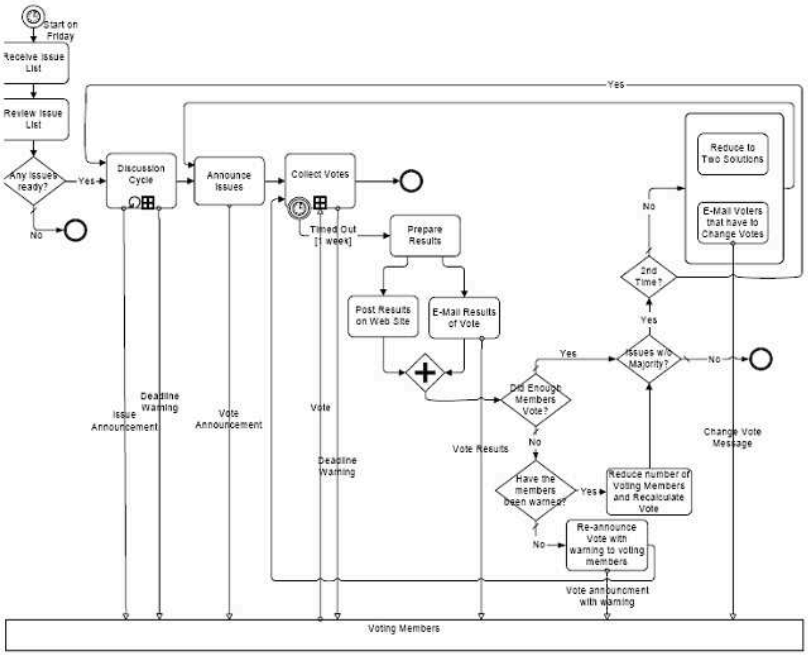


Figura 4.6: Diagrama en BMPN. Proceso de voto electrónico.

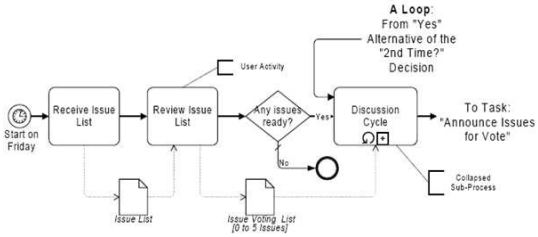


Figura 4.7: Diagrama en BPMN. Visión detallada del proceso de voto electrónico

Capítulo 5

XPDL

5.1. ¿Qué es XPDL?

XPDL (XML Process Definition Language) es un lenguaje de la WfMC (Workflow Management Coallition) que es *"Una organización sin ánimo de lucro para desarrolladores, analistas, consultores e investigadores en el campo de la gestión de procesos de negocio"*.Fue fundada en 1993 y actualmente es miembro de la OMG siendo uno de los participantes que más han influido sobre la especificación de UML 2.0.

Para la WfMC deben existir 5 interfaces funcionales en un proceso o servicio de workflow [18]:

- Definición de procesos e importación/exportación de los mismos.
- Interoperabilidad entre distintos sistemas de workflow.
- Interacción con otros tipos de aplicaciones.
- Interacción con los interfaces de escritorio de los usuarios.
- Sistema para monitorizar los procesos que nos proporcione una serie de métricas que nos faciliten la gestión de los mismos.

XPDL forma parte de la documentación relativa al INTERFACE UNO que da soporte a la definición y a la importación/exportación de procesos, con el objetivo de que, aunque se modele un proceso en una aplicación, este modelo pueda ser usado por otras aplicaciones de modelado y/o por otras aplicaciones que trabajen en el entorno de ejecución.

La versión más reciente de XPDL es la 2.0 y mantiene compatibilidad total con las versiones anteriores. Según los propios creadores de XPDL, dejando muy claro el propósito de su especificación, *"las especificaciones XPDL y BPMN afrontan el mismo problema de modelado desde diferentes perspectivas. XPDL proporciona un formato de fichero XML para ser intercambiado entre aplicaciones. BPMN proporciona una notación gráfica para facilitar la comunicación humana entre usuarios de negocio y usuarios técnicos"*[38]. Y precisamente esta última versión surge para dotar a XPDL de los elementos de BPMN 1.0 que no poseía XPDL 1.0.

La relación entre XPDL y BPMN puede apreciarse con más claridad en la imagen 5.1, donde quedan también patentes los dos principales objetivos de XPDL, la definición de procesos y el intercambio de estas definiciones utilizando XML como mecanismo para realizar estos intercambios.

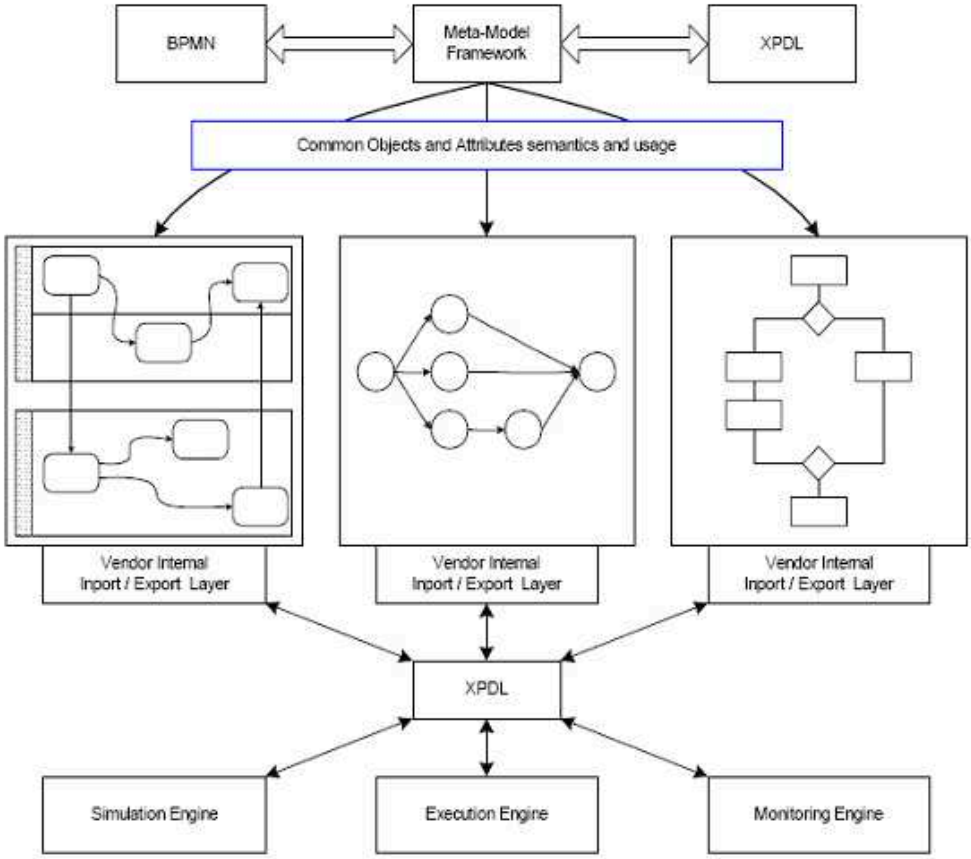


Figura 5.1: Relación BPMN-XPDL

5.2. El metamodelo XPDL

Para llevar a cabo lo que se propone con XPDL la WfMC define un meta-modelo para XPDL que cubre:

- Las entidades de más alto nivel en el dominio de la definición de procesos.
- Atributos de procesos.
- Agrupaciones de diferentes procesos en modelos relacionados.
- Definiciones de datos comunes que puedan ser usados en variedad de modelos.

Para todos estos aspectos tenemos dos meta-modelos principales:

- El metamodelo *Package*, que se encarga de las agrupaciones de procesos, del intercambio de mensajes entre éstos y de las diferentes características que poseen los mismos.
- El metamodelo *Process* que describe las principales entidades que componen un proceso así como los atributos de éstas.

5.2.1. Metamodelo Package

Este metamodelo queda descrito en la figura 5.2

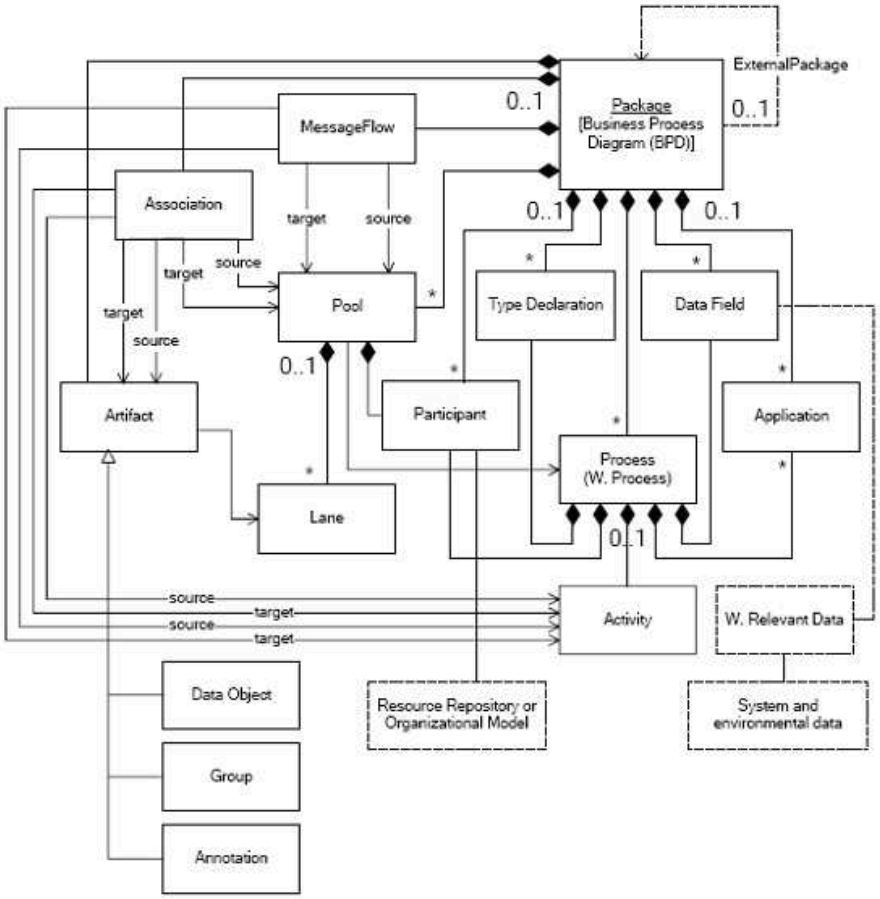


Figura 5.2: Metamodelo Package

5.2.2. Metamodelo Process

Este metamodelo queda descrito en la figura 5.3

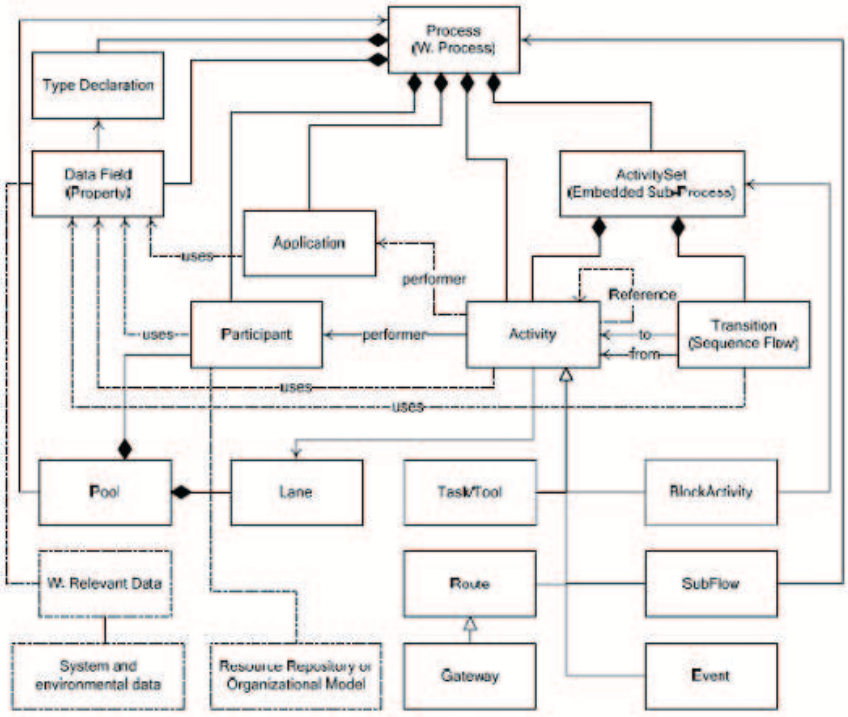


Figura 5.3: Metamodelo Process

5.3. Entidades básicas de los metamodelos

De las entidades que conforman los dos metamodelos vamos a describir brevemente las de más alto nivel, que son precisamente las que tienen relación directa con los elementos de la notación gráfica BPMN:

- **Pool:** Contenedor de actividades y transiciones entre ellas.
- **Lane:** Entidad que nos va a permitir subdividir un Pool; típicamente en relación a los roles participantes.
- **Process Definition:** Proporciona información contextual que se aplica a una serie de entidades a lo largo de un proceso.
- **Actividad:** Trabajo dentro de un proceso que será desempeñado por una combinación de recursos humanos y computacionales.
- **Task:** Unidades de trabajo que componen una actividad.
- **Event:**Cualquier suceso que ocurre mientras se está ejecutando un proceso y que normalmente afecta al flujo del mismo.

- **Transition:** Paso de una actividad a otra cuando se cumplen determinadas condiciones.
- **Participant:** El que realiza una serie de actividades, ya sea un elemento humano o un elemento computacional.
- **Relevant Data:** Los datos que son creados y usados por una instancia de un proceso durante su ejecución.
- **Application:** Elementos computacionales que nos van permitir automatizar, total o parcialmente, una o varias actividades.
- **Artifact:** Elementos del proceso que no pertenecen al conjunto de elementos básicos (actividades, secuencia, flujo de mensajes) y que se relacionan con objetos del flujo(*flow objetcs*) mediante asociaciones. Pueden ser (siguiendo la notación BPMN), un *Data Object*, una *Annotation* o un *Group*.
- **Message Flow:** Flujo de mensajes entre dos participantes y/o procesos que están preparados para enviar y recibir información.
- **Association:** Para asociar un *Artifact* con un objeto de flujo (*Object Flow*) y para mostrar las actividades que se usan para compensar otra actividad.
- **System and Enviromental data:** Datos que son mantenidos por el sistema de workflow o el sistema de entorno local y que son accedidos por las actividades para ser usados de la misma manera que los Relevant Data.

5.4. Herramientas XPDL

Existen multitud de herramientas que nos permiten trabajar con XPDL, las reconocidas por la propia WfMC son las siguientes [9]:

- **ADVANTYS.**
- **Amazonas Workflow.**
- **BOC ADONIS 3.7.**
- **Brein BV offers InProces**
- **CapeVisions**
- **CARNOT Process Engine**
- **CHALEX BPM Framework**
- **The Cubetto Toolset**
- **Enhydra Shark**
- **Enhydra JaWE**
- **Finantix Studio (FXS)**
- **Fuego**

- **Fujitsu Interstage**
- **HOGA.PL**
- **IDS Scheer**
- **Integic e.POWER WorkManager Builder tool**
- **Interstage Business Process Manager Studio**
- **ITP-Commerce**
- **Jenz & Partner GmbH**
- **Flow Designer, Eclaire Group**
- **Metoda S.p.A**
- **Tell-Eureka's**
- **OfficeObjects®WorkFlow**
- **Open Business Engine**
- **Oracle9i Warehouse Builder 9.2**
- **Projekty Bankowe Polsoft**
- **TIBCO® Staffware Process Suite**
- **TIBCO® Business Studio.**
- **Together Workflow Editor**
- **Vignette Process Workflow Modeler**
- **ZAPLET 3 PROCESS BUILDER**

5.5. Conclusiones sobre XPDL

XPDL es una notación para definir e intercambiar modelos de procesos de negocio. A su vez, XPDL puede ser considerado como la notación textual de BPMN, o al revés, BPMN la notación gráfica de XPDL. Eso al menos para la versión de XPDL 2.0 que, como ya dijimos antes, se modificó precisamente para reflejar todos y cada uno de los elementos de BPMN. Por lo tanto XPDL y BPMN son un binomio a tener muy en cuenta dentro de campo del modelado de procesos de negocio, un campo que cada vez está adquiriendo más importancia

Para darle efectividad a esta pareja, y siempre que mantengan compatibilidad (WfMC es miembro de la OMG y BPMI también, aunque nunca se sabe ...) lo ideal sería encontrar una herramienta que nos permita usar ambas especificaciones de la siguiente manera:

- Usar BPMN para modelar de manera gráfica los modelos de procesos de negocio (lo cuál es más amigable tanto para los ingenieros como para los clientes).

- XPDL para guardar los modelos e intercambiarlos entre las diferentes aplicaciones.

Esta asociación toma aún mas valor cuando le añadimos BPEL. BPEL se está convirtiendo en el estándar de facto para la orquestación de Servicios Web, otro de las temáticas punteras dentro del mundo de la ingeniería del software. Pese a las confusiones que pudieran surgir BPEL no es un competidor de XPDL. Ambos tienen propósitos diferentes que van permitir que en determinadas ocasiones puedan complementarse tal y como podemos apreciar en el siguiente en la figura 5.4 obtenido de [30].

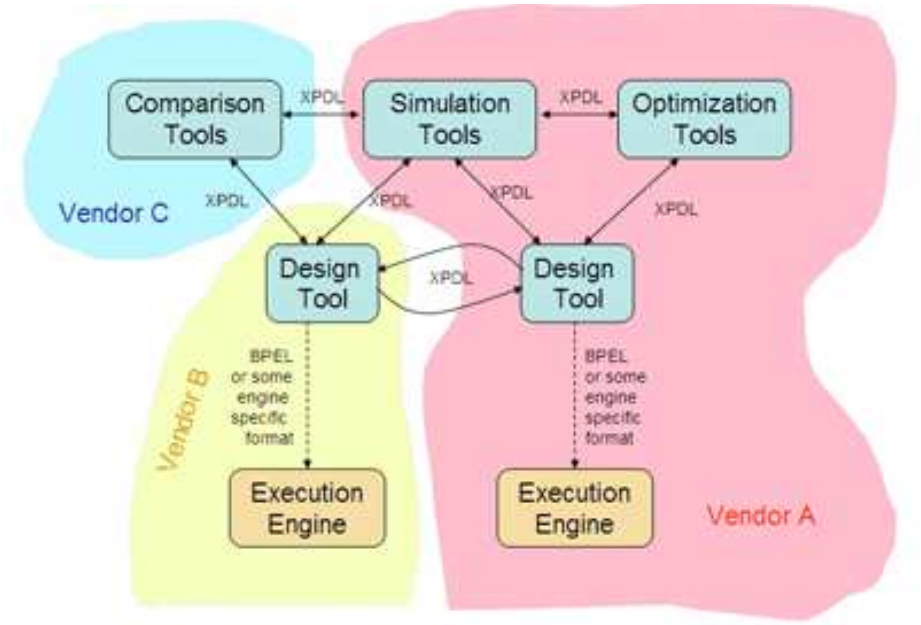


Figura 5.4: Relación entre XPDL y BPEL

5.6. Ejemplos en XPDL

A continuación podemos ver algunos extractos de documentos XML que cumplen con el estándar XPDL y tras ellos su equivalente gráfico en BPMN.

```
<Activities>
    <Activity Id="3" Name="a1"/>
    <Activity Id="4" Name="g1">
        <Route GatewayType="XOR" XORType="Data" MarkerVisible="TRUE"/>
        <TransitionRestrictions>
            <TransitionRestriction>
                <Split Type="XOR">
                    <TransitionRefs>
                        <TransitionRef Id="9"/>
                        <TransitionRef Id="10"/>
                        <TransitionRef Id="11"/>
                    </TransitionRefs>
                </Split>
            </TransitionRestriction>
        </TransitionRestrictions>
    </Activity>
    <Activity Id="5" Name="a2"/>
    <Activity Id="6" Name="a3"/>
    <Activity Id="7" Name="a4"/>
</Activities>
<Transitions>
    <Transition Id="8" Name="" From="3" To="4" FlowType="SequenceFlow"/>
    <Transition Id="9" Name="" From="4" To="5" FlowType="SequenceFlow">
        <Condition Type="CONDITION">a&amp;lt;b</Condition>
    </Transition>
    <Transition Id="10" Name="" From="4" To="6" FlowType="SequenceFlow">
        <Condition Type="CONDITION">a&amp;gt;b</Condition>
    </Transition>
    <Transition Id="11" Name="" From="4" To="7" FlowType="SequenceFlow">
        <Condition Type="OTHERWISE"/>
    </Transition>
</Transitions>
```

Figura 5.5: Ejemplo XPDL nº 1. Una decisión exclusiva.

El texto de la figura 5.5 se corresponde con la figura 5.6:

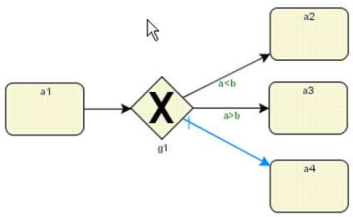


Figura 5.6: Gráfico BPMN correspondiente al ejemplo nº1 en XPDL.

En el segundo ejemplo podemos ver el XPDL correspondiente a una actividad que comienza tras un evento de tipo mensaje:

```
<Activities>
    <Activity Id="3" Name="">
        <Event >
            <StartEvent Trigger="Message"></StartEvent>
        </Event>
    </Activity>
    <Activity Id="4" Name="a1"/>
```

Figura 5.7: Ejemplo XPDL nº2. Actividad que comienza tras un evento de mensaje

El texto de la figura 5.7 se corresponde con la figura 5.8:

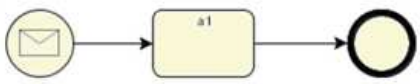


Figura 5.8: Gráfico BPMN correspondiente al ejemplo nº 2 en XPDL

Capítulo 6

jBPM y jPDL

6.1. ¿Qué son jBPM y jPDL?

jBPM (jBoss Business Process Management) es *"un sistema flexible y extensible de administración de workflow que cuenta con un lenguaje de proceso intuitivo para expresar gráficamente procesos de negocio en términos de tareas, estados de espera para comunicación asíncrona, temporizadores, acciones automatizadas"*[22].

Este sistema jBPM ha sido desarrollado para ser utilizado con jBoss, uno de los servidores de aplicaciones más usados. Aunque está centrado en un dominio específico, el desarrollo de aplicaciones web, se ha decidido incluirlo en este documento porque proporciona tanto una notación gráfica para modelar los procesos como una notación basada en XML (jPDL) para almacenar e intercambiar procesos.Nos va permitir, además, generar código de manera automática y realizar un seguimiento gráfico de la evolución de la ejecución del procesos definido.

6.2. Arquitectura de jBPM

jBPM está compuesto por una serie de elementos:

- jbpm-server que es un servidor jBoss preconfigurado para funcionar de manera conjunta con jbpm.
- jbpm-designer que es un plug-in de eclipse que nos permite crear de manera gráfica modelos de procesos de negocio expresados en jBPM.
- jbpm, el componente central, que contiene todas las librerías necesarias y toda la documentación.
- jbpm-bpel. Extensión para BPEL de jBPM.

A su vez el jbmp-server está compuesto por los siguientes elementos:

- El proceso servidor propiamente dicho.
- Un servidor de bases de datos integrado (HSQL).

- Aplicaciones de administración y configuración del servidor jBoss.
- Un programador jBPM para planificar y automatizar la ejecución de tareas.
- Una herramientas para ejecutar ordenes y monitorizar órdenes.

6.3. Modelando gráficamente procesos de negocio con jBPM

Para modelar gráficamente procesos de negocio en jBPM disponemos de una notación gráfica propia que podemos utilizar a través de un plug-in de eclipse de fácil instalación. A través de este plug-in podremos además de modelar gráficamente el proceso:

- Editar el jPDL correspondiente al modelo.
- Modificar todas las propiedades de los elementos gráficos.
- Visualizar y editar el código java que se genera automáticamente a partir del modelo gráfico del proceso de negocio.

Podemos ver la apariencia de ese editor gráfico de procesos de negocio en la figura 6.1.

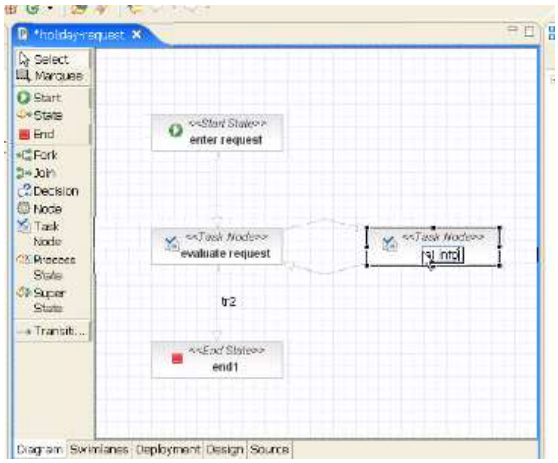


Figura 6.1: Plug-In de Eclipse para jBPM

Las entidades de alto nivel que nos proporciona el editor jBPM son las siguientes:

- **Start:** Para indicar el estado de inicio de nuestro proceso de negocio. Es un estado válido pero que no puede ser ejecutado.
- **End:** Para indicar un estado final para nuestro proceso de negocio.
- **Task-Node:** Para indicar una actividad, dentro del proceso de negocio, que debe ser realizada por el usuario. El proceso mantendrá el token, sin propagar la ejecución, hasta que el usuario no haya realizado su tarea.
- **State:** Para indicar que el sistema espera a que llegue un evento que típicamente suele provenir de un sistema externo no humano. Una vez ha llegado el evento se propaga el testigo de la ejecución. Se diferencia del task-node en que no pone ninguna tarea en la lista de tareas a realizar por los usuarios de la aplicación.
- **Decisión:** Tenemos dos formas para modelar una decisión. La primera de ellas es cuando los propios procesos toman la decisión, es en este caso cuando utilizaremos los nodos Decisión y añadiremos elementos de condición en las transiciones que partan de dicho nodo. La otra forma de modelar la decisión es cuando las decisiones la toma una parte externa. En ese caso usaremos las transiciones partiendo de un nodo State para crear las condiciones de la decisión.
- **Fork:** Pasa el token de ejecución a todas las ramas salientes provocando de esa manera, varias rutas de ejecución concurrentes.
- **Join:** Estado del proceso en el que espera a que le lleguen los tokens de todas las ramas de ejecución entrantes. Una vez le han llegado todos los tokens únicamente propaga un token de salida.
- **Node:** Nodo de características especiales donde podremos definir de manera programática las acciones a llevar a cabo una vez lleguen los tokens. Se suele usar para implementar en java alguna lógica que sea importante para el analista de negocio. El código es invisible en la representación gráfica, lo que es ideal si simplemente estamos analizando el proceso y no la lógica.
- **ProcessState:** jbpm admite la composición de estados mediante este tipo de nodos. El ProcessState es un tipo de nodo o estado que está asociado a una definición de proceso. Cuando la ejecución llega a ese nodo se crea una instancia del proceso al cuál esta asociado y no se continua la ejecución hasta que esa instancia o subproceso ha finalizado.
- **SuperState:** Un SuperState es un grupo de nodos que pueden estar anidados de forma recursiva. Se suelen usar para añadir jerarquía a la definición del proceso.
- **Transition:** Para representar el paso de un nodo origen a un nodo destino. Pueden tener tanto nombre como condiciones asociadas, de tal manera que el token no pasará del estado origen al estado destino si no se cumple dicha condición.

Además de todos estos elementos gráficos desde el plug-in de eclipse podremos añadir, aunque sea de manera textual (mediante jPDL) los siguientes elementos a nuestro modelo de proceso de negocio:

- **Events:** Representan momentos concretos durante la ejecución de un proceso.
- **Actions:** Porciones de código Java que se ejecutarán cuando suceda algún determinado tipo de evento
- **Excepciones:** El mecanismo de excepciones en jbpm sólo atiende las excepciones del código java generado. La ejecución del gráfico en si misma no genera excepciones.
- **Swinlanes:** Nos sirven para indicar la asignación de los distintos actores a las tareas.
- **Timers:** Los timers o temporizadores nos van a permitir realizar acciones o ejecutar transiciones una vez se acabe el tiempo definido para ellos.

6.4. jPDL

jDPL es la notación XML que, de acuerdo a un XLM Schema determinado, nos va a permitir empaquetar todos los archivos asociados a una definición de proceso en un archivo de proceso. Básicamente jPDL se puede considerar como el equivalente XML a lo expresado gráficamente mediente el Plug-In jBPM para Eclipse. Esta correspondencia no es completa ya que existen ciertas entidades que, aunque podemos representarlas de manera textual mediante jPDL, no tienen su equivalente en la notación gráfica.Todos los archivos relacionados con la definición del proceso se empaquetan en un fichero .zip cuyo fichero central es el fichero processdefinition.xml que contiene las etiquetas XML que describen el modelo de acuerdo al XML Schema de jPDL. Estas etiquetas se definen con detalle en [22] y son las siguientes:

- **Process-definition**
- **Node**
- **Common node elements**
- **Start-state**
- **End-state**
- **State**
- **Task-node**
- **Process-State**
- **Super-State**
- **Fork**
- **Join**

- **Decision**
- **Event**
- **Transition**
- **Action**
- **Script**
- **Expresion**
- **Variable**
- **Handler**
- **Timer**
- **Create-timer**
- **Cancel-timer**
- **Task**
- **Swimlane**
- **Assignment**
- **Controller**
- **Sub-Process**
- **Condition**
- **Exception-Handler**

Podemos ver un ejemplo en jPDL de un Process-State en la figura 6.2:

```
<!DOCTYPE process-definition PUBLIC
    "-//jBpm/jBpm Mapping DTD 2.0 beta2//EN"
    "http://jbpm.org/dtd/processdefinition-2.0-beta2.dtd">

<process-definition name="the state process">

  <!-- SWIMLANES -->
  <swimlane name="initiator" />


  <!-- START-STATE -->
  <start-state name="start" swimlane="initiator">
    <transition to="only state" />
  </start-state>


  <!-- NODES -->
  <state name="only state" swimlane="initiator">
    <transition to="end" />
  </state>


  <!-- END-STATE -->
  <end-state name="end" />


</process-definition>
```

Figura 6.2: Ejemplo de Process State en jPDL

6.5. Conclusiones sobre jBPM-jPDL

Ciertamente jBPM+jPDL son dos notaciones estrechamente ligadas a un dominio muy concreto, el desarrollo de aplicaciones web sobre el servidor de aplicaciones jBoss. Partiendo de esa afirmación podríamos pensar en descartarlas si el proceso de negocio a modelar fuera un proceso de desarrollo de aplicaciones y si estuviéramos buscando notaciones y aplicaciones lo más flexibles posible. Sin embargo jBPM posee algunas característica que lo hace muy interesante:

- Se instala como un plug-in para Eclipse, que es uno de los entornos de desarrollo más utilizados.
- Añade un editor gráfico de procesos.
- Genera automáticamente mucho código a partir del diagrama generado de manera gráfica.
- Nos permite comprobar nuestra evolución en el gráfico del proceso mientras estamos ejecutándolo.
- Nos permite la definición e intercambio de definiciones de procesos a través de la notación jPDL, basada en un XML schema.
- Va asociado a JBOSS, una de los servidores de aplicaciones más usados.

Según [7] (uno de los creadores de jBPM) antes de hablar de jPDL debemos tener en cuenta varios problemas en relación a la gestión de procesos de negocio (BPM):

- Los vendedores de herramientas para BPM suelen asegurar que sus herramientas pueden unificar la fase de análisis y de implementación cosa que luego no es cierto.
- Hay una gran falta de integración entre los lenguajes de definición de procesos y los lenguajes de programación de propósito general.
- jPDL fue creado con esos problemas siempre presentes y teniendo en cuenta una máxima le separa de los objetivos de las aplicaciones BPM tradicionales: *Puedes analizar o puedes implementar pero no puedes hacer ambas cosas a la vez.* Para lograr eso jPDL viene con una pequeña serie de nodo básicos pero facilita que un desarrollador pueda escribir código y enlazarlo como la implementación del comportamiento de un nodo del grafo. Con estas características lo que se consigue es que la creación de procesos llegue más allá de la programación visual, acercándolo mucho más al lenguaje de programación de propósito general (en este caso Java).

En lo relativo a la expresividad según sus propios creadores (no se han encontrado estudios independientes) [23] jPDL es capaz de soportar las mayoría de los patrones de workflow para dar soporte a las construcciones más complejas del modelado de procesos pero siempre intentando mantener los detalles más complejos sólo a la vista de los usuarios más expertos.

Lo que no tiene en cuenta este autor es la utilización del modelado de procesos de negocio ya no en el ámbito del desarrollo de aplicaciones web sino en el

ámbito de la integración de aplicaciones. En estos dominios lo que se busca precisamente es minimizar la fase de implementación mediante el aprovechamiento de aplicaciones ya existentes. El modelado de procesos de negocio aplicado al desarrollo de software nos permitirá comprender mejor la integración o la arquitectura basada en servicios, con el añadido de poder transformar ese proceso de negocio en una orquestación BPEL, que es además una de las posibilidades de jBPM y que se está convirtiendo en el estándar de facto para la orquestación de servicios web.

Por todo ello, aunque no elijamos jBPM por ser para un dominio muy específico aporta ideas muy útiles para el modelado de procesos de negocio dentro de un ámbito determinado. De hecho el querer obtener una herramientas de modelado de procesos de negocio de carácter general puede ser un error, el objetivo debe ser un dominio concreto para poder ir automatizando paulatinamente las fases que compongan nuestro proceso, siempre teniendo en cuenta que mientras haya alguna información incompleta no podremos automatizar, ni un proceso completo ni un nodo en particular.

Capítulo 7

ARIS

7.1. ¿Qué es ARIS?

ARIS(Architecture of integrated Information Systems) es un framework de la compañía IDS Scheer para describir estructuras organizativas, procesos y aplicaciones de negocio. Esta compañía es de lejos la compañía líder en el campo de las herramientas BPR (Businees Process Reengineering).

El objetivo principal de este framework es el proceso de negocio de las compañías aunque con el conjunto de herramientas asociadas cubre todas las áreas, independientemente del número de departamentos de la compañía, el tamaño de las mismas o del software disponible. Nos proporciona herramientas para la definición, la configuración, la ejecución y el control de los procesos de negocio. Permite utilizar distintas notaciones dependiendo de la actividad que estemos realizando pero para la descripción y definición de los procesos, que es la parte que nos ocupa principalmente en esta memoria, utiliza diagramas EPC(Event-Driven Process Chain).

En los apartados posteriores vamos a ver una primera aproximación al framework completo para en la conclusión poner más atención en los diagramas EPC y en sus características.

7.2. Arquitectura de ARIS

El modelo de arquitectura de ARIS tiene como objetivo fundamental la integración de sistemas tras un análisis del proceso de negocio [19]. Este proceso se lleva cado mediante una serie de pasos:

- Crear un modelo que contenga los aspectos fundamentales del proceso de negocio.
- Descomponer el modelo en diferentes vistas para reducir su complejidad. De esta manera se pueden utilizar los métodos que mejor se adapten a cada una las vistas.
- Análisis de las distintas vistas por separado.
- Incorporación de las distintas vistas para obtener como resultado un proceso global sin ninguna redundancia.

7.2.1. Vistas descriptivas

La arquitectura del método ARIS y las diferentes vistas que lo componen se pueden apreciarse en la figura 7.1.

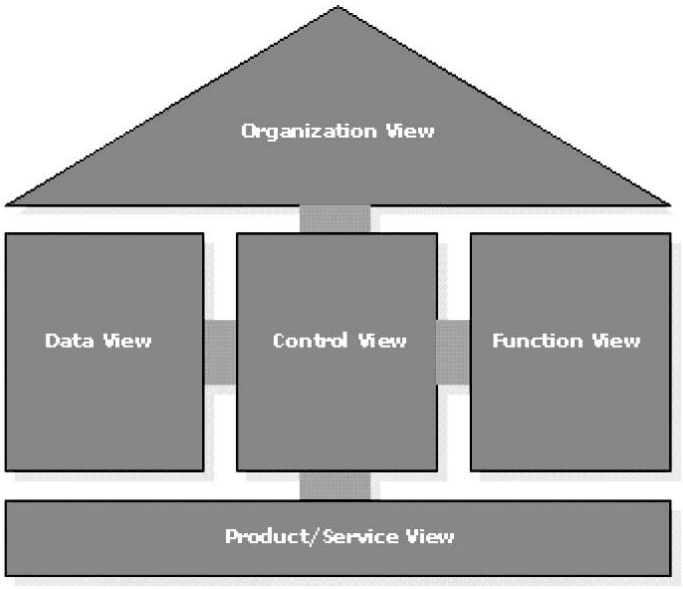


Figura 7.1: ARIS. Vistas de la arquitectura.

- **Product/Service View:** Para representar las relaciones entre las realizaciones de los distintos productos y servicios del proceso que se está modelando.
- **Function View:** Para representar las funciones (procesos a desarrollar) y las relaciones existentes entre ellas.
- **Organization View:** Para representar los usuarios, las unidades organizativas, sus relaciones y sus estructuras.
- **Data View:** Para representar la información que debe ser gestionada por el proceso.
- **Control View:** Es una vista introducida de manera adicional para representar las relaciones entre las diferentes vistas.

7.2.2. Niveles descriptivos

La metodología ARIS describe un ciclo de vida propio. Este ciclo de vida no tiene carácter procedural y lo que hace es establecer distintos niveles de acuerdo a su proximidad a la tecnología. De esta manera tendremos un punto de partida para el proceso, **la descripción del problema del negocio** (operational business problem), que carece de detalles y para la que se utiliza un lenguaje que no es formal. Desde este punto de partida y hasta la implantación del sistema de información que estamos desarrollando pasaremos por una serie de niveles descriptivos cuya relación podemos apreciar en la figura 7.2

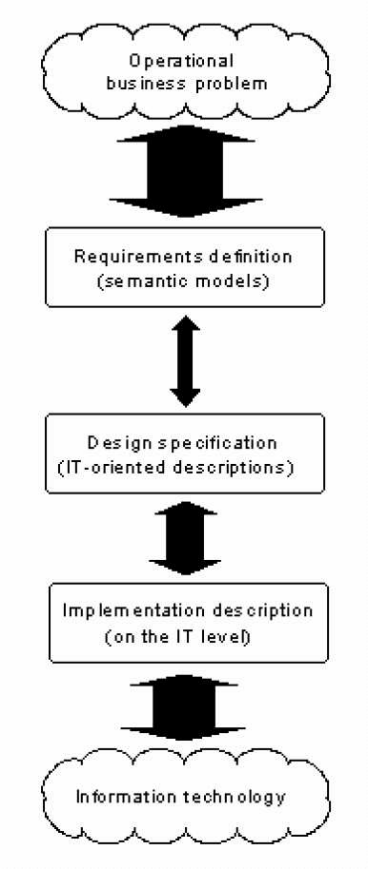


Figura 7.2: ARIS. Niveles Descriptivos y sus relaciones.

- **Definición de requisitos (requirements definitions):** En este nivel describiremos de manera formal la aplicación para que sirva como punto de partida para el traslado de esta información a un nivel tecnológico. Es lo que se denomina el nivel semántico.
- **Especificación del diseño (design specification):** Donde se produce el paso de la descripción del nivel de requisitos a una nueva descripción orientada a las tecnologías de la información.
- **Implementación (implementation):** Concretaremos la descripción tecnológica del nivel anterior a un software y hardware determinado.

La unión de estos niveles descriptivos juntos con las vistas del apartado anterior constituyen el núcleo de la arquitectura ARIS. Cada uno de las vistas se describe desde el punto de vista de los tres niveles, requisitos, diseño e implementación, tal y como queda descrito en la figura 7.3

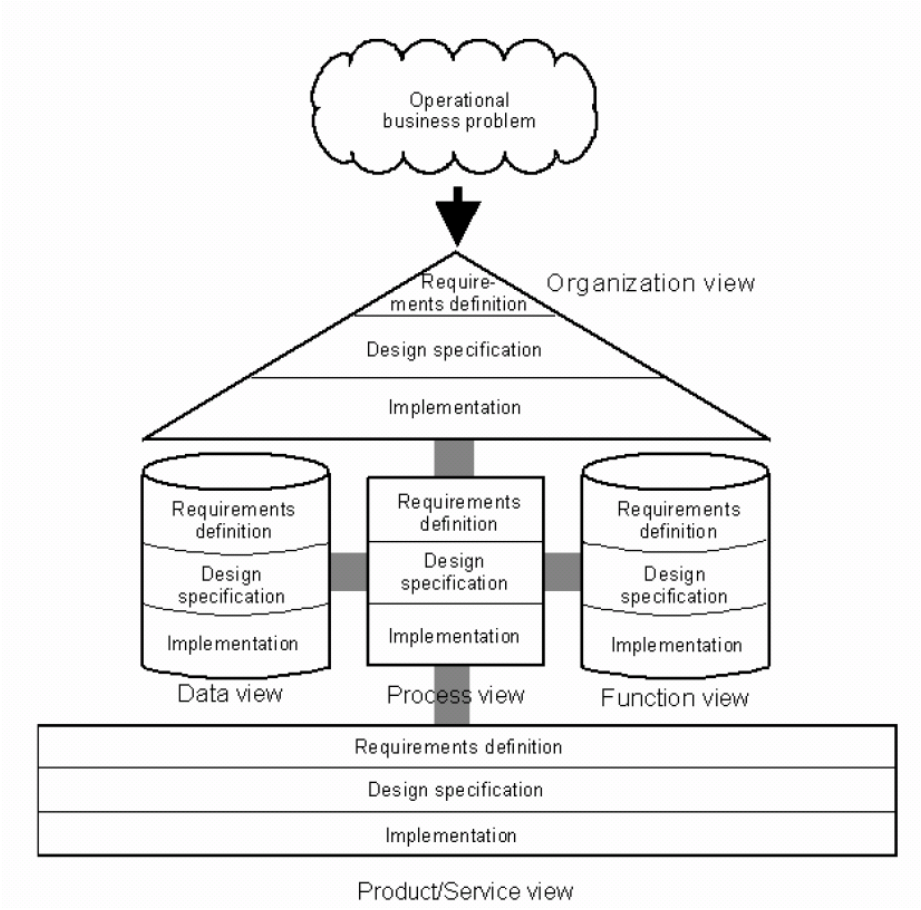


Figura 7.3: ARIS. Relación entre vistas y niveles.

7.3. Function View

7.3.1. Requirements Definition

Debemos de tener en cuenta la definición de función "tarea técnica o actividad para conseguir uno o más objetivos" [19]. Las funciones se representan gráficamente con el símbolo que podemos apreciar en la figura 7.4 y dentro de este nivel se pueden agrupar y representar de distintas formas:

Figura 7.4: ARIS. Símbolo de Función.

- **Mediante árboles de funciones**. Donde las funciones se pueden descomponer jerarquicamente tal y como podemos apreciar en la figura 7.5.

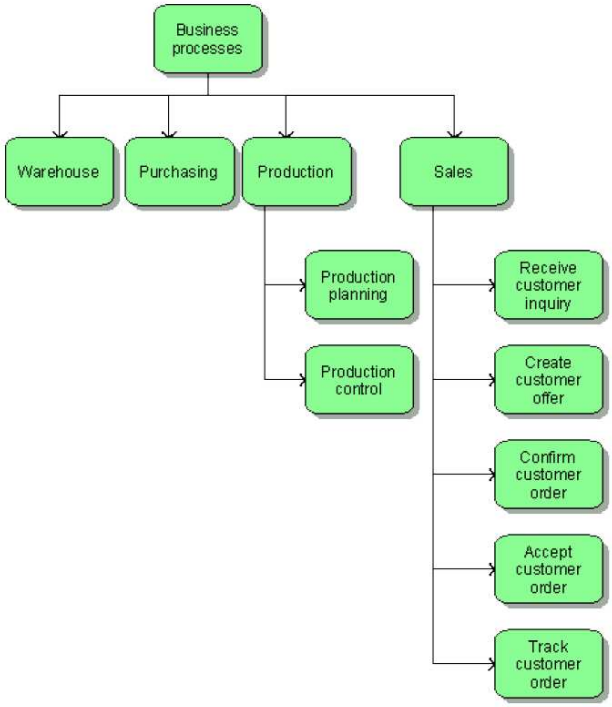


Figura 7.5: ARIS. Árboles de Funciones

- **Mediante diagramas Y**. Las funciones administrativas van en la rama izquierda de la Y mientras que las funciones de la rama derecha son funciones técnicas. Además tenemos un división adicional, las funciones

de planificación van en la parte superior mientras que las funciones de control van en la parte inferior. Podemos ver un ejemplo de este tipo de diagramas en la figura 7.6.

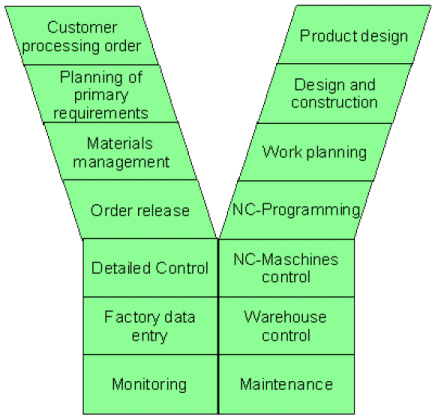


Figura 7.6: ARIS. Diagrama Y.

- **Mediante diagramas compatibles con el modelo de referencia SAP R/3.**
- **Mediante diagramas de objetivos**, que se describen mediante el símbolo de la figura 7.7 y que pueden también relacionarse de manera jerárquica tal y como podemos ver en la figura 7.8

Figura 7.7: ARIS. Objetivo.

7.3.2. Design Specification

La especificación del diseño de la Function View contiene la descripción del tipo de aplicación y la estructura modular de la aplicación entendiendo por modulo un componente de la aplicación que se puede ejecutar de manera independiente. El tipo de aplicación se va a representar gráficamente mediante el símbolo representado en la figura 7.9 y en la figura 7.10 podemos ver una descomposición modular de un tipo de aplicación. Cada una de estas aplicaciones o módulos pueden relacionar además con:

- Funciones provenientes del nivel de requisitos.
- Asignaciones de la aplicación para relacionarla con, por ejemplo, distintos sistemas operativos, interfaces o sistemas de bases de datos.

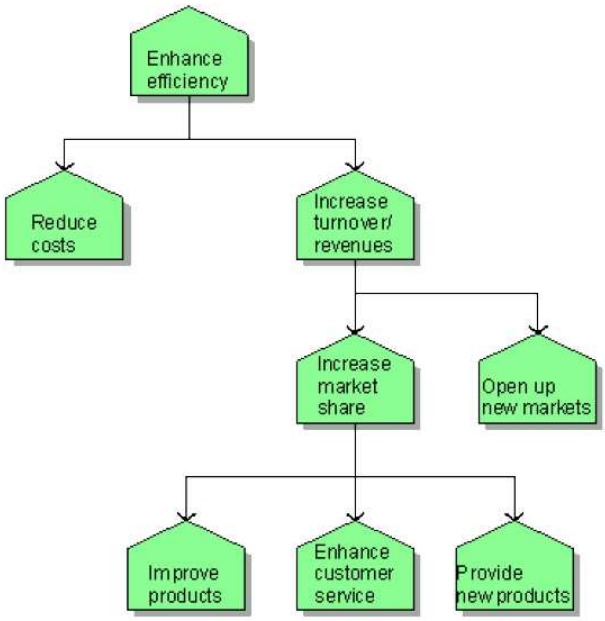


Figura 7.8: ARIS. Árbol de Objetivos.

Figura 7.9: ARIS. Aplicación.

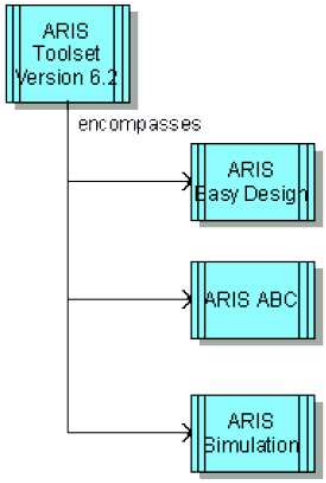


Figura 7.10: ARIS. Descomposición modular de una aplicación.

- El uso de pantallas y listas usadas o generadas por el sistema de información.

7.3.3. Implementation

En el nivel de implementación tendremos aplicaciones que pueden ser identificadas de manera unívoca, mediante por ejemplo, el número de licencia. No hay que confundir este concepto con el concepto de tipo de aplicación del apartado anterior donde nos referíamos a la aplicación como a un conjunto de sistemas con las mismas características tecnológicas. Los diagramas del nivel de implementación serían similares a los del nivel anterior, con mínimos cambios tal y como podemos apreciar en la figura 7.11 donde tenemos la representación de un módulo y una aplicación dentro del nivel en el que nos encontramos.

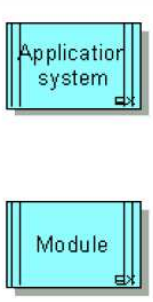


Figura 7.11: ARIS. Aplicación y módulo en el nivel de implementación.

7.4. Data View

7.4.1. Requirements Definition

La definición de requisitos de la vista de datos incluye una descripción del modelo de datos del dominio sobre el cuál vamos a trabajar. Para llevar a cabo esta descripción del modelo de datos en [19] se nos dan varias posibilidades:

- El modelo relacional.
- El modelo relacional extendido.
- SAP SERM.
- IE Data Model.
- SEDAM MODEL.
- DTD.

Además dentro de este nivel ARIS introduce una serie de diagramas adicionales:

- Diagramas de materiales que son entradas y salidas de las funciones del proceso de negocio.

- Estructura del DataWareHouse.
- Las diagramas de jerarquías de autorización para el acceso de los distintos roles a los datos.
- Diagramas de costes.
- Modelos de datos de gestión del proyecto.

7.4.2. Design Specification

En este nivel se transforman las estructuras lógicas definidas en el nivel de requisitos de tal manera que adopten una forma que nos permita posteriormente plasmarlas en un sistema gestor de base de datos concreto.Para ello utilizaremos los diagramas de relaciones y los diagramas de asignaciones de atributos.

El concepto de relación describe la asignación de valores, dentro de un dominio concreto, a los atributos de una ocurrencia de una entidad. Se representa mediante la figura que podemos ver en la figura 7.12 y su relación con la entidad de origen y los atributos que lo componen la podemos ver en la figura 7.13

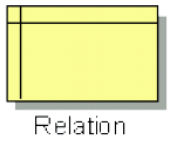


Figura 7.12: ARIS. Relación.

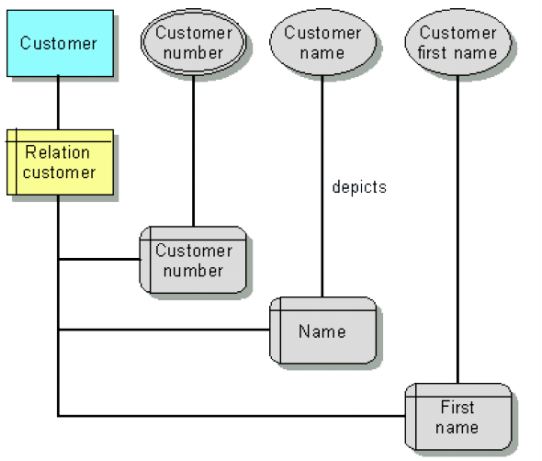


Figura 7.13: ARIS. Relación Entidad-Relación-Atributos.

7.4.3. Implementation

Consiste en la elaboración de un diagrama de tablas concretándolo en un sistema gestor de bases de datos concreto. Tendremos dos entidades gráficas

básicas para dichos diagramas, la tabla y el campo cuya representación la podemos ver en la figura 7.14. Con estas dos entidades básicas podemos representar

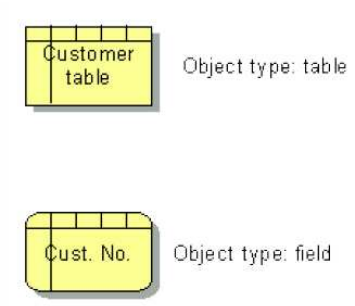


Figura 7.14: ARIS. Tabla y campo de tabla.

varios tipos de diagramas entre ellos el de asignación de campos a su representación en un sistema concreto. Un ejemplo de este tipo de diagrama lo podemos ver en la figura 7.15.

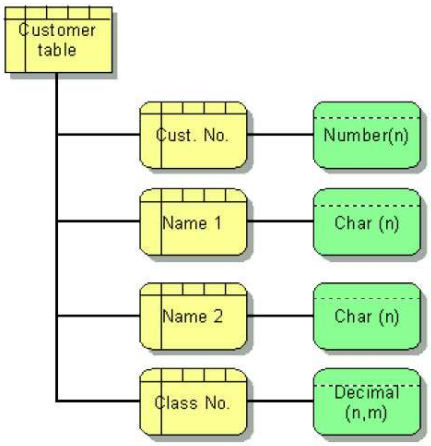


Figura 7.15: ARIS. Asignación de campos a un SGBD concreto.

7.5. Organization View

7.5.1. Requirements Definition

Las compañías son estructuras sociales complejas, sin embargo, hasta hace poco no se ha tenido en cuenta el análisis de la organización como uno de los factores a tener en cuenta en el desarrollo de sistemas de información. Estas estructuras complejas se pueden considerar desde dos puntos de vista, el organizativo que recoge las reglas que rigen la estructura estática de la propia empresa, y el procedural que comprende las reglas que rigen su comportamientos.

En líneas generales el objetivo de toda organización es siempre conseguir una reducción de costes.Tradicionalmente para alcanzar este objetivo se han

seguido dos enfoques, o bien estructurar la compañía de manera funcional o bien estructurar la compañía de acuerdo a las diferentes áreas de productos existentes. En el momento actual las situaciones y el mercado son tan complejos que hacen necesarios la existencia de estructuras flexibles e híbridas que puedan modificarse para adaptarse lo más rápidamente posible tanto a los cambios externos como internos.

El método y la arquitectura ARIS tienen muy presente todo lo anterior y nos proporciona, dentro de este nivel de la vista organizativa, una serie de diagramas para poder representar la estructura de nuestra empresa, sea del tipo que sea.

Organizational Chart

El organizational chart es una forma de representar la estructura organizativa de una empresa mediante unidades organizativas (organizational units) y sus relaciones atendiendo a una serie de criterios como los puestos de trabajos y las relaciones jerárquicas entre ellos, la localización de los mismos o las persona concretas que ocupan dichos puestos.

Unidad organizativa se define como "quienes realizan las tareas que deben ser desarrolladas en orden para alcanzar objetivos de negocio"[19].

Podemos ver un ejemplo de este tipo de diagramas en la figura 7.16 donde la unidades organizativas se representan mediante elipses y además mediante rectángulos podemos representar los componentes de esa unidad y que personas encarnan a dichos componentes.

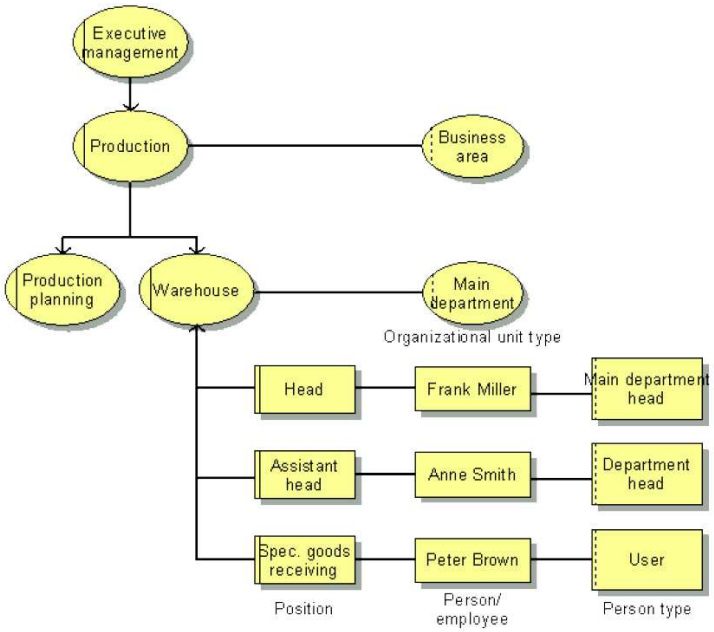


Figura 7.16: ARIS. Organizational Chart.

Shift Calendar

Shift Calendar es un diagrama para representar disponibilidad tanto de recursos como de personas. Es un diagrama multinivel. En el nivel más bajo se representan los Breaks que son periodos de tiempo durante los cuáles no se realiza trabajo y de los que conocemos el comienzo y la duración. En el siguiente nivel se colocan los Shift que son periodos donde sí que se realiza trabajo y de los que también conocemos su comienzo y su duración. Además para estos dos elementos podemos añadir atributos que nos sirvan para representar la periodicidad.

7.5.2. Design Specification- Topología de Red

En este nivel de la vista organizativa podremos representar los diferentes elementos de red que van a permitir las comunicaciones entre los componentes de un Organizational Chart. Para ello dispondremos de elementos como nodos de red, elementos de interconexión y hardware conectado a la red

7.5.3. Implementation

En el nivel de implementación de la vista organizativa distinguimos dos tipos de diagramas:

- **Network Diagrams:** Que es la realización concreta de la topología de red que ha sido descrita en el nivel anterior. Podemos ver un ejemplo en la figura 7.17.
- **Material Flow Modeling-Technical Resources:** Sirve para ilustrar el flujo de materiales en los modelos de procesos. Los tipos de materiales son asignados como entrada o salida a las funciones. Nos van a servir para representar las transformaciones de los materiales.

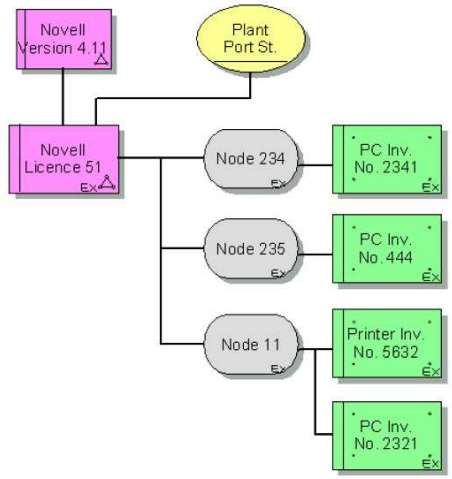


Figura 7.17: ARIS. Network Diagram

7.6. Control View/Process View

La Control View/Process View es la vista que nos va a permitir relacionar las vistas anteriores.

7.6.1. Requirements Definition

Relación entre Function View y Organizational View

El objetivo del diagrama con el que vamos a expresar la relación entre estas dos vistas es asignar las responsabilidades de la realización de las distintas funciones a los distintos componentes que forman un Organizational Chart. La figura 7.18 muestra un ejemplo de este tipo de diagramas.

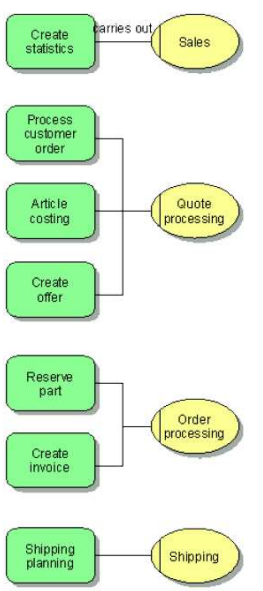


Figura 7.18: ARIS- Ejemplo de relación entre function view y organizational view.

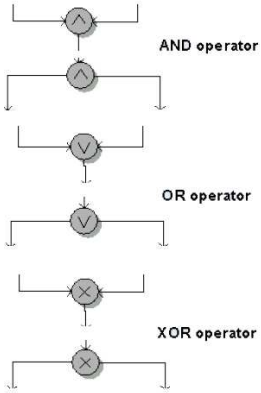
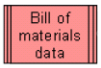
Relación entre Function View y Data View

Para establecer esta relación vamos a utilizar varios tipos de diagramas que posteriormente podremos combinar.

- **EPC(Event Driven-Even Process Diagram)** que nos van a a permitir representar la secuencia de las distintas funciones dentro del proceso de negocio.
- **Function Allocation Diagram:** Para ilustrar la transformación de los datos de entrada en los datos de salida. Podremos hacerlo a distintos niveles de detalle y añadirle los roles o responsabilidades que realizan esta transformación.

- **Information Flow Diagram:** Para representar el flujo de información entre funciones. También se pueden establecer distintos niveles de detalle.
- **Event Diagram:** Para describir la evolución del estado de un objeto a lo largo del tiempo.

Todos estos diagramas poseen una serie de elemento gráficos básicos que podemos ver en la tabla 7.1 y que se pueden combinar para formar diagramas complejos tal y como podemos apreciar en la figura 7.19.

Tipo	Imagen
Función	
Eventos	
Operaciones	
Datos	

Cuadro 7.1: Elementos de EPC

Relación entre Function-Organizational-Data

Juntamos las tres vistas anteriores, nos permitirá obtener una serie de diagramas adicionales:

- **EPC** con elementos de las tres vistas anteriores.
- **Value-Add Chain:** Para enlazar las funciones con el valor añadido que éstas crean para la compañía.
- **Rule Diagramas:** Para expresiones complejas con los operadores OR/AND/XOR:
- **Communication Diagram:** Para agrupar los procesos de acuerdo a las comunicaciones entre unidades organizativas.
- **Classification Diagrams.**

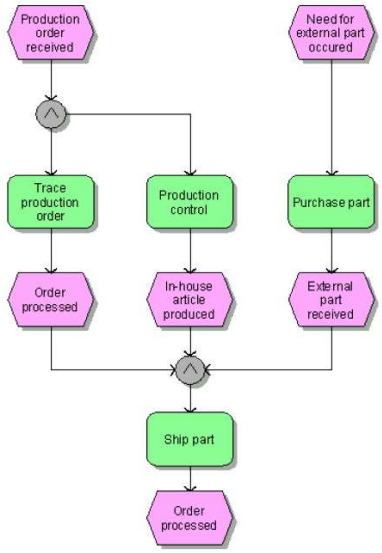


Figura 7.19: ARIS. Ejemplo de diagrama EPC

- **Input/Output Diagrams.**

Además de estos diagramas ARIS para este nivel nos da la posibilidad de utilizar otros muchos diagramas, tal y como queda especificado en [19].

7.6.2. Design Specification

En el nivel de diseño de la vista Control View/Process View tendremos la posibilidad de utilizar los siguientes diagramas:

- **Access Diagrams:** Para expresar las relaciones entre los elementos del nivel de diseño de las diferentes vistas. Podemos ver un ejemplo en la figura 7.20.
- **Program Flow Chart:** Para representar el flujo de control de un programa.
- **Screen Diagram:** Para representar las pantallas de la aplicación a lo largo del proceso de desarrollo.

7.6.3. Implementation- Access Diagram (Physical)

En el nivel de implementación de la vista actual se tienen en cuenta los mismos aspectos que en el nivel de diseño pero concretándolo a un sistema en concreto. De esta manera tendremos por ejemplo diagramas con flujos concretos de datos y diagramas con asignaciones de roles y datos a un hardware concreto.

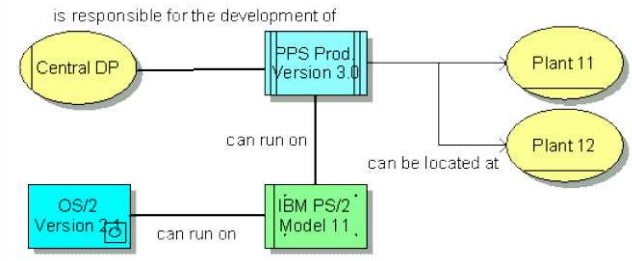


Figura 7.20: ARIS. Ejemplo Access Diagram Nivel de Diseño

7.7. Product/Service View

ARIS proporciona distintas formas (diagramas) de describir los productos y servicios proporcionados por una compañía.

- **Product/service exchange diagram:** Para mapear la creación de productos y servicios y su intercambio dentro de lo límites de la compañia.
- **Product/service tree:** Para representar de manera jerárquica los productos.
- **Product allocation diagram:** Para representar que unidades organizativas proporcionan o usan los productos, que funciones son necesarias para su creación y los ojetivos relacionados con los mismos. Podemos ver un ejemplo de este tipo de diagrama en la figura 7.21.
- **Product tree diagram:** En el nivel de requisitos, se utiliza para analizar la composición de los productos.
- **Product selection matrix diagram:** Para representar la relación entre los roles, los productos generados y los objetivos. Podemos ver un ejemplo de este tipo de diagramas en la figura 7.22.
- **Competition model diagram :** Para representar relaciones de los productos y servicios de una compañía con los clientes que están utilizándolos y con los partners de la propia compañia.

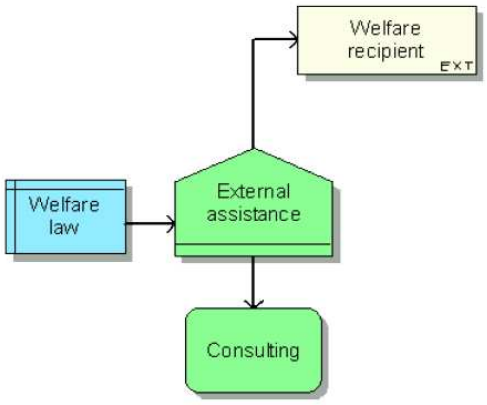


Figura 7.21: ARIS. Ejemplo de Product Allocation Diagram.

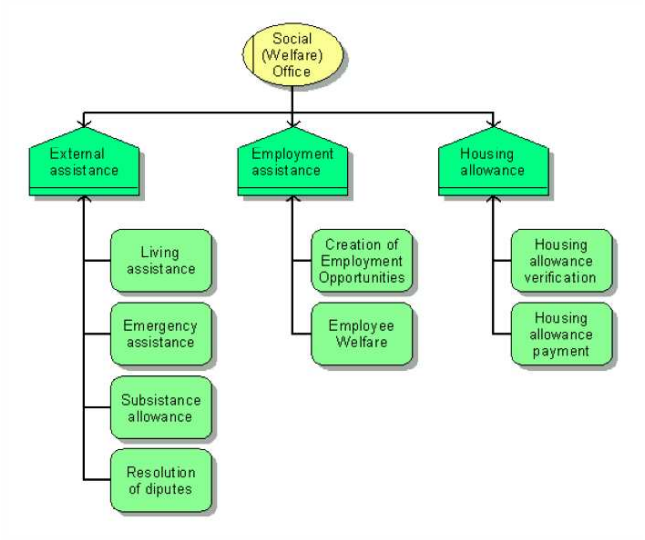


Figura 7.22: ARIS. Ejemplos de Product Selection Matrix Diagram.

7.8. Conclusiones sobre ARIS

ARIS es una metodología compleja (por su gran extensión) cuya notación cubre todos y cada uno de los aspectos relacionados con el desarrollo, optimización, integración e implementación de un sistema de información. Fue creada conjuntamente con un conjunto de herramientas [5](para cubrir todo tipo de actividades) por la compañia IDS Scheer [4]. Actualmente está teniendo gran éxito dentro de campo de los procesos de negocio, pese a lo elevado del precio de todas y cada una de las distintas herramientas.Incluso se está utilizando dentro del ámbito de nuestra propia comunidad autónoma en el campo de la gestión de los servicios de salud, tal y como queda descrito en [24].

Posee ciertas características que hacen que sea, siempre que se pueda pagar el precio de una licencia, una opción a considerar:

- Los modelos de procesos que se crean (Diagramas EPC) son comprensibles para aquellos que no son especialistas en modelado lo que es un requisito fundamental en cualquier proyecto de gestión de procesos de negocio (BPM)[21].
- Permite una vista multinivel de los procesos, con la posibilidad de tener modelos de alto nivel y también niveles con gran nivel de detalle.
- Integración con UML [19].
- Soporte tecnológico para el desarrollo colaborativo.

Frente a todas estas características ventajosas debemos analizar la expresividad de los diagramas EPC que es la notación de la metodología ARIS para expresar los procesos de negocio que, por otra parte, son el objeto de estudio de esta memoria de investigación. Estos diagramas han tenido una gran difusión debido precisamente al éxito de la metodología ARIS y de otras como SAP R/3 pero pese a ello hay que tener en cuenta que, tal y como queda descrito en [31] y en [36] los diagramas EPC pueden ser ambiguos y no tienen una sintaxis bien definida. Este hecho, la ausencia de un semántica formal bien definida, es un gran impedimento para el intercambio de modelos entre herramientas de distintos vendedores (lo que favorece los intereses comerciales de algunas compañías), para el desarrollo de herramientas de análisis de procesos e impide un estudio preciso sobre los patrones de workflow a los que da los que da soporte este tipo de diagramas.

En [31] el autor hace un intento de formalización de los diagramas EPC transformándolas en redes de Petri, una notación totalmente formalizada, pero de difícil comprensión para los no expertos y que que presenta problemas a la hora de expresar patrones que involucran múltiples instancias, sincronización avanzada y cancelación [35].

Capítulo 8

IDEF

8.1. ¿Qué es IDEF?

IDEF (ICAM Definition Languages, siendo ICAM Integrated-Aided Manufactoring) es el resultado de una iniciativa de la *United States Air Force* cuyo objetivo es modelar, gestionar y mejorar procesos de negocio. Fue un proyecto iniciado en los años 70, años en los que convivían multitud de especificaciones y métodos incompatibles entre sí. A lo largo de los años ha ido produciendo diversas metodologías para distintos aspectos relacionados con la creación de sistemas de información. Entre los distintos métodos que ha logrado producir cabe destacar:

- IDEF0 para el modelado de procesos dentro de una organización.
- IDEF1 para el modelado de información.
- IDEF1X para el modelado de datos.
- IDEF2 para el diseño de modelos de simulación.
- IDEF3 para la captura de descripciones de procesos.
- IDEF4 para el diseño orientado a objetos.
- IDEF5 para describir ontologías para la captura de descripciones.

Además de esos métodos citados en la lista anterior existen más métodos descritos por IDEF pero que todavía no han sido desarrollados en profundidad. En el ámbito en el que nos encontramos (notaciones y lenguajes de procesos) nos debemos centrar principalmente en IDEF0 e IDEF3.

IDEF0 *"es una metodología, basada en SADT(Structured Análisis and Design Technique) que pretende representar de manera estructurada y jerárquica las actividades que conforman un sistema o empresa y los objetos o datos que soportan la interacción de esas actividades"*[11].

IDEF3 *"es una metodología para representar el flujo de trabajo de un proceso, así como sus objetos participantes, a partir de la descripción dada por un experto"*[11].

Podría parecer que tanto IDEF0 como IDEF3 pretender cosas parecidas pero existen 3 diferencias fundamentales:

- IDEF0 nos sirve para describir qué hacemos mientras que IDEF3 nos sirve para describir cómo lo hacemos.
- IDEF0 nos da una visión estratégica mientras que IDEF3 nos proporciona detalles de actividades terminales.
- IDEF0 esta pensado para la comunicación con usuarios no técnicos mientras que IDEF3 es para la comunicación con el propietario mismo del proceso.

Y por tanto IDEF0 se aplica principalmente a:

- Comunicar reglas y procesos de negocio.
- Obtener una visión estratégica de un proceso.
- Facilitar un análisis para identificar puntos de mejora.

IDEF3, por el contrario se aplica principalmente a:

- Documentar un proceso actual (a nivel de detalle).
- Identificar y capturar conocimiento crítico sobre un proceso.
- Facilitar un análisis de un proceso en particular.
- Proponer alternativas a un proceso.
- Planear cambios en un proceso

8.2. Elementos de IDEF0

Según [26] *"... en contraste a los procedimientos no formalizados de modelado de procesos (p.ej. en diagramas de flujo, que bastan para descripciones de flujos más sencillos, IDEF0 facilita el trabajo en situaciones de mayor complejidad y mayores exigencias en cuanto al tratamiento"*.IDEF0 nos va a guiar en la descripción de un proceso (función o actividad) que es considerado como la combinación de cinco unidades básicas que interactúan tal y como podemos apreciar en la figura 8.1

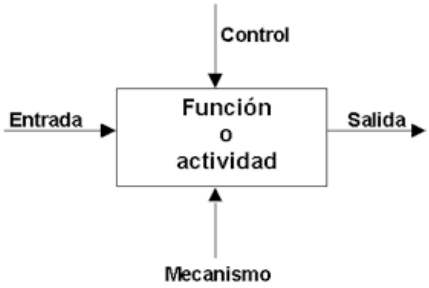


Figura 8.1: Unidades básicas de IDEF0

- **Entradas:** Designan la materia o información que es transformada o consumida por la actividad.

- **Controles:** Objetos que regulan cómo, cuándo y si una actividad se ejecuta o no.
- **Salidas:** Todo aquello que es producido por la actividad o proceso.
- **Mecanismos:** Todos aquellos recursos que son necesarios para llevar a cabo un proceso (personas, herramientas, software, información...).

Además de esos conceptos tenemos más conceptos relacionados con los diagramas IDEF0.

- **Flecha:** Línea que modeliza un traspaso de objetos o información desde una fuente hasta su uso.
- **Etiqueta de flecha:** Nombre que se le asigna a una flecha.
- **Flecha límite:** Flecha con un extremo no conectado a ninguna caja o diagrama.
- **Caja:** Rectángulo que contiene un nombre y es usado para representar una función.
- **Flecha de control:** Flecha que expresa las condiciones requeridas para producir una salida correcta. Se suelen asociar con la parte superior de las cajas.
- **Descomposición:** División de una función o proceso en las funciones que la componen.
- **Bifurcación:** Punto en el que se dividen uno o más segmentos de flecha.

Todos los elementos anteriores, y alguno más que podemos ver en [26], nos sirven para construir los diagramas en IDEF0 que son diagramas jerárquicos que van introduciendo gradualmente más y más nivel de detalle conforme vamos profundizando en la estructura del modelo. Según la profundidad podemos distinguir los siguientes tipos de diagramas:

- **Diagramas Top-Level o diagramas A-0:** Son los diagramas de más alto nivel que nos sirven para representar un proceso de negocio completo y que suelen estar dotados de un nombre muy descriptivo.
- **Diagrama Hijo:** Diagramas en los que se puede descomponer el diagrama A-0 y que a su vez pueden descomponerse en otros procesos de mayor detalle.
- **Diagrama Padre:** Los diagramas padres son aquellos que contienen una o más cajas padre.
- **Texto y glosario:** Asociados a otro tipo de diagramas para otorgar un punto de vista conciso sobre el mismo.
- **Diagramas de exposición:** Diagramas que sólo se usarán cuando se requiera un nivel adicional de conocimientos extra.

Todos estos tipos de diagramas y los elementos citados anteriormente se rigen por una serie de reglas generales:

- Los diagramas de alto nivel deben tener un número de nodo A-n, donde n es igual o mayor que 0.
- El modelo debe contener un diagrama de contexto A-0 que contenga una sola caja.
- El número de caja de la caja del diagrama A-0 debe ser 0.
- Cada caja de un diagrama que no sea de A-0 debe numerarse en su esquina inferior derecha de 1 hasta 6.
- Cada caja que ha sido detallada debe tener la expresión de la referencia a su diagrama hijo escrito bajo la esquina inferior derecha de la caja.
- Las flechas deben dibujarse con trazos horizontales o verticales, nunca diagonales.
- Cada caja debe tener como mínimo una flecha de control y una flecha de salida.
- Una caja puede tener 0 o más flechas de entrada.
- Una caja puede tener 0 o más flechas de mecanismo.
- Una caja puede tener 0 o 1 flechas de llamada.

Podemos apreciar la jerarquización de los diagramas IDEF0 en la figura 8.2:

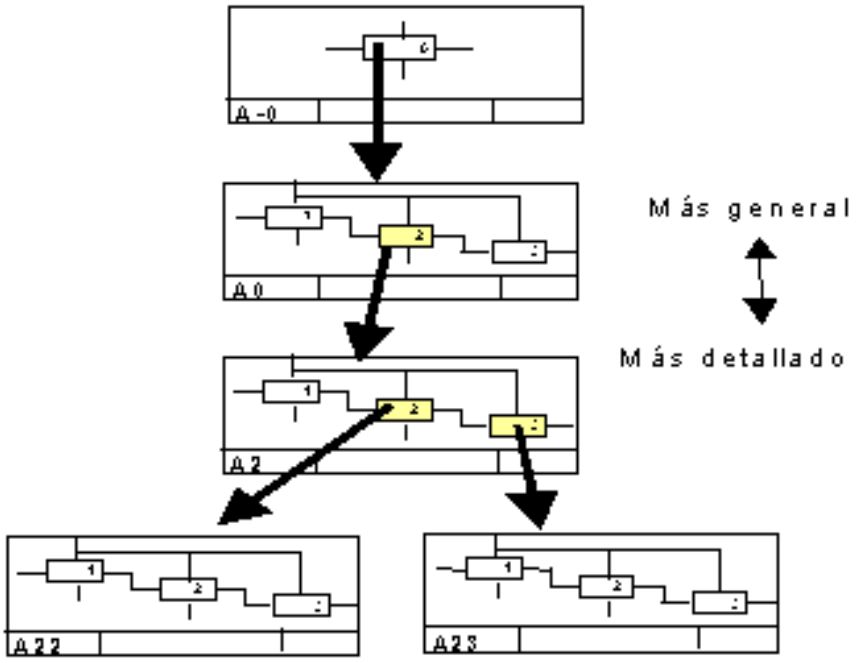


Figura 8.2: Diagrama jerárquico en IDEF0

8.3. Elementos de IDEF3

IDEF3 pretende, tal y como dijimos antes, especificar cómo hago lo que he especificado con IDEF0. Para ello utilizaremos diagramas cuyos elementos principales son los siguientes:

- **Unidad de trabajo**
- **Links**
- **Functions**
- **Referents**

8.3.1. Unidad de trabajo

Las unidades de trabajo representan una actividad, siempre van a tener un identificador único y pueden tener una referencia asociada a una actividad de un diagrama de IDEF0. Podemos ver un ejemplo en la figura 8.3

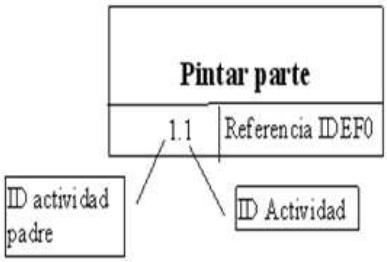


Figura 8.3: Ejemplo de Unidad de Trabajo en IDEF3

8.3.2. Links

Los links sirven para representar relaciones restrictivas entre actividades, son unidireccionales, pueden iniciarse y terminar en cualquier parte de la actividad (caja). Tenemos links de distintos tipos:

- **De precedencia temporal:** El proceso origen debe acabar antes de que el proceso destino pueda comenzar. Se expresan tal y como vemos en la figura 8.4.

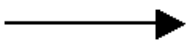


Figura 8.4: IDEF3 Link de Precedencia Temporal

- **De flujo de objetos:** Enfatiza la participación de un objeto entre dos procesos. La semántica es igual a la de precedencia. Se expresan mediante la figura 8.5.

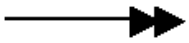


Figura 8.5: IDEF3 Link de Flujo de Ojetos

- **Relacional:** Existencia de una relación entre los procesos relacionados. La semántica no está definida (la define el usuario), tan solo sirve para indicar que el proceso origen empezará antes que el destino. Se representa con la figura 8.6:

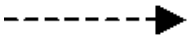


Figura 8.6: IDEF3 Link Relacional

8.3.3. Conexiones

Las conexiones en los diagramas IDEF3 nos van a servir para lo siguiente:

- Representar los puntos en los que un proceso se ramifica en múltiples subprocesos.
- Repressentar los puntos en los cuáles múltiples procesos convergen en un solo proceso.
- Representar la temporalidad (sincronía/asincronía) en el flujo de actividades de un procesos.

Podemos ver un ejemplo de diagrama con conexiones en la figura 8.7.

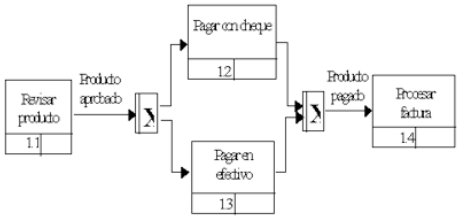


Figura 8.7: IDEF3 Ejemplo de Diagrama con conexiones

Y podemos distinguir a su vez entre dos tipos de ramificaciones:

- **Divergencia (Fan-Out):** Distribuye el flujo del proceso. La activación de una actividad causa la activación de múltiples actividades. Existen distintos tipos que quedan explicados en el siguiente cuadro resumen:

Tipo de Conexión	Significado
AND-Asíncrono	Todas las actividades que suceden a la conexión se iniciarán
AND-Síncrono	Todas las actividades que suceden a la conexión se iniciarán al mismo tiempo.
OR-Asíncrono	Se iniciarán una o más actividades que suceden a la conexión.
OR-Síncrono	Se iniciarán al mismo tiempo una o más actividades que suceden a la conexión.
XOR	Sólo una de las actividades que suceden a la conexión ocurrirá.

Cuadro 8.1: IDEF3. Tipos de divergencia

- **Convergencia (Fan In):** La terminación de múltiples actividades provoca el inicio de una actividad. Existen distintos tipos que quedan explicadas en el siguiente cuadro resumen:

Tipo de Conexión	Significado
AND-Asíncrono	Todas las actividades precedentes deben terminar.
AND-Síncrono	Todas las actividades precedentes deben terminar al mismo tiempo.
OR-Asíncrono	Una o más actividades precedentes terminarán.
OR-Síncrono	Una o más actividades precedentes terminarán al mismo tiempo.
XOR	Exactamente una de las actividades precedentes terminará.

Cuadro 8.2: IDEF3. Tipos de convergencia

8.3.4. Referents

Los Referents son símbolos especiales cuyo objetivo es dirigir la atención del lector a otras partes importantes del modelo. Hay varios tipos:

- **Object:** Para describir la participación de un objeto importante en una actividad.
- **Goto:** Para construir ciclos.
- **UOB(Unit of behavior):** Para incluir una actividad descrita sin implicar un ciclo.
- **Note:** Añadir cualquier información importante a un elemento gráfico.
- **ELAB:** Para documentar de manera detallada algún gráfico.

8.4. Herramientas para IDEF0 e IDEF3

Existen numerosas herramientas que soportan IDEF, entre ellas podemos citar las siguientes[1]:

- **4Keeps**
- **AI0 WIN**
- **BPWin**
- **Business Object Modelling Workbench**
- **CORE**
- **Design IDEF**
- **Design Leverage**
- **IDEF Tools**
- **Popkins Systems Architect**
- **Pro CAP Pro SIM**
- **Process Maker**
- **SA/BPR Professional**
- **Workflow Modeler**

8.5. Conclusiones sobre IDEF

IDEF0 es una técnica sencilla pero poderosa que lleva años utilizándose de manera eficiente en la industria sobre todo en la etapa de ingeniería de procesos de negocio.Nos permite modelar actividades y es independiente del tipo de organización y del tiempo, por lo que hay que tener en cuenta que desde ese punto de vista no es ni un organigrama ni un diagrama de flujo. De igual manera, los modelos IDEF0 tampoco reflejan de manera correcta las interacciones entre los miembros del equipo.Como punto a destacar existe la posibilidad de combinarlo con otras metodologías para agregar secuenciación y sincronización de actividades.

IDEF3 nos va a permitir documentar procesos para su estandarización o para utilizarlos como guía para nuevos integrantes del equipo para así reducir la curva de aprendizaje. Nos permite también capturar la secuencia temporal y la lógica de decisión que afecta al proceso.IDEF3 sirve como herramienta para analizar procesos existentes y para diseñar y probar nuevos procesos antes de iniciar cambios reales que pueden ser muy costosos.

Lo ideal, al menos para afrontar un desarrollo de software como proceso de negocio sería usar de manera conjunta IDEF0 e IDEF3 representando los detalles de implantación así como lo procesos al nivel apropiado en cada momento.

En cuanto a su expresividad, IDEF0 e IDEF3 dan soporte a casi todos los patrones de workflow, tal y como podemos apreciar en [25]. En ese mismo artículo se ponen de manifiesto las deficiencias de IDEF a la hora de reflejar estructuras organizativas y aspectos relacionados con los objetivos y la características cualitativas del proceso.

8.6. Ejemplos sobre IDEF

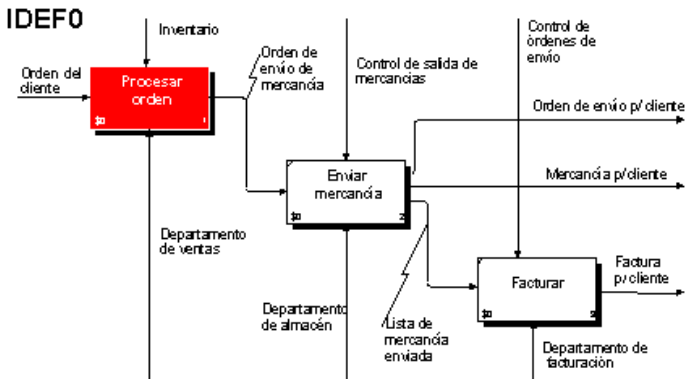


Figura 8.8: Diagrama de Ejemplo en IDEF0

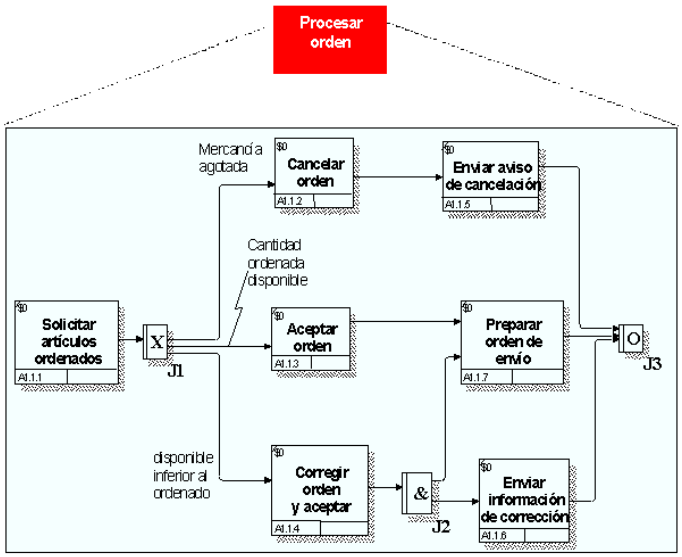


Figura 8.9: Diagrama de Ejemplo en IDEF3

Capítulo 9

Comparativa de las notaciones presentadas

Una vez se han presentado en los capítulos anteriores las diferentes notaciones y lenguajes para el modelado y definición de procesos de negocio en este capítulo vamos a realizar una comparación de las mismas atendiendo a una serie de características.

El primer paso es establecer las características en las que nos vamos a fijar a la hora de realizar la comparativa. Desde nuestro punto de vista una notación, dentro del ámbito de la ingeniería del software, debería poseer idealmente las siguiente características.

- La capacidad de modelar la complejidad de los procesos de negocio, es decir la expresividad. Para ello hemos comprobado el soporte que dan las distintas notaciones a los patrones de workflow.
- La capacidad de representar roles y su asignación a diferentes tareas.
- Capacidad para especificar las características de calidad de los procesos de negocio.
- Capacidad para especificar repositorios de procesos que nos permitan la reutilización de procesos mediante la utilización de conceptos como la variabilidad y la extensibilidad.
- Capacidad para especificar atributos que nos permitan gestionar los procesos (monitorizar, controlar o planificar los mismos).
- Permitir una vista multi-nivel de los procesos para partiendo de descripciones más comprensibles de alto nivel tener la posibilidad de alcanzar niveles con gran cantidad de detalles.
- Ser comprensible para aquellos que no son especialistas en modelado. Esta característica es especialmente útil si con posterioridad se pretende utilizar los modelos para la fase de requisitos.
- Permitir la integración y soporte para otro tipo de notaciones que nos facilitará una mejor interacción entre las herramientas que den soporte a estas notaciones.

- Posibilidad de enlazar de manera directa una actividad con un fragmente de código en un lenguaje de programación.
- La existencia de herramientas para trabajar con ella.

Una vez se han fijado esta serie de características deseables y teniendo en cuenta que en cada uno de los capítulos se han analizado detalladamente cada una de ellas por separado, en la tabla que se muestra a continuación podemos ver un resumen de las relaciones entre las características y las notaciones y lenguajes recogidos en esta memoria de investigación.

Características / **Notación**	**SPEM**	**BPMN**	**XPDL**	**jBPM-jPDL**	**ARIS**	**IDEF**	**AD**
Expresividad	(1)	✓	✓	✓	(1)	✓	✓
Roles	✓	✓	✓	✓	✓	×	✓
Calidad	×	×	×	×	✓	×	×
Reuso	✓	×	×	×	×	×	×
Gestión	✓	×	×	×	✓	×	×
Multinivel	(1)	✓	✓	✓	✓	✓	✓
Comprensible	(1)	✓	×	×	✓	×	×
Integración y soporte	BPMN AD XPDL UML	XPDL	BPMN	×	UML SAPR/3	×	×
Código	×	×	×	✓	×	×	×
Herramientas	×	✓	✓	✓	✓	✓	✓

Cuadro 9.1: Comparativa de notaciones

(1). Se deja a terceras notaciones.

Capítulo 10

Conclusiones y Trabajo Futuro

El mundo del modelado y definición de procesos de negocio es,aún a día de hoy, y centrándonos en el ámbito de la ingeniería del software, un mundo confuso donde conviven multitud de notaciones, lenguajes, grupos de investigación con distintos enfoques, herramientas y compañías con muy distinta actividad y objetivo empresarial. Además de las notaciones y lenguajes descritos en esta memoria, que se han elegido por su difusión y aceptación dentro de la industria, no debemos olvidarnos de citar otros como:

- **Yawl (Yet Another Workflow Language)**
- **Redes de Petri**
- **ebXML**
- **UBL**
- **RosettaNet**
- **Alf**
- **IPSE**
- **Marvel**
- **PMDB**
- **SOCCA**
- **SPADE**
- **TRIAD**
- **POP***
- **SMDSDM**
- **ISO/IEC 10746**

- **EKA**

En relación a la elección de las herramientas, como ya se dijo en anteriores capítulos, no se hizo de acuerdo a factores cualitativos sino que se tuvo en cuenta factores tan apriori poco técnicos como la disponibilidad de las mismas.

Proponer una única notación como resultado de esta memoria sería sin duda un error. Cada una de estas notaciones,lenguajes y herramientas tiene sus propias peculiaridades y características. Antes de cualquier posible elección deberemos analizar detenidamente cuál es el dominio del problema a solucionar. La notación elegida debe dar soporte al problema, al menos permitiéndonos expresar el proceso de trabajo,y proporcionarnos una herramienta que nos permita realizar las actividades necesarias, ya sea únicamente el modelado y la definición o sean otras como la simulación o la automatización de ciertas partes del proceso.

Sin embargo, pese a que hay que hacer un análisis previo del problema, también es cierto que actualmente nos encontramos en un período donde se está produciendo una convergencia hacia una situación en donde los procesos se tienden a expresar de dos formas:

- Mediante una notación gráfica que de soporte a todos los *workflow patterns* [32].
- Mediante un lenguaje expresado en XML que represente los descrito gráficamente y que nos permita el intercambio de las definiciones de procesos entre distintas herramientas.

En esta situación, y tras unos años donde han convivido gran cantidad de notaciones, la industria del modelado de procesos de negocio parece que tiende a centrarse en tres estándares[27]:

- BPMN como notación gráfica para describir los procesos de negocio de un workflow y con el propósito de ser comprensible por todos los usuarios participantes.
- XPDL como formato para almacenar e intercambiar definiciones de procesos permitiendo a una herramienta modelar procesos en BPMN para que otra herramienta pueda leerlos.
- BPEL como lenguaje de ejecución concretando en la tecnología de servicios web lo expresado con alguna de las dos notaciones anteriores.

Una vez realizada la revisión bibliográfica, si seguimos el programa establecido dentro del trabajo de investigación del que surgió esta memoria, "Definición de Modelos de Procesos de Desarrollo Software para Fábricas BDD/SOA.[el] siguiente paso sería estudiar el proceso a modelar y elegir la notación más acorde con dicho proceso. Esta tarea que está directamente relacionada con el trabajo que se está llevando a cabo dentro del grupo de investigación WEBFACTORIES, que pertenece al Departamento de Lenguajes y Sistemas de la Universidad de Sevilla,tiene como requisito previo la descripción del proceso de desarrollo de una fábrica BDD/SOA.

Sin embargo, la revisión bibliográfica que se ha hecho abre bastantes nuevas líneas en la investigación. Al profundizar en el campo del modelado y definición de procesos de negocio han surgida nueva posibilidades interesantes hacia las

que dirigir una posible futura tesis, algunas de las que más interés suscitan al autor de esta memoria son:

- La aplicación de estas técnicas para la elicitación de requisitos para comprobar, mediante estudios de campo, aspectos como que los diagramas BPMN sean más fáciles de entender que los Diagramas de Actividad de UML y que sean comprensibles por todos los usuarios, especialmente los clientes, sirviendo así como elemento de gran utilidad para la comunicación cliente-desarrollador-analista.

- La obtención de casos de uso de manera automática a partir de modelos de procesos con la posibilidad de integrar esta operación dentro de herramientas para la gestión de requisitos como REM. En ese caso será necesario una transformación entre metamodelos, desde el metamodelo que define el proceso hasta el metamodelo de requisitos de la aplicación.

- El estudio y análisis de la eficiencia de procesos de desarrollo ya establecidos. Para ello en un primer paso deberemos modelar los procesos y posteriormente deberemos utilizar herramientas de simulación que nos permitan analizar los procesos con el objetivo de poder detectar los posibles puntos de mejora.

- La integración de modelos de procesos dentro del enfoque de desarrollo MDD/MDD mediante la realización de transformaciones de estos modelos en otros modelos PIM o PSM dentro de un dominio determinado.

- La aplicación de las tecnologías de modelado de procesos a casos reales, como la gestión de procesos sanitarios, dominio en el que ya se está investigando en Andalucía[24] y que requiere el estudio de especificaciones adicionales como HL7v3[3].

El siguiente paso será pues, elegir cuál es la línea a seguir y profundizar en ella para obtener esta vez resultados investigadores.

Apéndice A

Curriculum vitae

Nombre: Juan Diego Pérez Jiménez.

Estudios:

- Ingeniería Superior Informática (Universidad de Zaragoza).
- Master en Imagen de Síntesis y Animación por Ordenador (Universidad de la Islas Baleares).
- Cursos de Doctorado. (Universidad de Sevilla).

Experiencia Profesional:

- Programador de aplicaciones de banca electrónica (C++)(Intercomputer).
- Técnico de automatismos (Plataforma Europa: Grupo Inditex).
- Funcionario de carrera. Profesor de Enseñanza Secundaria especialidad Informática (Junta de Andalucía).

Bibliografía

[1] Herramientas que soportan idef. http://www.isa.its.tudelft.nl/ hommes/toolsub.html.

[2] Herramientas uml. http://uml-directory.omg.org/.

[3] Hl7 v3. http://www.hl7.org/v3ballot/html/welcome/environment/index.htm.

[4] Ids scheer. http://www.ids-scheer.com/.

[5] Ids scheer products. http://www.ids-scheer.com/international/english/products/53961.

[6] Object management group. http://www.omg.org.

[7] Tom Baeyens. About bpm miracles an what you can expect in real life, Julio 2006. http://blogs.jboss.com/blog/tbaeyens/2006/07/05/.

[8] BPMI. *Business Process Notation Specification*. BPMI, Bussiness Process Management Inititive, final adopted 1.0 edition, Febrero 2006.

[9] WfMC. Workflow Management Coalition. Xpdl tools. http://www.wfmc.org/standards/xpdl.htm.

[10] Martin Fowler. *UML Distilled*. Addisson Wesley, 3 edition, 2004.

[11] Jesús Martinez San Germán. Métodos de modelado idef0 e idef3 y uso básico del programa bpwin. Technical report, Comisión Federal de Electricidad, 2003.

[12] OMG. Object Management Group. Business process management initiative. http://www.bpmi.org/.

[13] OMG. Object Management Group. Mda. the architecture of choice for a changing world. http://www.omg.org/mda.

[14] OMG Object Management Group. *Software Process Especification Metamodel Specification*. OMG Object Management Group, request for proposal edition, Noviembre 2004.

[15] OMG Object Management Group. *Software Process Especification Metamodel Specification*. OMG Object Management Group, 1.1 final adopted edition, Enero 2005.

[16] OMG Object Management Group. *Software Process Especification Metamodel Specification*. OMG Object Management Group, 2.0 final adopted edition, Enero 2007.

[17] OMG Object Management Group. *Unified Modeling Language*. OMG Object Management Group, 2.1.1 edition, Febrero 2007.

[18] David Hollingsworth. *The Workflow Reference Model*. Workflow Management Coalition, 1.1 edition, Enero 1995.

[19] IDS Scheer. *Aris Method*, Julio 2004.

[20] Business Process Management Initative(BPMI). Bpmn implementors and quotes, 007. http://www.bpmn.org/BPMN_Supporters.htm.

[21] Curtis B. Kellner M. Over J. Process modelling. communications of the acm, vol 35,pp75-90. Technical report, ACM, 1992.

[22] jBoss Company. *jBOSS jBPM USER GUIDE*. jBoss a division of Red Hat Company, 3.1 edition, Febreo 2006.

[23] jBOSS Inc. jbpm overview, Enero 2006. http://www.jboss.com/products/jbpm/overview.

[24] Rafaeñ Ruiz-Usano Pedro L. González José M. Framiñan, Carlos Parra. Experiencias en la aplicación de modelado de procesos de negocio(bpm) en el sector sanitario. Technical report, Escuela Superior de Ingenieros. Universidad de Sevilla, 2004.

[25] Beate List Birgit Korhnerr. An evaluation of conceptual business process modelling language. Technical report, Vienna University of Technology, 2006.

[26] NIST. *Integrated definition for function modeling (IDEF0)*. National Institute of Standars (NIST), December 1993.

[27] J. Pyke. Is XPDL the silentworkhorse of BPMN? http://www.ebizq.net.

[28] J.D. Pérez A. Durán A. Ruiz. ¿Por qué omg ha elegido bpmn para modelar procesos de negocio si ya existe uml? Technical report, Universidad de Sevilla, 2007.

[29] N. Russel, W. VanderAlst, A. Hofstede, and P. Wohed. On the suitability of uml activity diagrams for business process modelling. In *Proceedings of the Third Asia-PAcific Conference on Conceptual Modelling (APCCM)*, volume 53 of *Conferences in Research y Practice Information Technologies*, pages 195–104. Hobart, 2006.

[30] Keith Swenson. The bpmn-xpdl-bpel value chain, Mayo 2006. http://kswenson.wordpress.com/2006/05/26/bpmn-xpdl-and-bpel/.

[31] W. van der Aalst. Formalization and verification of event-driven process chains. Technical report, Eindhoven University of Technology, 1998.

[32] W. van der Aalst A. Hofstede B. Kiepuszewski A.P. Barros. Workflow patterns. Technical report, Department of Technology Management Eindhoven University of Technology, 2000.

[33] Petia Wohed Wil van der Aalst Marlon Dumas Arthur Hofstede Nick Russel. Patter-based analysis of uml activity diagrams. Technical report, Department of Technology Management Eindhoven University of Technology, 2004.

[34] Petia Wohed Wil van der Aalst Marlons Dumas Arthur Hofstede Nick Russell. Pattern-base analysis of bpmn. Technical report, Department of Technology Management Eindhoven University of Technology, 2005.

[35] A. Hofstede W. van der Alst. Workflow patterns: On the expressive power of (petri-net-based) workflow languages. Technical report, Eindhoven University of Technology, 2002.

[36] E. Klinder W. van der Alst, J. Desel. On the semantics of epcs: A vicious circle. Technical report, Eindhoven University of Technology, 2002.

[37] Stephen A. White. Introduction to bpmn. Technical report, IBM Corporation, 2004.

[38] Workflow Management Coalition. *Process Definition Interface – XML Process Definition Language*, 2.0 edition, Octubre 2005.