Universidad del País Vasco
Euskal Herriko Unibertsitatea

Departamento de Lenguajes y Sistemas Informáticos
Hizkuntza eta Sistema Informatikoak Saila
Facultad de Informática / Informatika Fakultatea

Técnicas Conceptuales en la Gestión de Proyectos Software

Tesis doctoral presentada por
Francisco Javier Ruiz Bertol

Una tesis presentada para optar al título de
Doctor en Informática
por la Universidad del País Vasco / Euskal Herriko Unibertsitatea

Tesis doctoral dirigida por
Dr. Daniel Rodríguez García
Cat. Dr. José Javier Dolado Cosín

San Sebastián, Junio de 2011

Universidad del País Vasco Euskal Herriko Unibertsitatea

AUTORIZACION DEL/LA DIRECTOR/A DE TESIS

PARA SU PRESENTACION

Dr. D. Daniel Rodríguez García con N.I.F. 34.100.366-Z y Cat. Dr. D. José Javier Dolado Cosín con N.I.F. 15.948.508-D como Directores de la Tesis Doctoral: "TÉCNICAS CONCEPTUALES EN LA GESTIÓN DE PROYECTOS SOFTWARE" realizada en el Departamento de Lenguajes y Sistemas Informáticos por el Doctorando Don Francisco Javier Ruíz Bertol, autorizan la presentación de la citada Tesis Doctoral, dado que reúne las condiciones necesarias para su defensa.

En Donostia/San Sebatián, a ___ de Junio de 2011

LOS DIRECTORES DE LA TESIS

Fdo.:Dr. D. Daniel Rodríguez García Fdo.: Cat. Dr. D. José Javier Dolado Cosín

Universidad del País Vasco Euskal Herriko Unibertsitatea

CONFORMIDAD DEL DEPARTAMENTO

El Consejo del Departamento de Lenguajes y Sistemas Informáticos en reunión celebrada el día ____ de _____ de 2011 ha acordado dar la conformidad a la admisión a trámite de presentación de la Tesis Doctoral titulada: "TÉCNICAS CONCEPTUALES EN LA GESTIÓN DE PROYECTOS SOFTWARE" dirigida por el Cat. Dr. D. JOSÉ JAVIER DOLADO COSÍN y por el Dr. D. DANIEL RODRIGUEZ GARCÍA y presentada por Don FRANCISCO JAVIER RUIZ BERTOL ante este Departamento.

En San Sebastián, a ___ de _____de 2011

Vº Bº DIRECTOR/A DEL DEPARTAMENTO SECRETARIO/A DEL DEPARTAMENTO

Fdo.: _____ Fdo.: _____

Universidad del País Vasco Euskal Herriko Unibertsitatea

ACTA DE GRADO DE DOCTOR

ACTA DE DEFENSA DE TESIS DOCTORAL

DOCTORANDO DON. FRANCISCO JAVIER RUIZ BERTOL

TITULO DE LA TESIS: TÉCNICAS CONCEPTUALES EN LA GESTIÓN DE PROYECTOS SOFTWARE

El Tribunal designado por la Subcomisión de Doctorado de la UPV/EHU para calificar la Tesis Doctoral arriba indicada y reunido en el día de la fecha, una vez efectuada la defensa por el doctorando y contestadas las objeciones y/o sugerencias que se le han formulado, ha otorgado por_____la calificación de:

unanimidad ó mayoría

Idioma/s defensa: CASTELLANO_____

En San Sebastián, a ___ de _____de 2011

EL/LA PRESIDENTE/A, EL/LA SECRETARIO/A,

Fdo.: Fdo.:

Dr/a: _____ Dr/a: _____

VOCAL 1º, VOCAL 2º, VOCAL 3º,

Fdo.: Fdo.: Fdo.:

Dr/a: _____Dr/a:_____Dr/a:_____

EL DOCTORANDO,

Fdo.:Francisco Javier Ruíz Bertol

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Departamento de Lenguajes y Sistemas Informáticos
Hizkuntza eta Sistema Informatikoak Saila
Facultad de Informática / Informatika Fakultatea

Técnicas Conceptuales en la Gestión de Proyectos Software

Tesis doctoral presentada por
Francisco Javier Ruiz Bertol

Una tesis presentada para optar al título de
Doctor en Informática
por la Universidad del País Vasco / Euskal Herriko Unibertsitatea

Tesis doctoral dirigida por
Dr. Daniel Rodríguez García
Cat. Dr. José Javier Dolado Cosín

San Sebastián, Junio de 2011

eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Abstract

Faculty of Computer Engineering
Department of Computer Languages and Systems

Conceptual Techniques in Software Project Management

by Francisco Javier Ruiz Bertol

In the current software development environment, in which high productivity and quality levels of both products and processes is required, software process and product management techniques need to be improved. This can be performed through formalisms that allow reliable planning and estimation, better control and tracking of tasks in the development and analysis of historical data.

In this thesis we deal with the theoretical study of representational techniques and knowledge management in project management from two different points of view. On the one hand, studying and improving current representation methods in order to show project information (tasks, resources and plans) in project management tools. On the other hand, exploring the use of ontological techniques that allow us project modelling and reasoning for constraint verification as well as extraction of new knowledge that can facilitate the decision making task of project managers.

As a consequence, two proposals from the representation viewpoint are presented focusing on information not currently available in tools project management tools. Such proposals are: (i) PARMA (Project Activity Representation MAtrix) a model for the matrix representation of project information; and (ii) the Extended Gantt chart that uses the Gantt chart as a basis for showing the assigned resources at

task level and the communication needed between them. From the modelling point of view, an ontology that captures the domain knowledge for project management has been developed. In addition to queries, this ontology can be used for reasoning purposes and in the generation of new knowledge. For such purpose, current semantic technologies have been used, including OWL (*Ontology Web Language*) and SWRL (*Semantic Web Rule Language*).

eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Resumen

Facultad de Informática
Departamento de Lenguajes y Sistemas Informáticos

Técnicas Conceptuales en la Gestión de Proyectos Software

por Francisco Javier Ruiz Bertol

En los actuales contextos del desarrollo del software, en los que se demanda mayor productividad y calidad tanto de los productos como de los procesos, la gestión de los mismos necesita seguir mejorándose mediante formalismos que permitan planificaciones y estimaciones fiables, mejor control y seguimiento de las tareas en curso, y análisis del histórico de datos.

En esta tesis se aborda el estudio teórico de técnicas de representación y del conocimiento en la gestión de proyectos desde dos puntos de vista diferentes: por una parte, estudiando y mejorando los actuales métodos de representación utilizados para mostrar información relativa a proyectos (tareas, recursos, planificación) en las herramientas utilizadas por los gestores de proyecto; por otra parte, explorando técnicas ontológicas que permitan el modelado de proyectos y razonamiento sobre ellos, tanto para el aseguramiento de restricciones como la extracción de nuevo conocimiento implícito que facilite la toma de decisiones a los gestores de proyectos.

En base a este estudio se presentan, desde el punto de vista de la representación, dos propuestas que inciden sobre la información que no queda en la actualidad mostrada en las herramientas para la gestión de proyectos. Estas propuestas son:

PARMA (*Project Activity Representation MAtrix*), una propuesta de modelo para la representación matricial de la información de proyectos; y el diagrama de Gantt extendido, que utiliza la base del diagrama Gantt para mostrar los recursos asignados a nivel de tarea y las necesidades de comunicación entre ellos. Desde el punto de vista del modelado, se desarrolla una ontología que recoge el conocimiento para el dominio de la gestión de proyectos. Esta ontología es utilizada para realizar razonamiento sobre ella para la consulta y generación de nuevo conocimiento utilizando reglas. Para ello se han utilizado tecnologías recientemente desarrolladas para la Web Semántica como son OWL (*Ontology Web Language*) y SWRL (*Semantic Web Rule Language*).

eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Laburpena

Informatika Fakultatea
Hizkuntza eta Sistema Informatikoak Saila

Teknika Kontzeptualak Software Proiektuen Kudeaketan

Egilea: Francisco Javier Ruiz Bertol

Uneko software garapen garaian, produktu eta prozesuen produktibitate eta kalitate handiagoa eskatzen den momentuan, haien kudeaketa hobetu behar da planifikazio eta estimazio fidagarriak ahalbidetzen duten formalismoekin, prozesuan dauden lanen kontrola eta jarraipenekin eta baita datuen analisi historikoaren bidez.

Tesi honetan errepresentazio teknikei eta proiektuen kudeaketari buruzko ikerketa teorikoa burutzen da bi ikuspuntu ezberdinetatik: alde batetik, proiektuen kudeatzaileek erabilitako tresnetan proiektuen informazio erlatiboa erakusten duen garaiko errepresentazio metodoak ikasten eta hobetzen (lanak, errekurtsoak eta planifikazioa). Beste aldetik, proiektuen moldeaketa eta haien arrazonamendua ahalbidetzen duten teknika ontologikoak ikuskatzen, bai murrizketaren ziurtatzerako eta baita proiektuen kudeatzaileen erabakien hartzea errazten duen ezagupena berriaren lorpena ere.

Ikerketa honetan, errepresentazioaren ikuspuntutik, gaur egun proiektu kudeaketan erakusten ez den informazioari buruzko bi proposizio aurkezten dira. Proposizio hauek dira: PARMA (Project Activity Representation Matrix), proiektuen informazioaren errepresentazio matrizialaren ereduaren proposamena; eta Gantt

diagrama luzapena. Azken hau Gantt diagrama erabiltzen du lanaren nibela eta haien arteko komunikazioaren beharrei esleituriko beharrak aurkezteko. Modelaketaren ikuspuntutik, proiektuen kudeaketa menderatzeko ezagupena bereganatzen duen ontologia garatzen da. Ontologia hau bere arrazoibiderako erabiltzen da, bai kontsultarako eta baita arauak erabilita ezagupen berri baten garapenerako ere erabiltzen da. Horretarako Web semantikarako une honetan erabili diren teknologiak erabili dira: OWL (*Ontology Web Language*) eta SWRL (*Semantic Web Rule Language*).

Agradecimientos

Durante del desarrollo de la investigación asociada a esta tesis se han producido cambios en mi vida que me han curtido tanto personalmente como profesionalmente. En el momento de estar preparando la redacción final he observado con optimismo que casi siempre he tratado de rodearme de los mejores. Y entre ellos se encuentran desde compañeros de facultad hasta compañeros de trabajo.

Entre esas personas que me han servido de modelo en esta última etapa debo citar a Fabio Vergari, de la Universidad de Bolognia, en cuya estancia en el *European Software Institute* tuvo la oportunidad de hacer todo lo necesario para finalizar también su tesis, con el resultado de que ahora es doctor. También debo enviar un agradecimiento especial a Sergio Bandinelli, que ha proporcionado lo que estaba en su mano para mandarme a concentrarme para terminar esta tesis.

No debo dejar pasar el agradecimiento a varios compañeros, doctores todos ellos, por el apoyo moral que han ofrecido durante varios meses, para dar ese espaldarazo necesario que le faltaba este trabajo, entre los que se encuentran Xabi Larrucea, Huáscar Espinoza, Carmen Alonso o Estibaliz Delgado. Ellos me han servido de modelo con sus ánimos para que en la última etapa de la escritura pensara que iba a convertirme en *uno de los suyos*. También mi agradecimiento a Joseba Laka, que con respeto, siempre me ha animado a *terminarla de una vez*. Alejandra Ruiz, que con su apoyo y amistad ha puesto su granito de arena en esta tesis. A mis compañeros de despacho, que me han hecho pasar unos buenos momentos con su sentido del humor. Finalmente, y para completar los agradecimientos corporativos, debo congratularme con el apoyo ofrecido anteriormente por el *European Software Institute*, ahora *Tecnalia Research & Innovation* que han puesto los medios y la financiación para que la eclosión tuviera lugar.

Por otra parte, VTT y Artem Katasonov, han proporcionado los medios, el tiempo y el entorno para estar en el estado adecuado para finalizar la escritura de esta tesis en Tampere. Jussi Yliaho, mi compañero de despacho durante mi estancia, me ha ayudado a relajarme entre capítulo y capítulo de tesis, invitándome a ver los sitios más maravillosos de Tampere.

El agradecimiento también es extensible a todos aquellos compañeros de los proyectos IN2GESOFT y ARGO que me apoyaron en su momento, proporcionando un apoyo moral que aún está fresco en mi memoria. Me estoy refiriendo a gente como Mercedes Ruiz, Javier Tuya, Claudio de la Riva, José García Fanjul, Antonia Mas e Isabel Ramos. Seguro que me dejo a alguno, pero sospecho que me lo perdonarán.

A Olli-Pekka Puolitaival y Saara Tölli, por ofrecerme su amistad y apoyo cuando hemos tenido la oportunidad de compartir más de un fin de semana, y a los que les deseo lo mejor para su recien estrenado proyecto de vida.

A Lacramioara Sinziana Dranca le guardo un especial agradecimiento por estar siempre ahí, en contacto, y por haberme ofrecido su amistad, tanto en mi etapa de investigación, como en la etapa docente.

A Alberto Tablado, cuyo modelo durante la carrera fue fundamental para llegar hasta donde hemos llegado, y cuya vida le ha dado otra serie de alegrías dificilmente superables.

Finalmente, pero no menos importante por ello, a mis dos directores de la tesis, Javier Dolado y Daniel Rodríguez, que nunca perdieron la fe en la finalización de esta tesis, y que gracias a ello, hoy es posible el poder presentarla.

"It is a mistake to try to look too far ahead. The chain of destiny can only be grasped one link at a time."

Winston Churchill

Hello everyone. I suppose you think that nothing much is happening at the moment. Well, that's what I want to talk to you all about; endings. Now, endings normally happen at the end. But as we all know, endings are just beginnings. You know, once these things really get started, it's jolly hard to stop them again. However, as we have all come this far, I think, under the circumstances the best solution is that we all just keep going. Let's keep this going in sight, never an ending. Let's remember that this world wants fresh beginnings. I feel here, in this country, and throughout the world, we are crying out for beginnings. We never want to hear this word "endings". I know we all want to sit down. I know you want to take it easy. Of course we're looking for the good. Of course we're looking for the fresh start.

Mike Oldfield, Amarok

Índice general

Índice de figuras

Índice de tablas

Dedicado a mis padres Paco y María

Capítulo 1

Introducción

1.1. Introducción

La gestión de proyectos en un área de estudio que aún tiene un largo recorrido. Aunque se ha avanzado mucho en el desarrollo de la investigación básica en la representación visual y la representación del conocimiento experto en el área de la gestión de proyectos, aún quedan por desarrollar métodos, modelos y herramientas que utilicen todo el potencial de las representaciones semánticas y que éstas puedan plasmarse en una visualización que abstraiga fielmente el proyecto. Sin embargo, en la actualidad tanto las herramientas como los modelos de conocimiento está aún en un estado poco maduro. Desde el punto de vista de las herramientas de visualización, éstas utilizan modelos estáticos y rígidos de proyectos. Desde el punto de vista de la semántica, la limitación viene dada por el limitado uso de las ontologías como motor semántico, incluyendo la ejecución de axiomas y reglas que llevan asociadas las ontologías, y no sólo basándose en la representación del conocimiento.

De hecho, son pocas las incursiones que se han realizado desde el área de conocimiento de la gestión de proyectos para, por una parte, modelar la información para que esta pueda interoperar entre aplicaciones, y por otra, utilizar esta información para adaptarla a la visualización del gestor de proyectos y de las circunstancias.

La mayoría de empresas que desarrollan software y que siguen una metodología ordenada en su gestión de proyectos, encuentran verdaderas dificultades a la hora de gestionar los proyectos software. El conjunto de herramientas debería ser capaz de mostrar no solo la singularidad de las tareas a realizar, sino también el estado actual de desarrollo y la visibilidad del avance del producto. Sin embargo, las herramientas de gestión actuales resultan muy complejas y a la vez, extremadamente sencillas. La complejidad se encuentra en los numerosos conceptos que entran dentro de las valoraciones, incluyendo líneas base, cálculo de costes, balanceo de recursos, etc. Sin embargo, desde el punto de vista de la visualización la información representada está supeditada a los diagramas tradicionales utilizados en la gestión de proyectos. La gran mayoría de los *project managers* que utilizan dichas herramientas no consiguen exprimir la totalidad de la funcionalidad ofrecida. Gran parte de esa limitación se debe a que estas herramientas de gestión son aplicaciones cerradas, donde el modelo de la información del proyecto está proporcionado por el desarrollador de la apliación de gestión.

Un caso especial se produce con el desarrollo software. Las tareas, lejos de poderse definir completamente al inicio del proyecto, estableciendo su alcance, planificación, recursos y producto, generalmente sufre de desviaciones, faltas en la funcionalidad y calidad objetivo, y genericidad en la definición de requisitos y trabajo a realizar. Como resultado de todo ello, se obtiene mucho más frecuentemente de lo esperado, un producto que no cumple con todas las funcionalidades ni la calidad estimadas inicialmente.

En esta de tesis se pretende realizar un estudio de los factores que influyen durante el desarrollo software desde dos puntos de vista bien diferenciados. Por una parte, desde el punto de vista de la visualización de la información asociada a la gestión del proyecto. Las herramientas de visualización de la información proporcionan una percepción de la adecuación de la planificación realizada, tanto en esfuerzo, dimensionamiento, recursos y tiempo. Por otra parte, desde el punto de vista del modelado de la información asociada a la gestión del desarrollo software. Aunque en la representación del conocimiento se ha avanzado mucho durante la última década, es necesario buscar un equilibrio entre los modelos surgidos, la definiciones de los procesos de gestión y la representación de este conocimiento, sin olvidar el potencial que proporcionan los modelos semánticos.

1.2. Motivación

Como Ingenieros de Software tenemos mala fama: a pesar de nuestras capacidades de proporcionar soluciones tecnológicas, nuestras capacidades de gestión y estimación de los proyectos software son limitadas. Muchos de los proyectos terminan incumpliendo plazos, costes y/o funcionalidades. Esta problemática ha sido extensamente estudiada por varios investigadores. Varios de los estudios más interesantes sobre estos problemas se centran sobre los factores especificados en el *Iron Triangle*, denominación forjada por Atkinson [Atk99]. Atkinson establecía que los factores que influyen en una buena gestión se centran en la atención puesta sobre tres criterios: tiempo, coste y calidad (ver figura 1.1). Otro ejemplo de las evidencias de una deficiente gestión se plasman regularmente en el informe CHAOS [Gro09a]. En este informe se analizan una serie de proyectos finalizados, y cuáles son los factores de fallo, derivandose del estudio una serie de factores críticos para el éxito del proyecto. En la actualidad, el debate sobre el que se inició esta tesis sigue vigente, y se continuan proponiendo métricas, metodologías y factores que influyen negativamente sobre los proyectos de caracter tecnológico [BI11, AR06, JI07]. La motivación para la realización de esta tesis es la detección de los elementos conceptuales que influyen negativamente sobre los proyectos TIC desde el punto de vista de la gestión, y en la medida de lo posible estudiar vías de resolución, tratando de acercar el mundo de la gestión de proyectos con el mundo del desarrollo de proyectos software.

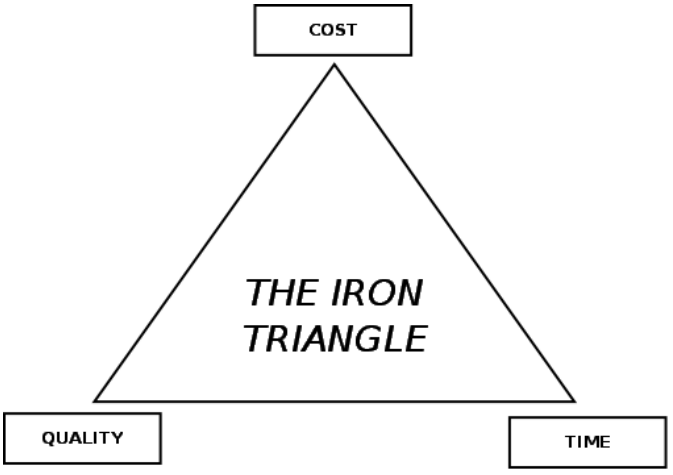


Figura 1.1: Factores de éxito en los proyectos definidos por Atkinson

1.3. Áreas de Conocimiento

Para el desarrollo del trabajo asociado al desarrollo de la tesis se han abarcado varias áreas de conocimiento. Estas áreas de conocimiento investigadas comprenden la mayor parte de temas que se tratan durante la tesis.

- Representación de la Información. La representación de la información, y más concretamente su visualización, proporcionan un campo de investigación que aporta visibilidad a la información representada.
- Procesos de Gestión de Proyectos. La evolución en los últimos años en la definición del modelado de los procesos asociados a la gestión de proyectos permite la apertura de un campo de investigación.
- Procesos de Proyectos Software. El área de conocimiento de los proyectos software es relativamente nueva. La definición de los procesos es un campo de investigación abierto, donde SPEM 2.0 proporciona la refencia formal más exhaustiva.
- Representación del Conocimiento. Desde la propuesta inicial de la Web Semántica a los frentes abiertos relacionados con este tema de investigación, es necesario mapear el conocimiento experto consensuado utilizando las herramientas que proporcionan las ontologías.
- Inteligencia Artificial. Para explotar la capacidad de las representaciones de conocimiento, muchas veces es necesario con algún tipo de inferencia o razonamiento sobre los datos que se representan. En este sentido, en esta tesis se pretende explorar brevemente este área para demostrar la validez de las representaciones que se vayan a desasrrollar.

1.4. Contenido de la Tesis

Esta tesis se divide en un total de ocho capítulos, donde hay dos partes bien diferenciadas: la visualización de la representación de proyectos software, que comprende

los capítulos dos, tres y cuatro; y la representación del conocimeinto de proyectos, que comprende los capítulos del quinto al séptimo.

El capítulo 2 explora los métodos de representación que se utilizan en la actualidad para la visualización de la representación en la gestión de proyectos. El capítulo 3 realiza una incursión sobre la visualización, proponiendo una visualización que solventa ciertas limitaciones observadas en los modelos de representación detallados en el capítulo anterior. El capítulo 4 proporciona una segunda propuesta de visualización de los datos de proyecto, detallando la importancia de la reprentación de los recursos como parte del avance de la ingeniería del software como disciplina.

En el capítulo 5 se realiza un estudio de los principales modelos de representación de los procesos de gestión de proyectos y los lenguajes que permiten plasmar ese modelado. Así mismo se realiza la identificación de los elementos críticos de modelado de la información para la gestión del proyecto. El capítulo 6 desarrolla un modelo semántico para la representación del conocimiento en la gestión de proyectos, utilizando el conocimiento obtenido en el capítulo anterior. A este modelo semántico se le ha denominado *Project Management Ontology*. El capítulo 7 extiende la ontología presentada en el capítulo anterior para permitir que se puedan realizar inferencia sobre los datos. De esta manera, se valida el modelo de representación. Además, se expone cómo, a trevés del razonamiento se genera nuevo conocimiento a partir de la información explicitada a través de la ontología.

Finalmente, el capítulo 8 resume el trabajo de investigación realizado durante todo el periodo de investigación conducente a esta tesis, incluyendo objetivos alcanzados y contribución de esta tesis.

1.5. Objetivos

Entre los objetivos que se desean obtener durante la investigación asociada a esta tesis se encuentran los siguientes:

- Realizar un estado del arte sobre las representaciones visuales para la gestión de proyectos que permita realizar un análisis sobre los factores influyentes durante la planificación y gestión de desarrollos software.
- Evaluar aquellas representaciones del conocimiento y modelos que permitan definir los procesos conducentes a la visualización del proceso de desarrollo software, centrado en los aspectos de gestión y planificación.
- Plantear tanto en representación visual como en representación del conocimiento modelos o herramientas que incidan sobre las debilidades detectadas en los dos puntos anteriores, que permitan el avance de los procesos de gestión en el área de la ingeniería del software.

En resumen, los objetivos establecidos para esta tesis tratan de influir en los problemas existentes para la gestión de proyectos software, teniendo en cuenta los factores críticos que afectan al proceso de desarrollo. Desde mi punto de vista estos elementos influyentes se pueden dar, bien por una deficiente definición de los procesos que toman parte durante el desarrollo software, o bien, una inadecuada percepción de la evolución del desarrollo que impiden una adecuada tangibilidad. Para ello, se investiga tanto en la visualización de la información de gestión del proyecto, como en el modelado de la información, con el objetivo de obtener el conjunto de factores que permitan un avance en las técnicas conceptuales para la planificación y gestión de los desarrollos software.

1.6. Metodología de investigación

Para desarrollar el trabajo formal de esta tesis ha sido necesario seguir una metodología que permitiera alcanzar los objetivos marcados en la tesis. Para ello, se ha utilizado el proceso que se muestra en la figura 1.2. Este proceso se va a llevar a cabo para cada una de las dos partes diferenciadas de esta tesis: (i) el estudio de visualizaciones para la gestión de proyectos software; y (ii) la representación de los procesos que afectan a la gestión de proyectos software.

La metodología de estudio para esta tesis va a estar basada en el siguiente proceso: (i) realizar estado del arte de los temas de investigación sobre los que se va a realizar

la tesis; (ii) estudiar las soluciones que se proponen en los trabajos de investigación ya publicados en las áreas objetivo de esta tesis; (iii) análisis de los resultados que se obtienen de los puntos anteriores, detectando los posibles problemas que existan en relación a la visualización y modelado de la información en los proyectos; (iv) a partir de los problemas que se detecten, se investigarán soluciones que resuelvan parcial o totalmente las limitaciones de las herramientas y modelos actuales; (v) validación de las propuestas presentadas; y (vi) presentación de los resultados en los entornos de publicaciones científicas adecuadas al área de conocimiento, a partir de las propuestas presentadas y las validaciones realizadas.

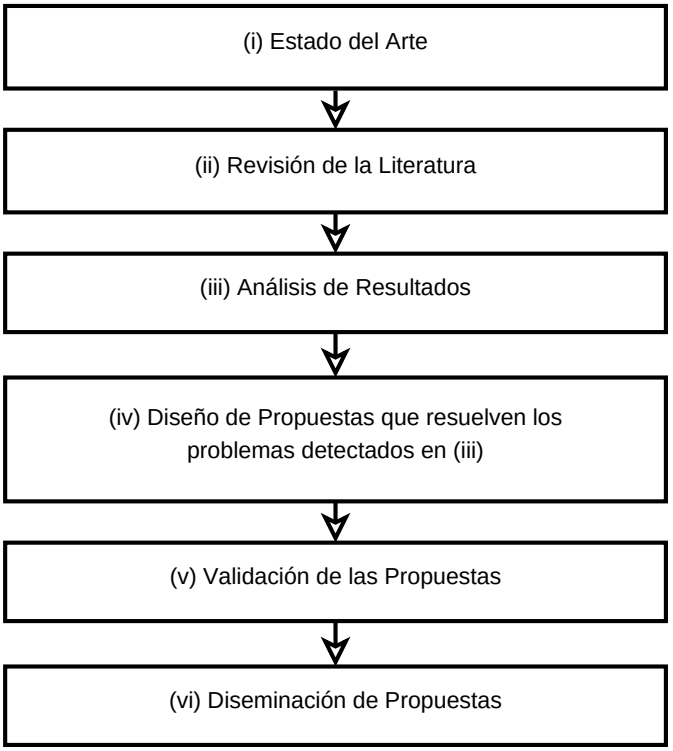


FIGURA 1.2: Proceso seguido para el desarrollo de la tesis

En este sentido hay que destacar que los estudios formales sobre las visualizaciones y la representación de los procesos de gestión eran escasos en el momento de comenzar la tesis, y la investigación sobre los riesgos que se podían producir durante un desarrollo software se veía (y se sigue viendo) como uno de los principales problemas que afectaba a la gestión de proyectos [WF02]. Ya dentro de los proyectos software, se comienza a estudiar en profundidad las diversas áreas de conocimiento que afectaban al desarrollo software, con el objetivo de formalizar la ingeniería del

software como una disciplina bien establecida. Entre los trabajos significativos en esta dirección, el SWEBOK [AMBD04] determina el cuerpo de conocimiento de la ingeniería del software, cuya tercera edición está a punto de publicarse en el momento de escritura de esta tesis.

En esta tesis se pretende alcanzar los objetivos que se han marcado en la sección 1.5 utilizando esta metodología y siendo conscientes de la problemática que se trata de atajar. Hay que valorar que, para el desarrollo de esta tesis, ha sido necesario tocar conceptos relacionados con dos disciplinas tan variopintas como la Ingeniería del Software y la Gestión de Proyectos.

1.7. Contribución

Esta tesis no pretende ser la panacea para resolver los problemas que aquejan a la gestión de proyectos software. Sin embargo, sí que se pretende detectar algunos de los principales problemas desde el punto de vista de la gestión que hacen que los proyectos software no se terminen en tiempo, presupuesto y con las funcionalidades necesarias [Gro09a]. El autor es consciente que el campo de estudio es lo suficiente ámplio y que afecta a varias áreas de conocimiento: captura de requisitos inadecuada, mala gestión de riesgos, recursos inadecuados o mal balanceados, falta de información para la toma de decisiones, limitada aplicación de procesos de validación y verificación, etc. En este sentido, según se expone durante el desarrollo de la tesis, en vez de centrarnos en las fases asociadas al proceso de desarrollo software, nos centramos en el proceso de gestión. El alcance de esta tesis se limita a investigar sobre los factores que influyen de manera definitiva en los proyectos software, como son la visualización de la información para un gestor de proyectos y el modelado de los procesos que se utilizan para representar dicha información. Las propuestas que se presentan tratan de mostrar alternativas que mitiguen la problemática analizada.

Por otra parte, la web semántica y el desarrollo de ontologías se utiliza como herramienta durante esta tesis, no como un foco de estudio. El modelo semántico proporciona una herramienta útil para la representación del conocimiento y para razonamientos sobre la información, como se podrá comprobar en el capítulo 7.

Las ontologías proporcionan el contexto de modelado necesario para definir un proyecto, y los motores de razonamiento permiten utilizar la información modelada para inferir sobre los datos, bien para obtener información a través de consultas o bien para crear nuevo conocimiento a través de reglas.

Capítulo 2

Visualización de la información de los proyectos

2.1. Introducción

La visualización de las representaciones de proyecto constituyen una herramienta esencial para la estructuración, gestión y abstracción de los proyectos software. Su utilidad es incuestionable en la gestión de proyectos debido a que proporcionan una visión general de los componentes a utilizar durante todo el proceso de desarrollo y ayudan a la planificación y representación de las tareas a realizar durante el proyecto. La representación es importante en las etapas iniciales de un proyecto, donde se determina el modelo de planificación, desarrollo, control y gestión del resto del proyecto. En este capítulo se realiza una descripción de los modelos de representación más significativos, y se muestran las conceptualizaciones que mejor proporcionan la visión global del proyecto y que proporcionan una mayor abstracción de los componentes del mismo.

La necesidad de realizar un estudio de los modelos de representación aparece por los problemas que surgen durante el desarrollo de los proyectos software [Bro95]. Las sobrestimaciones de coste y calendario se deben por una parte a que aún no se aplican procesos, disciplinas y prácticas eficientes, pero también a que la

visualización de los componentes software no se realiza de una manera adecuada, y por tanto, es difícil detectar una situación de riesgo durante el proyecto.

2.2. La visualización en la gestión de proyectos

La Gestión de Proyectos ha avanzado mucho desde su concepción inicial hasta la época actuado prácticamente desde la prehistoria. La construcción de la Pirámide de Gizeh en el antiguo Egipto constituye un claro ejemplo de la necesidad de realizar una planificación y gestión para construir grandes obras arquitectónicas. Si bien hasta hace muy pocos años la planificación y la gestión de proyectos ha estado asociada de manera casi exclusiva a la arquitectura, a principios del siglo XX se comenzó a aplicar en otras áreas de conocimiento. Entre ellas, los procesos de fabricación, la realización de experimentos químicos y físicos, la observación y exploración del Universo, la agricultura, la publicación de los primeros periódicos, la logística, etc. son algunos de los ejemplos de la necesidad de realizar una coordinación adecuada para conseguir la meta final, y por lo tanto, es necesaria una secuenciación adecuada de las tareas y un control sobre dichas tareas.

La necesidad de planificar, controlar y dirigir las actividades de un proyecto, viene marcado por el desarrollo del estudio formal de la mejora en la Gestión de Proyectos. La disciplina Gestión de Proyectos incluye áreas de conocimiento como la Gestión de Recursos Humanos y la forma de intercambio de información e interacción delimitada de las tareas realizadas, la distribución de tareas secuenciadas y planificación de dichas tareas, la representación adecuada de la planificación, el estudio de costes de un proyecto, la gestión de riesgos, o la planificación de las compras.

Aunque tradicionalmente la gestión de proyectos ha estado asociada a la arquitectura, en el siglo XX el desarrollo y expansión del ser humano en diversas áreas ha hecho que la Gestión de Proyectos se extendiera a la mayor parte de las disciplinas, como la medicina, la informática, la política, etc. De todas estas disciplinas, la informática es la más reciente, y su exponente más destacable, donde la Gestión de Proyectos se ha convertido en fundamental, es la Ingeniería del Software. La Ingeniería del Software, con apenas cuatro décadas de singladura, constituye uno

de los mayores avances de la humanidad, ya que su extensión ha sido exponencial en la población, principalmente debido al fenómeno *Internet*, y poco a poco se está convirtiendo en uno de los elementos necesarios para el desarrollo de las labores de la vida diaria.

Sin embargo, aunque la aparición de la Ingeniería del Software como disciplina y el estudio formal en la Gestión de Proyectos fue prácticamente simultáneo en los años 60, aún existen aspectos en los que divergen fuertemente. Por una parte, la Ingeniería del Software es un área de conocimiento que es compleja, ya que similares productos con similar funcionalidad pueden desarrollarse usando diferentes metodologías, con diversas tecnologías, con interfaces distintas, con procesos totalmente opuestos, incluyendo o excluyendo temas de calidad, gestión de pruebas, validaciones y verificaciones, con ciclos de vida en el desarrollo muy diferentes, etc. Esto hace que la Ingeniería del Software carezca aún de la madurez suficiente para poder afirmar que se trata de un área de conocimiento bien establecida, con métodos formales, con desarrollos predecibles, con metodologías correctamente utilizadas, con requisitos y funcionalidades que se satisfacen, interfaces usables, conjuntos de pruebas que validen la aplicación, etc.

Por otra parte, la Gestión de Proyectos ha tenido un gran apoyo en las aplicaciones software para su desarrollo, tanto a nivel de modelos gráficos (Gantt, PERT, diagramas de recursos, planificación, trazabilidad de las tareas, planificación y costes), como a nivel de control, por lo que hoy en día puede decirse que es un área de conocimiento casi completamente desarrollada. Pero el gran problema que surge es que la Gestión de Proyectos está proyectada hacia un ámbito genérico, en detrimento de áreas de conocimiento específicas, como por ejemplo, la Ingeniería del Software.

Una de las áreas de conocimiento que no aprovecha adecuadamente las herramientas y técnicas de la gestión de proyectos es precisamente la Ingeniería del Software, debido no sólo a que los requisitos se definen de una manera vaga, las planificaciones son en muchos casos deficientes, no se realizan las pruebas para verificar los requisitos, o surgen defectos en el fase de desarrollo, sino también porque es un área donde el producto no es físicamente tangible, y depende de la tecnología. Aunque

la tangibilidad del software, está siendo solventada en cierta manera por las métricas, aún hay mucho camino que recorrer en el establecimiento de la Ingeniería del Software como un área de conocimiento formalmente establecida.

Como se ha mencionado anteriormente, productos de similares características y con similar funcionalidad pueden haberse desarrollado de maneras completamente diferentes y con gran diferencia de costes, de tiempo y de personal. Medir la bondad de un producto o servicio software pertenece al ámbito de métricas y calidad. En Ingeniería del Software, aunque se dispone de metodologías, ciclos de vida, y procesos estándar, el control de éstos es bastante variable, tanto en el fondo como en la forma. Adicionalmente, al ser un área relativamente nueva, los recursos humanos también forman parte de este descontrol existente. No obstante, se puede decir que parte de la culpa proviene de la propia Gestión del Proyecto, y más concretamente de la representación de proyectos, ya que generalmente se trata de establecer una paridad entre actividades realizadas anteriormente, y no se tiene en cuenta el hecho de que dos productos iguales pueden variar significativamente tanto en costes, tiempo y/o recursos.

Uno de los factores más débiles de la representación deriva precisamente de la sencillez con la que se trata de exponer los datos históricos de los proyectos. Esto lleva a que la síntesis también provoca la pérdida de datos significativos de apoyo a la decisión.

2.3. La representación desde el punto de vista histórico

A comienzos del siglo XX, la Gestión de Proyectos no se consideraba un elemento fundamental en los proyectos, y por lo tanto, ésta era nula en prácticamente la gran mayoría de los ámbitos. Y esto se debía a que la ejecución de los proyectos se llevaba realizando de la misma manera durante siglos, y los plazos, necesidades de materiales, necesidades de personal, el rendimiento del personal y las condiciones de trabajo habían sido eficientes, efectivas y completamente calculadas dentro de las organizaciones. Además, cualquier retraso era inmediatamente solventado

mediante horas extras, hasta extenuar a los trabajadores. Y es que la jornada laboral era lo suficientemente extensa, los proyectos tan sumamente simples, y los pedidos de cliente lo suficientemente previsibles para que surgiera la necesidad de gestionar algo que muchas empresas habían estado realizando durante toda su vida. La revolución industrial del siglo XIX había asentado bien sus bases, sin riesgos, con la tasa de productividad bien ajustada –prácticamente la totalidad del trabajo era manual–, con unas condiciones laborables que separaban claramente la clase obrera de las demás clases sociales, lo que no dejaba lugar a retrasos.

Por entonces, los proyectos eran todo uno: se comenzaba la producción, se ejecutaba todo el trabajo a desempeñar y se obtenía el producto resultado. Sin embargo, los tiempos estaban cambiando, hacia una nueva sociedad que exigía cada vez más, con mayor calidad, con la necesidad de planificación, realizando proyectos de una manera más eficiente. Los principales hitos de esta época, históricamente hablando fueron el nacimiento de una nueva sociedad, el establecimiento de Estados Unidos como potencia mundial, y el estallido de la I Guerra Mundial.

2.3.1. El Diagrama Gantt

En la primera década del siglo XX, Frederic Taylor (1856-1915) había comenzado a investigar de una manera científica la manera de mejorar la productividad. La conclusión a la que llegó Taylor es que un proyecto no era una tarea monolítica, sino un conjunto de actividades manejables relacionadas entre sí, y que para llegar a mejorar la productividad, se debía actuar individualmente sobre estas unidades de trabajo para que éstas se ejecutaran más eficientemente. La solución que dio Taylor para lograr que las tareas se ejecutaran más eficientemente, era emplear a los trabajadores una mayor cantidad de tiempo extra.

Durante esa época, Henry L. Gantt (1861-1919), colaborador durante muchos años de Taylor, continuó esta investigación, utilizándola en la empresa Frankford Arsenal para la construcción de barcos de guerra para la I Guerra Mundial. Fue en este momento cuando desarrolló el diagrama de barras o diagrama Gantt, dividiendo los proyectos de construcción de barcos en unidades más pequeñas, denominadas tareas, y estableciendo las primeras estimaciones de duración de éstas [Leg09].

El diagrama Gantt representa las actividades de un proyecto en una barra horizontal dispuesta sobre un calendario o línea temporal [McD01]. Las actividades a ejecutar se obtienen de la descomposición que se realice del proyecto, en base al nivel definido de división necesario para que dichas actividades sean manejables tanto para la Estudio sobre métodos de representación para la gestión de proyectos software planificación, como para el seguimiento y/o la comunicación. Para obtener esta descomposición en actividades es recomendable utilizar el Work Breakdown Structure (WBS), del que hablaremos posteriormente. La otra opción –WBS como tal no se desarrolló hasta los años 60– fue simplemente dividir el trabajo en tareas, y más concretamente la subdivisión se basaba en las tareas que realizaba cada trabajador. Las actividades obtenidas de la descomposición, se agrupaban posteriormente en el diagrama por unidades lógicas del proyecto, como fases o procesos. Para la representación de las actividades en el diagrama, no existe una notación estándar, y cada desarrollador puede establece una notación diferente, pero siempre siguiendo unas guías determinadas por el diagrama Gantt. Así, observando estas guías, se puede observar en la tabla 2.1 la notación estándar *"de facto"*.

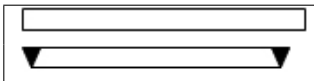
Símbolo	**Descripción**
	Actividad Grupo de Actividades Estado de una Actividad

Tabla 2.1: Notación estándar utilizada para los diagramas Gantt.

En esta notación se puede observar que la actividad se define como una barra rectangular, situada en el diagrama Gantt utilizando una escala temporal. En este diagrama también se pueden definir agrupaciones de tareas o actividades, representadas por una barra delimitadas por triángulos invertidos. La barras que representan grupos de actividades tienen la extensión correspondiente desde que se inicia la primera tarea en el tiempo de dicho grupo de actividades, hasta que finaliza la última tarea planificada. En los diagramas actuales, cada vez más es necesario realizar una monitorización de las actividades realizadas, en comparación con la planificación realizada. En este caso es cuando se utiliza la barra del estado de las actividades, que permite predecir el progreso actual de la actividad frente a

su planificación, y se representa como una actividad, pero indicando en el interior de dicha barra el progreso alcanzado en dicha actividad. Más adelante estudiaremos cómo se representa e interpreta la monitorización de un proyecto utilizando el diagrama Gantt.

Este método ha sido utilizado durante la gran cantidad de proyectos desde entonces, variando muy poco desde aquella primera propuesta. A inicios de los años 90 surgió la necesidad de añadir las dependencias entre tareas, para intentar establecer la planificación óptima de los proyectos. En los últimos años, ha surgido una nueva extensión al modelo Gantt, basado en la necesidad creciente de los proyectos en realizar seguimiento.

El diagrama Gantt surgió de la necesidad en los comienzos del siglo XX de realizar una división de los trabajos, e intentar actuar sobre la eficiencia de las tareas individuales, para lograr mejores resultados y productos, pero también para prever la duración de los proyectos, y así tener mejor controlados tanto los tiempos de entrega como el coste de los proyectos. Partiendo del hecho de que nadie anteriormente había aplicado una planificación de estas características con éxito, el diagrama Gantt surgió como la herramienta definitiva para la planificación, utilizándose intensivamente aún hoy, donde el efecto gráfico aún sirve de herramienta de comunicación entre los participantes del proyecto (*stakeholders*), tanto internos como externos.

Un diagrama Gantt consiste en una tabla bidimensional donde se listan las tareas a realizar en el proyecto, los hitos, y las dependencias que existen entre las tareas y con los hitos, junto con la representación en forma de barras (en el caso de las tareas) o rombos (en el caso de hitos) escaladas en el tiempo. Se puede observar un ejemplo en la figura 2.1.

En este diagrama, las tareas se pueden agrupar por fases, o cualquier otra categorización que se pueda determinar para la agrupación de tareas. La organización de las tareas en el diagrama suele ser la siguiente, en el orden descrito:

1. Ordenación por fases o grupo de actividades relacionadas.
2. Ordenación por fecha de comienzo de las actividades.

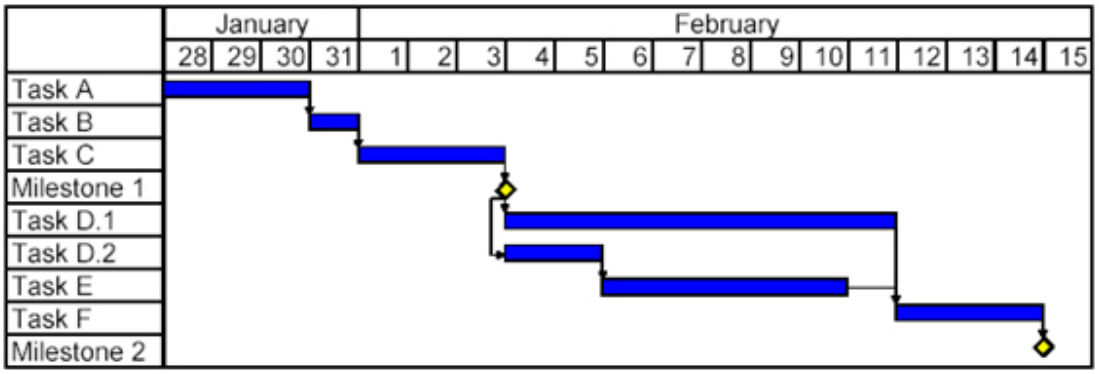


FIGURA 2.1: Un diagrama Gantt básico, incluyendo actividades e hitos.

De esta manera, se obtiene un diagrama donde las actividades se disponen en estructura de escalera dentro del diagrama, de tal manera que se pueden observar cuál va a ser el calendario planificado del proyecto, donde se listan todas las actividades a realizar en un calendario ideal, tal como se puede observar en la figura 2.1.

Inicialmente, el diagrama Gantt solamente contenía las barras de actividad. Posteriormente, se incluyeron los hitos –proveniente del diagrama de hitos–, las dependencias y el seguimiento. El diagrama de hitos lo explicaremos en la siguiente sección. Respecto a las dependencias, surgida a raíz de las necesidades de incremento de la productividad de los años 90, explican la relación entre actividades que pueden existir entre el comienzo y el fin de actividades. En la tabla 2.2 se pueden observar los distintos tipos de dependencias que pueden darse.

La más común de las dependencias que se suelen dar es *Finish-to-Start*, que define una secuenciación entre las tareas, y suele indicar que el producto resultado de la primera tarea (informe, código fuente, aplicación o entregrable) es necesaria para realizar la siguiente tarea. Las demás dependencias de la tabla no se suelen utilizar con tanta asiduidad en los diagramas Gantt, debido a que la definición de estas dependencias implica un mayor control sobre lo que determinan las tareas, los recursos que utilizan, etc. Esta tabla proporciona el conjunto básico y lógico de dependencias, utilizado en otros sistemas de representación que observaremos posteriormente.

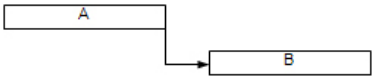
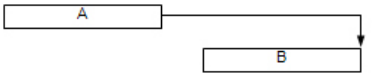
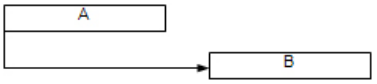
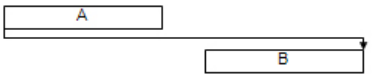
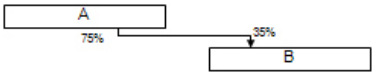
Denominación	**Representación Gráfica**	**Descripción**
Finish-to-Start (FS)		No se puede comenzar la tarea B hasta que finalice la tarea A
Finish-to-Finish (FF)		La tarea B no puede finalizar hasta que finalice la tarea A
Start-to-Start (SS)		La tarea B no puede comenzar antes de que comience la tarea A
Start-to-Finish (SF)		La tarea B no puede finalizar hasta que comience la tarea A
Percent Complete (PC)		La tarea B no puede sobrepasar un cierto porcentaje hasta que un cierto porcentaje de la tarea A se haya completado

Tabla 2.2: Posibles dependencias que se pueden dar en un diagrama Gantt.

Por otra parte, se puede determinar el seguimiento sobre un proyecto utilizando líneas de monitorización bajo la planificación realizada, determinando el grado de finalización en un momento dado, como se puede observar en la figura 2.2.

De esta manera, el Diagrama Gantt es un sistema de representación temporal de la planificación de un proyecto de forma visual, de tal manera, que actividades y procesos están completamente determinados dentro de esta herramienta de planificación. La disposición en forma de líneas durante el gráfico ayuda a determinar dependencias y relaciones entre tareas o actividades, pudiéndose agrupar éstas. Las extensiones han proporcionado las necesidades que fueron surgiendo desde su creación, como las propias dependencias, la realización del seguimiento, o la inclusión del diagrama de hitos.

El diagrama Gantt es el método de representación por excelencia en la Gestión de Proyectos, siéndose utilizado mayoritariamente en la totalidad de software de gestión de proyectos. Este método de representación es genérico y aplicable a cualquier ámbito, y de hecho constituye la herramienta gráfica de Gestión de Proyectos más práctica y potente de cara al cliente y entre los propios participantes del proyecto.

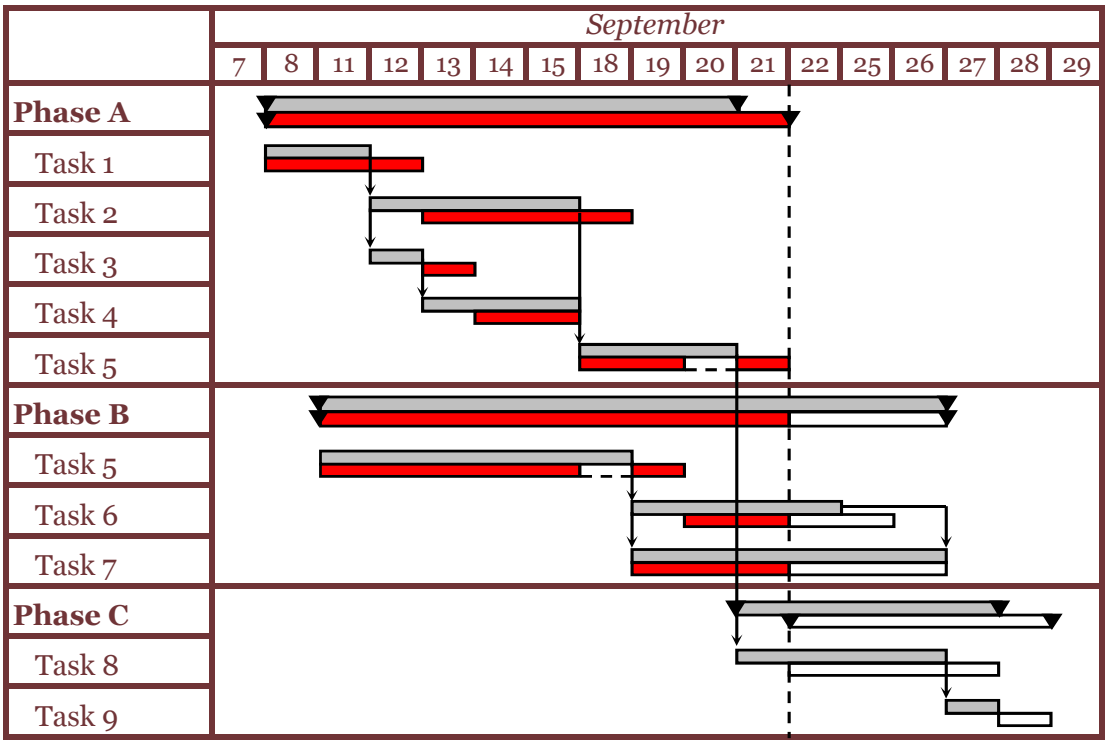


FIGURA 2.2: Un diagrama Gantt, incluyendo seguimiento.

2.3.2. El Diagrama de Hitos (*Milestone Chart*)

El diagrama Gantt que conocemos en la actualidad es una combinación del diagrama Gantt creado por Henry Gantt, y el diagrama de hitos. El diagrama de hitos se basa en eventos planificados en el proyecto, en vez de en la duración de las actividades, y proporciona una medida para decidir si se continúa con el proyecto o la toma de acciones correctivas que permitan finalizar dicho proyecto en tiempo y calidad planificado. De nuevo, no hay un estándar para la notación, aunque sí que han aparecido diversas propuestas para una estandarización de la notación. Desafortunadamente, el diagrama de hitos ha quedado prácticamente relegado en la gestión de proyectos debido en gran parte, a que su fusión con el diagrama Gantt, la opción más utilizada en la actualidad, se ha perdido prácticamente la utilización de este diagrama, excepto en entornos militares, donde aún se sigue utilizando, principalmente por el Departamento de Defensa.

Aunque de nuevo no existe un estándar en la definición del conjunto de símbolos para la representación de hitos, se ha utilizado el conjunto de símbolos fijados

para la comunicación de hitos en el Air Force Materiel Command (AFMC), que se puede observar en la tabla 2.3.

Símbolo	**Descripción**
⇧	Evento Planificado sin completar
⬆	Evento Planificado completado
⌂	Evento Planificado con anterioridad sin completar
◆	Evento Planificado con anterioridad completado

Tabla 2.3: Notación estándar utilizada para la comunicación de hitos.

El conjunto de símbolos descritos en la tabla 2.3 deja abierta la actualización del diagrama mediante la actualización de las fechas de los hitos o eventos. Este hecho se puede observar en la figura 2.3, donde eventos que estaban planificados (en las fases 2 y 3) se han atrasado a fechas posteriores. La actualización del diagrama permite, además de poder tener un diagrama actualizado, poder realizar un seguimiento sobre aquellos hitos que no se han cumplido y, en su caso, poder solicitar informes sobre la situación. Sin embargo, el diagrama de hitos no ofrece de por sí una evolución de las tareas del proyecto, sino que se centra en describir las fechas clave del proyecto, que pueden ser bien fechas de entrega de documentación o productos, o fechas de finalización de actividades o fases del proyecto.

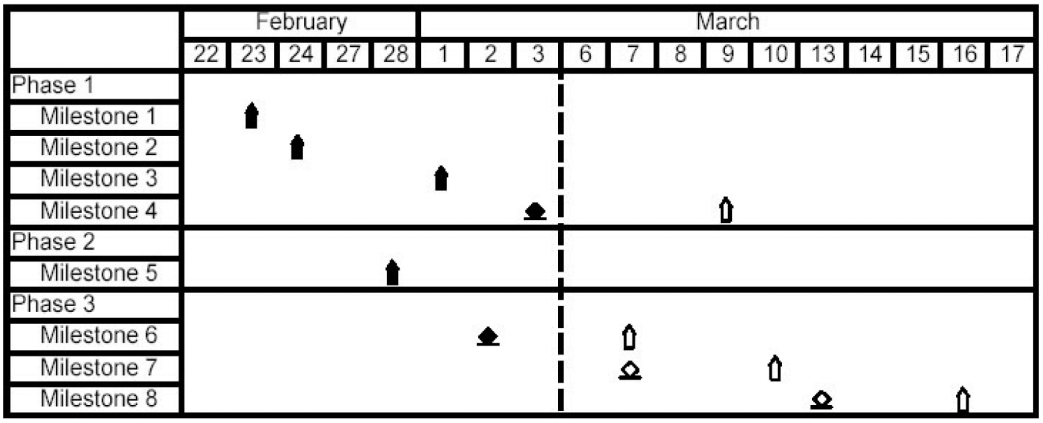


Figura 2.3: Un ejemplo de un diagrama de hitos.

El diagrama de hitos se encarga principalmente en la representación de hitos o eventos, en contraposición con el diagrama Gantt que representa actividades. El diagrama de hitos, que aún se utiliza en la actualidad, sobre todo en entornos

militares, pero generalmente complementan a otros sistemas de representación, como el *Master Scheduling Plan*, el diagrama Gantt (ver figura 2.1) o cualquier otro tipo de diagrama que observaremos en representaciones posteriores.

2.3.3. Teoría de Redes / Diagramas de Red

Más adelante, después de finalizada la I Guerra Mundial, se comenzó a trabajar en la teoría de grafos, desarrollada dos siglos atrás. Concretamente, Leonard Euler resolvió un problema que consistía en encontrar un camino entre los siete puentes de Königsberg, cruzando cada puente una única vez, y volviendo al punto de inicio. Gracias a este problema, Euler resolvió que era imposible encontrar dicho camino, y lo demostró gracias a su Teoría de Grafos [Dre80].

Sobre el año 1920, tras la I Guerra Mundial, observando que el gran handicap para las escasas técnicas de planificación existentes era la propia representación de las dependencias y las restricciones, se llegó a la conclusión de que este problema se podía solucionar aplicando la teoría de grafos al problema de representación de estas restricciones y dependencias. Determinar un paralelismo entre proyecto y la Teoría de Redes fue inicialmente en lo que se estuvo trabajando durante muchos años, desarrollándose los Diagramas de Red ya en los años 50, tras la II Guerra Mundial. Tanto la Teoría de Redes, como los Diagramas de Redes han sido masivamente utilizados desde entonces en la Gestión y Representación de Proyectos, tanto como herramienta de análisis y planificación, como artefacto para la representación de los propios proyectos.

Para comprender los diagramas de red, primero es necesario introducir la notación específica que permita identificar los distintos tipos de diagramas. Así, una actividad se define como una tarea específica o conjunto de tareas que son necesarias para el desarrollo del proyecto, y que consume recursos y necesita de una cierta cantidad de tiempo para ser llevada a cabo. Por el contrario, un evento es el resultado de completar una o más actividades, cuya duración es cero, y es determinable en un punto del tiempo. Y finalmente, una red es la combinación de todas las actividades y eventos que toman parte en un proyecto y las relaciones que existen entre ellos. De esta manera se pueden tener dos variantes de diagramas de red:

- *Activity–on–Node (AoN).* Determina que las actividades están representadas a través de nodos y, consecuentemente, los arcos que unen actividades representan relaciones y/o dependencias entre actividades. El evento se toma como una actividad de duración cero.
- *Activity–on–Arrow (AoA).* Es el tipo de diagrama de red donde las actividades se encuentran representadas a través de flechas, y los eventos a través de nodos.

Determinar cuál se utiliza más en la gestión y análisis de proyectos depende de las herramientas utilizadas. Por ejemplo, en la herramienta de análisis PERT se utiliza AoN, mientras que en CPM se utiliza AoA. Sin embargo, ninguna de las dos herramientas se ha impuesto hasta el día de hoy, ya que han existido propuestas para la gestión de proyectos utilizando ambos tipos de diagramas de red. De las herramientas PERT y CPM hablaremos en la siguiente sección.

Para la construcción del diagrama de red (vamos a utilizar el tipo AoN), se procede a listar las actividades y eventos que van a estar presente en el diagrama y las relaciones de dependencia entre ellas. Una vez determinadas las actividades y dependencias, se determina el orden lógico de las actividades observando sus dependencias. El diagrama se construye de izquierda a derecha y de arriba hacia abajo (en caso de ser necesario).

Las actividades se van representando en el diagrama de manera secuencial, y uniendo aquellas actividades dependientes entre sí, como se puede observar en la figura 2.4.

Una vez determinada la representación de las actividades en el diagrama de red correspondiente (AoA o AoN), se procede con el camino crítico, que se define como "*el camino a través de la red de actividades que tiene la mayor duración que cualquier otro camino*", definiéndose camino (*path*) como "*la suma de todas las actividades de una ruta dentro de la red, desde el inicio de todas las actividades hasta que éste ha finalizado*" [Tha97].

El diagrama de red proporciona una potente herramienta tanto de análisis, como de representación de proyectos, ya que por detrás es soportada por la teoría de redes,

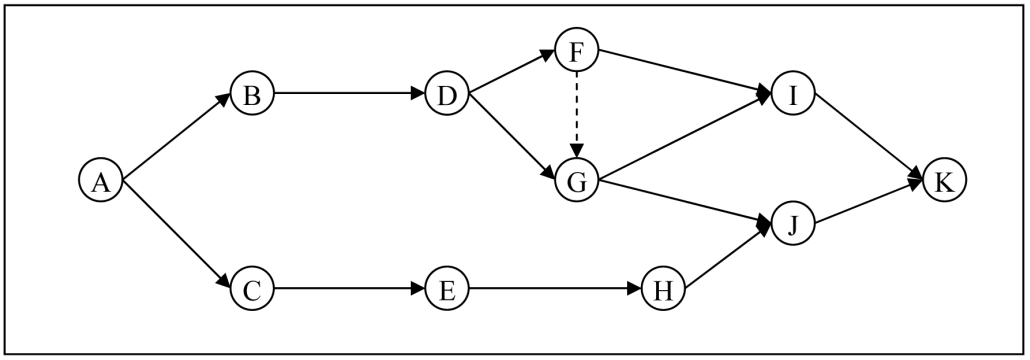


FIGURA 2.4: Ejemplo de un diagrama de red *Activity on Node (AoN)*.

además de ser una potente herramienta de visualización de tanto las actividades y sus dependencias, como de su camino crítico, del que hablaremos en el siguiente capítulo.

2.3.4. Critical Path: PERT/CPM

En el año 1956, E.S. Slagle, por entonces presidente del Instituto de Ingenieros Electricos (IEE), publicó un editorial sobre la necesidad de la utilización de distribuciones de probabilidad en la planificación de los proyectos [Sla56].

En el año 1958, tratando de optimizar los problemas de planificación, y teniendo en cuenta las líneas apuntadas por Slagle, el U.S. Navy publicó lo que hoy conocemos como *Program Evaluation and Review Technique* (PERT), un sistema de gestión de control para el desarrollo del programa armamentístico Polaris [McD01]. En el desarrollo del programa Polaris (1956-1961) intervinieron 250 contratos de desarrollo, y más de 9.000 subcontratas, con cientos de miles de actividades a realizar.

PERT se define como una herramienta de análisis para la optimización del calendario basándose en las estimaciones de tiempo establecido por tarea. PERT se basa en la teoría de grafos, desarrollada durante los años 20, para analizar y optimizar el tiempo estimado de ejecución del proyecto, cuando éste tiempo es el factor fundamental del proyecto.

La idea de PERT surgió para permitir a los gestores la visualización del proyecto completo, observar las relaciones y dependencias entre tareas, y para poder reconocer cuando y dónde los retrasos son aceptables [Sla56].

Paralelamente al desarrollo de PERT, surgió *Critical Path Method* (CPM), siguiendo la misma línea de desarrollo basado en grafos, y para la obtención del camino crítico del proyecto, centrándose en mejorar y visualizar tanto el tiempo como los costes. CPM fue desarrollado por J.E. Kelly de Remington-Rand y M.R. Walter de DuPont para la programación del mantenimiento de las plantas de procesado químico.

Entre CPM y PERT existen ciertas diferencias, aunque el enfoque conceptual es el mismo. Una de las mayores diferencias es que PERT es un método de análisis orientada a eventos, mientras que CPM es un método orientado a las actividades, que gráficamente se observa en el diagrama de red utilizado. PERT utiliza *Activity on Arrow* -AoA-, mientras que CPM utiliza *Activity on Node* -AoN-.

Para la construcción de un diagrama PERT, todas las tareas a ejecutar deben ser visualizadas de una manera suficientemente clara para que éstas puedan formar parte de una red que abarque tanto actividades como eventos (actividades de duración cero). Dichas actividades se dispondrán en una secuencia en la red siguiendo unas reglas de permisividad de visualización que permitan detectar rápidamente el camino crítico. Cada actividad disponen de dos tipos de referencias temporales para hallar este camino crítico: la duración óptima; y la peor duración. Basándose en estos parámetros, y los tiempos sin actividad (por ejemplo, cuando una actividad A para proseguir con la ejecución de la tarea C, necesita de que primero termine la tarea B), el camino crítico se puede definir como la secuencia de actividades lógica cuya ejecución requiera la máxima duración, comenzando desde la primera actividad hasta finalizar con la última actividad. Un ejemplo de diagrama PERT, y cómo se obtiene el camino crítico se puede observar en la figura 2.5.

En la figura 2.5 se puede observar un diagrama PERT con nueve actividades y dos hitos, que corresponden al inicio y final del proyecto. El camino crítico está marcado por las líneas rojas (además de por el recuadro negro).

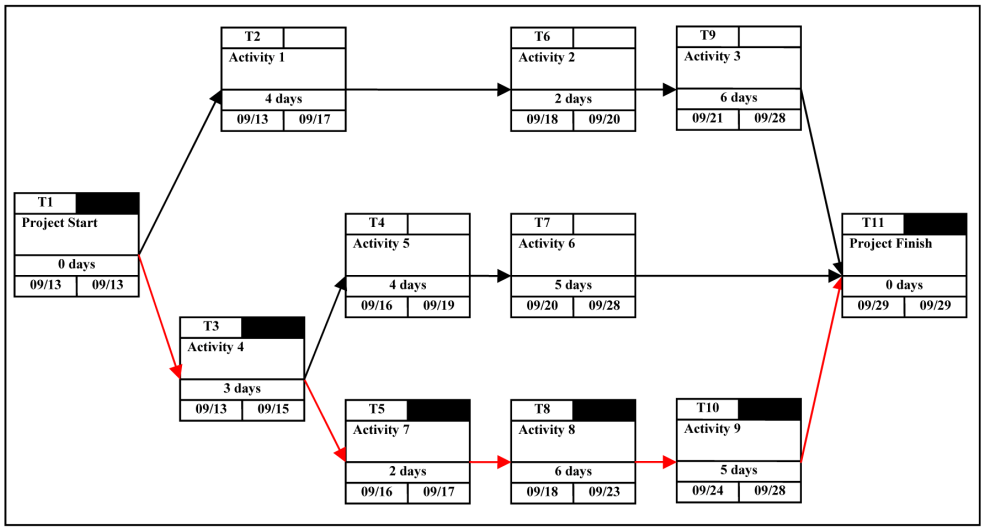


FIGURA 2.5: Ejemplo de un diagrama PERT.

Utilizando esta representación se obtiene una visión global de cómo puede ser la planificación del proyecto, cuáles pueden ser las tareas que pueden excederse de su planificación sin exceder el tiempo del proyecto (aquellas que no están en el camino crítico) y cuáles son las dependencias existentes entre tareas.

El diagrama PERT como herramienta de análisis se basa, dados el tiempo óptimo y peor, utilizar una distribución beta para calcular el tiempo medio de ejecución.

CPM es una variante de PERT, surgida también a finales de los años 50. La base para la representación de las actividades, dependencias y eventos es prácticamente la misma, aunque utiliza un enfoque AoA en vez del AoN utilizado en PERT.

El objetivo de CPM es igualmente hallar el camino crítico. De ahí su nombre "Método del Camino Crítico"(*Critical Path Method*).

Una de las diferencias de CPM frente a PERT es un el cálculo de la planificación, ya que PERT utiliza tres variables temporales para cada tarea (*optimistic*, *pesimistic* y *most likely*), mientras que CPM utiliza una única variable, basada en una duración determinada, y por ello, CPM se utiliza más para proyectos bien definidos, con poca incertidumbre, y donde el tiempo y los recursos a utilizar pueden ser bien determinados.

En la figura 2.6 se puede observar un ejemplo con un diagrama CPM, donde las actividades se encuentran en las flechas, y los eventos en los nodos (marcados con letras).

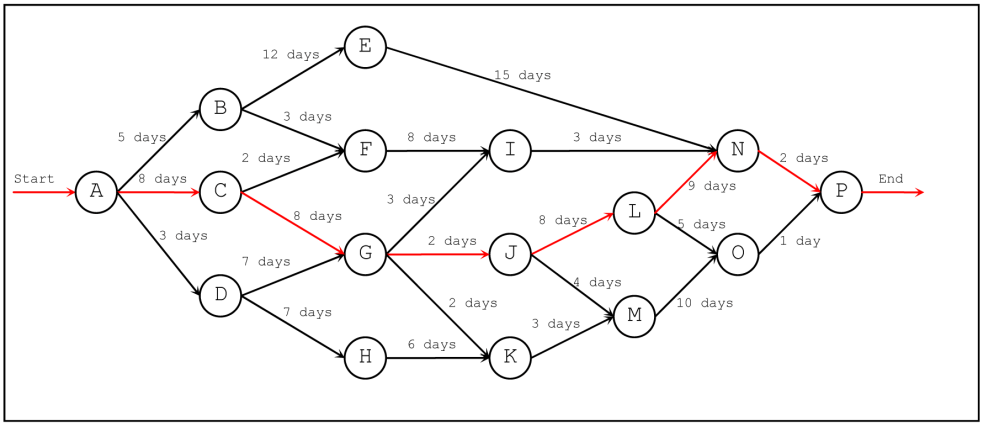


FIGURA 2.6: Ejemplo de un diagrama CPM.

El uso de PERT tuvo su época dorada en los años 60, donde la mayoría de desarrollos de proyectos, la mayoría de carácter militar impulsados por el Departamento de Defensa de los Estados Unidos, se desarrollaban utilizando esta técnica, aunque a la hora de realizar el cómputo de este método, tanto en tiempo que se tardaba en el cálculo del camino crítico, como el coste que éste suponía, hicieron caer en desuso esta técnica, a favor del diagrama Gantt, aunque en la última década, se ha retomado la utilización de esta técnica, sobre todo por parte de empresas del sector privado [Sla56].

2.3.5. Work Breakdown Structure

A inicios de los años 60, la necesidad imperiosa por llevar a cabo cada vez proyectos de mayor envergadura, tanto en los EE.UU. con la Guerra Fría, como en Europa, como en el resurgimiento de Japón como nación puntera en el desarrollo de sistemas de alta tecnología, hacía necesario un replanteamiento de si la división que se había realizado de los proyectos era la correcta.

Debido a que la complejidad de los proyectos se hizo inmanejable incluso para poder desarrollar las técnicas descritas anteriormente, el Gobierno de EE.UU. comenzó a investigar sobre una subdivisión de los proyectos en curso para definir un guía que permitiera, antes de comenzar un proyecto, definir los recursos y distribución que serían necesarios. Como consecuencia de esta investigación, se desarrolló Work Breakdown Structure (WBS) a finales de los años 60 [Dep68]. Posteriormente, este estándar fue actualizado en [Dep93]. En la actualidad se sigue utilizando como estándard para programas del sistema de defensa, estándo vigente en la actualidad del MIL-HDBK-881A [Dep05a].

Fuera de los entornos militares, WBS ha sido utilizado como herramienta para la división del trabajo en proyectos que por su extensión o duración, hacían necesaria una división adecuada de las tareas a realizar.

El principal objetivo a la hora de estructurar el proyecto, es que los elementos en los que divide el proyecto deben ser:

- Gestionables, donde se puede asignar una autoridad y responsabilidad específica,
- independientes, con mayor o menor dependencia y/o relación con los demás elementos,
- integrable, de tal manera que el conjunto forme el proyecto,
- y mensurable, donde puedan determinarse los costes, los recursos necesarios, y evaluar el progreso.

El Diagrama de Descomposición del Trabajo (*Work Breakdown Structure* - WBS) es una subdivisión orientado al producto de la familia de los árboles de los elementos físicos, servicios y datos necesarios para obtener el producto final [Ker09].

La principal aportación de WBS no es la mera división en tareas manejables del proyecto, sino que proporciona un marco de trabajo que puede ser posteriormente utilizada en otros entregables del proyecto, como la matriz de responsabilidad, la planificación, el presupuesto, el análisis de riesgos, la estructura de la organización, la coordinación de objetivos, el seguimiento o el control del proyecto. Con

WBS, el proyecto puede ser descrito en base a sus componentes. La división de las tareas puede ser realizada de varias manera dependiendo de las necesidades de cada proyecto particular, pero la manera más común se puede definir como una estructura de seis niveles descrita en [Ker09], tal como se puede observar en la tabla 2.4:

	Nivel	**Descripción**
Niveles de Gestión (Managerial Levels)	I	Programa Total (Total Program)
	II	Proyecto (Project)
	III	Tarea (Task)
Niveles Técnicos (Technical Levels)	IV	Sub-tarea (Subtask)
	V	Paquete de Trabajo (Work Package)
	VI	Nivel de Esfuerzo (Level of Effort)

Tabla 2.4: Notación estándar utilizada para la comunicación de hitos.

El nivel I es el programa de la empresa, y está formado por un conjunto de proyectos a realizar. En el nivel II, el proyecto se divide en tareas, que se subdividen en subtareas para poder realizar un mayor control, ya a nivel técnico. Si es necesario, se aplican los niveles V y VI, donde las sub-tareas quedan divididas en paquetes de trabajo y nivel de esfuerzo.

WBS proporciona una ayuda útil en la estructuración durante la duración del proyecto, proporcionando los siguientes beneficios [Dep05a]:

- Cada uno de los elementos es dividido en componentes, clarificando la relación entre dichos componentes y definiendo la relación de tareas a completar por cada elemento, hasta completar el proyecto.
- Facilita la planificación efectiva y la asignación de responsabilidades técnicas y de gestión.
- Ayuda a realizar el seguimiento de esfuerzos, riesgos, reserva de recursos, gastos y rendimiento del coste, temporal y técnico.

Un buen ejemplo de WBS se puede observar en [Dep68], una propuesta del DoD para elementos de defensa. En la figura 2.7 se puede observar un WBS obtenido de [AMBD04]:

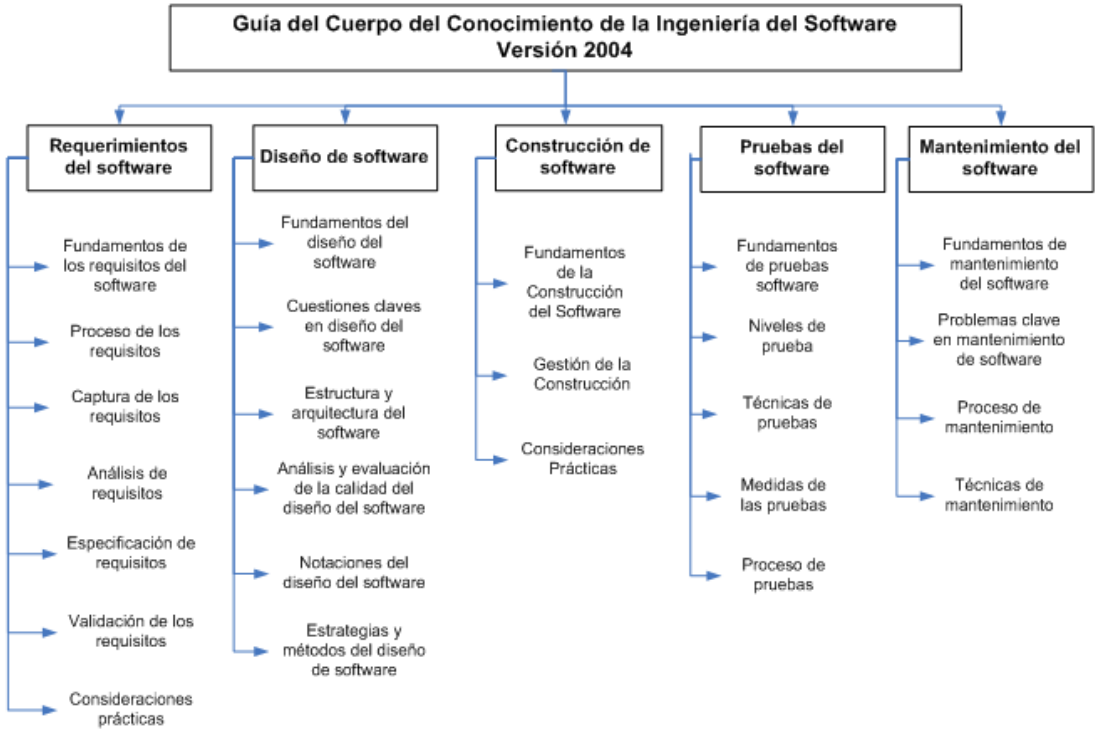


FIGURA 2.7: Un ejemplo de WBS para la división de áreas de conocimiento en el SWEBOK.

Existen variaciones del diagrama WBS, con la misma estructura de árbol, pero donde el enfoque a tomar cuando se realiza la división es diferente [Dep05b][Dep05a]:

- Organizational Breakdown Structure (OBS). Las divisiones en tareas se realizan tomando el punto de vista de la estructuración organizativa, de tal manera que el nodo raíz (proyecto) se divide en bloques organizativos (por ejemplo, departamentos), donde las tareas a realizar serán asignadas a dichos bloques organizativos, que pueden estar compuestos de personas o grupos de personas.

- Product Breakdown Structure (PBS). La división se basa en ir obteniendo productos intermedios en las distintas fases, que den como resultado el producto final. Para esta subdivisión del producto se establecen las diferentes partes que puede tener un producto y las tareas que se asocian a cada punto de diseño, fabricación y montaje.

- *Project Work Breakdown Structure* (PWBS). Esta es la estructura más común, ya que se toma el enfoque de alcanzar unos objetivos generales impuestos al proyecto. De esta manera, se asegura que para cada división realizada, hay que cumplir ciertas metas, objetivos, o hitos, que permitan seguir con el proceso de desarrollo. PWBS es uno de los enfoques más comunes en la estructuración de los proyectos software.
- Contract Work Breakdown Structure (CWBS). En el caso que exista algún contrato externo, éste también puede ser controlado por la empresa que esté desarrollando el proyecto. A este tipo de contratos también se les denomina outsourcing. En [Dep05b] se define cómo gestionar y utilizar un CWBS.

Posteriormente, se quiso establecer una utilidad adicional a los distintos tipos de WBS, y uniendo la organización (OBS) y la estructuración jerárquica (WBS), y se creó RAM –*Responsibility Assignment Matrix*– [Rus05]. Esta estructura determina una asignación entre tareas y personas, y está dispuesta en forma de matriz, de tal manera que las filas corresponden a las actividades (obtenidas del WBS), y las columnas a las personas (obtenidas de OBS). RAM está implementada en algunas aplicaciones de gestión de proyectos, y su utilidad viene dada por la posibilidad de asignación de las tareas en las etapas iniciales del proyecto sin necesidad de realizar estimaciones.

2.3.6. Redes de Petri (Petri Nets)

Los años 60 fue una época en la que Institutos, Departamentos del Gobierno, Universidades y empresas privadas proponían nuevos modelos para la representación de procesos, generalmente centrados en la solución de problemas particulares, y sin posibilidad de extensión a otros ámbitos. Sin embargo, en el año 1962, Carl Adam Petri, presentó su tesis *"Kommunikation mit automaten"* (Comunicación con autómatas) en el Instituto para Instrumentos Matemáticos de la Universidad de Bonn, donde presentaba un novedoso modelo del flujo de información en sistemas, denominado Petri-nets [Pet62, Gre65].

Desde la aparición de esta tesis en 1962 hasta la actualidad, las Redes de Petri o *Petri–Nets* han sido una base para el estudio y desarrollo de muchos trabajos

de investigación relacionados con el modelado de sistemas y la aplicación de éstos [Com11], de los cuales estudiaremos en secciones posteriores los más destacados.

Una Red de Petri es un grafo bipartito dirigido que puede tener dos tipos de nodos: los lugares o *places* (P) representados por círculos, y las transiciones o *transitions* (T) representadas por líneas. Las conexiones se realiza a través de las funciones de entrada (I) y salida (O), representadas por arcos dirigidos desde los places hasta las transiciones, y desde las transiciones hasta los lugares, respectivamente. Las conexiones entre el mismo tipo de nodo no está permitida. Las marcas de las Petri-Nets son asignaciones de elementos o *tokens* a los lugares de la red. Las marcas se representan como círculos rellenos dentro de los lugares. En la figura 2.8 se puede observar un ejemplo de una red de Petri.

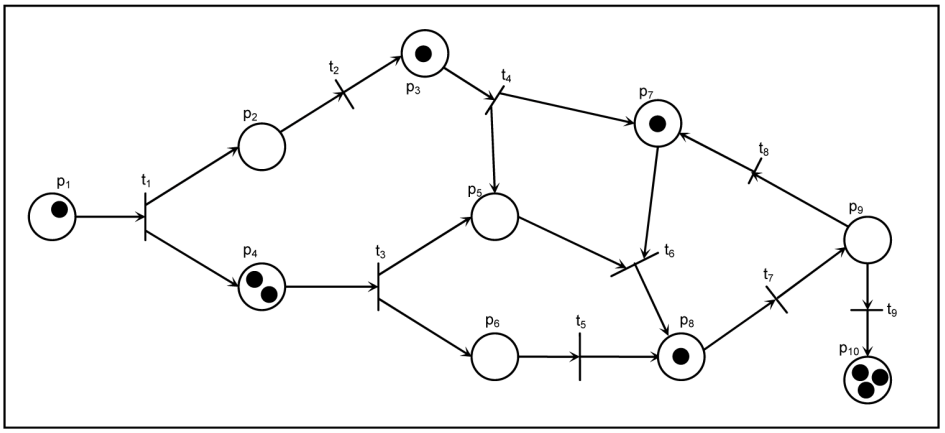


Figura 2.8: Ejemplo de una Red de Petri.

Teniendo en cuenta la construcción de la red de Petri, y las marcas o estados de los *places*, las reglas de ejecución para el lanzamiento en las transiciones (*firing*) se basan en que inicialmente la transición tiene que estar activa para poder ser lanzada. Una transición está activa en todas sus places de entrada existe al menos un token. En el ejemplo de la figura 2.8 se puede observar que las transiciones activas son t_1, t_3, t_4 y t_7. La transición t_6 no está activada porque en el place p_5 no hay tokens.

El proceso de lanzado funciona depositando en los places de salida uno de los tokens existentes en todos los places de la transición. En la figura 2.8, si se lanza

la transición t_4, esto depositará un token en el place p_7 y otro en el p_5, eliminando el token existente en el place p_3.

Aunque Petri Net es un método para el modelado de estados, eventos, condiciones, sincronización, paralelismo, selección e iteración, cuando los procesos tienden a ser complejos y largos, las redes de Petri no permiten el modelado a dicho nivel. En pocos años, observada el potencial que ofrecían las Redes de Petri para el modelado, se comenzó a investigar sobre extensiones al modelo para el modelado de redes complejas. Algunas de dichas extensiones tempranas se pueden observar en [Pet77], aunque las extensiones que revolucionaron el concepto de redes de Petri se basa en dos aportaciones: *Coloured Petri Nets* y *High-Level Petri-Nets*. Durante estos últimos años han seguido apareciendo extensiones para la resolución de problemas de modelado particulares, que han tenido mayor o menor éxito. En [Mcl05] puede observarse un resumen de herramientas y extensiones a las redes de Petri.

2.3.7. El Modelo IDEFØ

IDEFØ (*Integration DEFinition for Function Modeling*) es un método basado en la combinación de una representación gráfica de funciones y un texto explicativo para el modelado de decisiones, acciones, y actividades de una organización o sistema [KBS93]. IDEFØ forma parte de un conjunto de métodos [KBS94] definidos para el diseño y modelado del software. Entre los métodos más importantes se encuentran IDEFØ (Modelado de funciones), IDEF1 (Modelado de la información), IDEF1X (Modelado de datos), IDEF3 (Método de Captura de Descripción del Proceso), IDEF4 (Método de Diseño Orientado a Objetos) e IDEF5 (Método de Captura de Descripción de Ontologías). El modelo IDEFØ se basa en SADT (*Structured Analysis and Design Technique*) [MM05], que determina un modelado de las funciones para el análisis y comunicaciones de un sistema.

Aunque IDEFØ no sea un modelo extensamente utilizado en la gestión de proyectos, en entornos de proyectos acordados con el DoD (Departamento de Defensa de Estados Unidos) se utiliza como medio básico de comunicación y modelado, ya que permite determinar de una manera estructurada la información que es necesario

conocer [Han95]. Los principales componentes del modelo IDEFØ se muestran en la tabla 2.5.

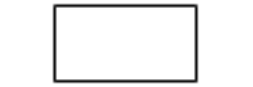
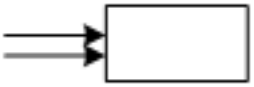
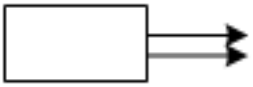
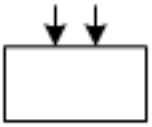
Símbolo	**Descripción**
	Función / Acción / Actividad (*function*)
	Entradas (*input*)
	Salidas (*output*)
	Control (*control*)
	Mecanismos / Herramientas (*mechanism*)

Tabla 2.5: Componentes representados en IDEFØ.

Así, IDEFØ tiene como nivel raíz un diagrama denominado A-0. A partir de A-0, se van construyendo nuevos niveles, dependiendo del nivel de detalle requerido para definir las funciones. Las funciones (actividades, transformaciones o acciones) son cajas negras que utilizando las entradas (*inputs*), bajos los mecanismos de control (*control*), y utilizando las herramientas o mecanismos necesarios (*mechanism*), obtienen la salida deseada (*outputs*).

El principal objetivo de IDEFØ es, por una parte, el desarrollo de una representación gráfica estructurada de las funciones (actividades, procesos, acciones u operaciones) de un sistema o proyecto que soporte la integración de sistemas, y por otra, la definición de una método de modelado independiente de las herramientas CASE (*Computer Aided Software Engineering*). IDEFØ se ha diseñado para que se pueda utilizar conjuntamente con herramientas CASE, y para proporcionar un método de modelado que posea las características deseables de ser general, precisa, concisa, y que proporcione la conceptualización y flexibilidad [KBS93]. En

la figura 2.9 se puede observar un ejemplo de un diagrama IDEFØ, donde se observa un conjunto de actividades que tienen un conjunto de entra-das resultado de las salidas de otras actividades, y controles y mecanismos que se aplican a las actividades.

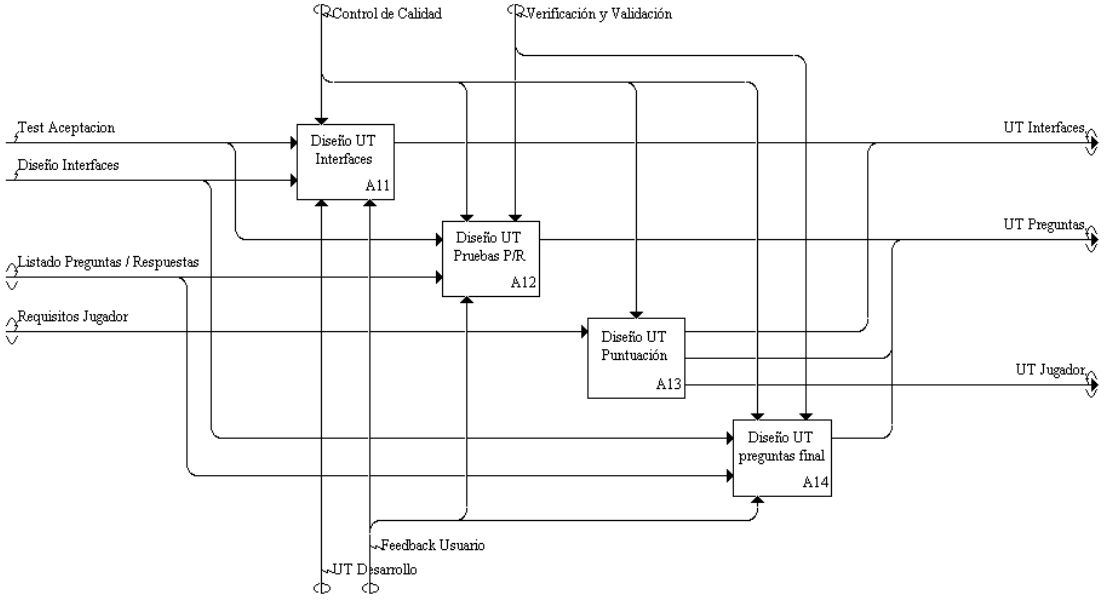


FIGURA 2.9: Ejemplo de un diagrama IDEFØ

2.4. Clasificación de Modelos de Representación

La gama de modelos de representación que se ha detallado en la sección anterior es muy amplia, aunque únicamente son los más genéricos los que están implementados en las principales aplicaciones informáticas de gestión de proyectos. Estos modelos de visualización abstraen a través de objetos gráficos y arcos la representación de los componentes de un proyecto, ya sean actividades, recursos, productos, hitos o dependencias. Esta abstracción proporciona esa representación visual para hacerlos cognitivamente entendibles por los implicados en el proyecto. Los principales modelos de representación se pueden clasificar, según su estructuración, en dos grandes grupos:

- Modelos Jerárquicos. La representación se realiza mediante la división de los componentes en niveles de refinamiento que permita, para cada nivel, una definición más detallada del componente a representar.
- Modelos Estructurales. En estos modelos se establece, para un nivel descriptivo dado, la visualización de la información en dicho nivel. Si se representan niveles más genéricos o abstractos, están enfocados más a tareas de gestión o dirección, mientras que si se definen de manera más descriptiva, se enfoca más hacia los desarrolladores.

En la figura 2.10 se muestra una clasificación de los modelos de representación más representativos. Hay que tener en cuenta que la división se ha realizado en función de las características de representación que son objeto de estudio dentro del contexto de la actual tesis.

2.4.1. Modelos Jerárquicos

Los modelos jerárquicos proporcionan una visión de arriba hacia abajo (*top–down*) de las actividades del proyecto, donde el nivel superior viene dado por el propio proyecto y las hojas corresponden a actividades o tareas que parten de la división que se realiza de los procesos a alto nivel. Esta estructuración proporciona un punto de vista general del proyecto y de las tareas que lo componen. Dentro de esta categoría cabe destacar los siguientes modelos: la Estructura de Descomposición del Trabajo (WBS), el Diagrama Gantt, e IDEFØ.

2.4.2. Modelos Estructurales

Los modelos de representación estructurales realizan una visión del proyecto a nivel de tareas, por lo general. De esta manera, se permite definir únicamente la información necesaria para, por una parte, poder ser manejada por la dirección o los gestores del proyecto, y por otra utilizarse para detallar, planificar y asignar como paquetes de trabajo a los desarrolladores.

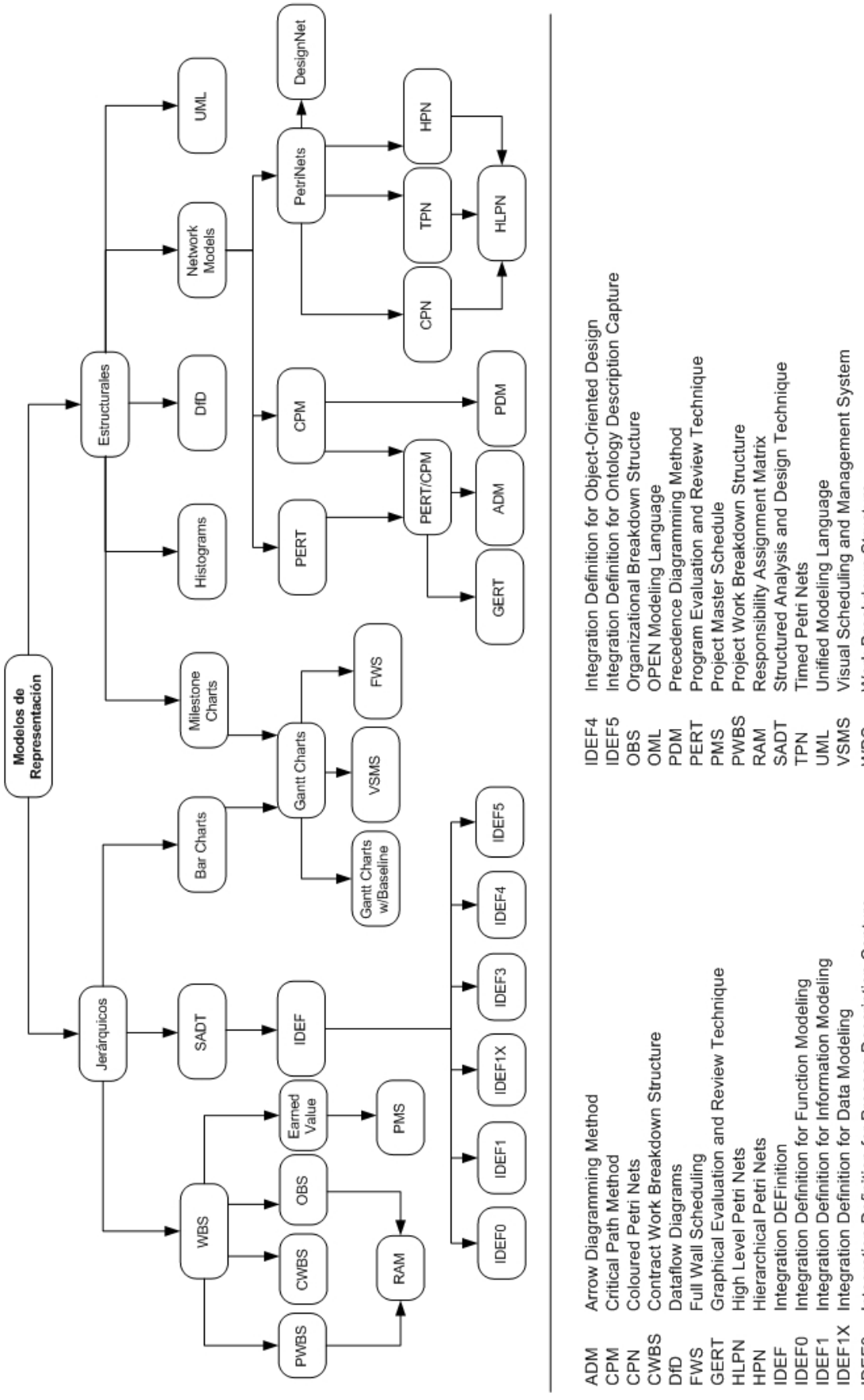


FIGURA 2.10: Clasificación de los Modelos de Representación

De entre los modelos descritos en la figura 2.10 caben destacar por su importancia para la planificación y modelado de proyectos software el diagrama PERT/CPM y las redes de Petri. Aunque en la actualidad los modelos presentes en UML han cobrado importancia en el diseño de sistemas software, su ámbito queda fuera de este capítulo por la variedad existente en los modelos y su incapacidad para realizar representaciones genéricas del proceso de desarrollo.

2.5. Conclusiones

Los modelos de representación constituyen una herramienta esencial para la estructuración, gestión y abstracción de los proyectos software. Su utilidad es incuestionable en la gestión de proyectos debido a que proporcionan una visión general de los componentes a utilizar durante todo el proceso de desarrollo y ayudan a la planificación y representación de las tareas a realizar durante el proyecto. La representación se convierte en fundamental en las etapas iniciales de un proyecto, donde se determina el modelo de planificación, desarrollo, control y gestión del resto del proyecto. La representación supone una gran ayuda al proceso de desarrollo software, ya que proporcionan una vía para su definición, planificación, verificación y seguimiento. En este capítulo se han presentado los principales modelos de representación utilizados habitualmente en la gestión de proyectos software. Así, desde la concepción y definición del proceso de desarrollo mediante el uso de la WBS, hasta la finalización del proyecto se observa la necesidad de utilizar alguno de estos modelos.

La representación proporciona una abstracción del proyecto en sus elementos más significativos. En muchos casos los componentes representados proporcionan una herramienta para la comunicación con claridad entre las personas implicadas en el proyecto, posibilitando verificar la corrección de las actividades y recursos que forman dicho proyecto. Muchos de los modelos presentados han evolucionado para adaptarse a nuevas metodologías y necesidades que han ido surgiendo en la ingeniería del software.

Los modelos presentados modelan parte del proceso de desarrollo del software, sobre todo cuando surge la necesidad de gestionar dicho desarrollo. Otros modelos

de representación que han quedado fuera de este capítulo no se han incluido bien porque su utilización en proyectos software es menor, o bien porque están basados en alguno de los modelos que ya se han descrito aquí.

Capítulo 3

Una representación visual de la información del proyecto: PARMA

3.1. Introducción

En el capítulo anterior se ha realizado un estudio de los métodos de representación, focalizando el interés sobre qué elementos son los fundamentales para una adecuada visualización que proporcione la información necesaria al gestor del proyecto. Sin embargo, dentro de la gama de modelos de visualización de la información de proyectos, la información que proporcionan es muchas veces limitada. Dentro del proceso metodológico definido para esta tesis, en este capítulo se presenta PARMA –*Project–Activity Representation MAtrix*-, una propuesta de modelo para la representación matricial de la información de proyectos software. Con el planteamiento expuesto en este modelo de representación se pretende resolver el enfoque en la visualización de los tres componentes principales de los proyectos: Personas, Procesos y Productos. Para ello, PARMA utiliza una construcción de la información del proyecto incremental guiada por el ciclo de vida, por lo que es un modelo válido para la representación, control, seguimiento, y gestión de la información.

Con el contenido de este capítulo se pretende concienciar de la necesidad de evaluar si las actuales herramientas para la gestión de proyectos proporcionan una visualización adecuada de los elementos que representan un proyecto.

3.2. Información fundamental en la visualización

Uno de los valores fundamentales para el éxito de los proyectos es la información que se tiene de éstos, incluyendo objetivos, estructura, organización, planificación, presupuesto, comunicación, y que toda esta información esté actualizada y visible en todo momento [Jur99]. Para ello, es necesario tener en cuenta los tres grandes componentes de un proyecto: Procesos, Personas, y Productos (genéricamente denominadas las 3Ps) [Phi04]. En Ingeniería del Software, mantener en todo momento la información global del proyecto se convierte en un hecho esencial, ya que el proceso de desarrollo está saturado de tareas a realizar, y el manejo de toda esta información es complejo. Y gestionar toda esta información no es natural para los participantes en el proyecto, ya que generalmente éstos se dedican más a las tareas de desarrollo [HKM01]. La propia naturaleza de desarrollo incremental del software, la gestión efectiva de los recursos, y la corrección del proceso de desarrollo del software posee una gran cantidad de información a gestionar. Pero el principal problema surge a la hora de representar de esta información del proyecto, ya que los modelos de representación que visualizan esta información no plasman en sus representaciones las 3Ps en su totalidad. Los modelos actuales tampoco representan el seguimiento del proyecto, al tratarse de modelos estáticos que son generados al inicio del proyecto, y cuya revisión (planificación, recursos, costes, alcance, refinamiento de la información) no se suele modificar según avanza éste, excepto en casos de desviaciones en los requisitos, calendario o necesidades de recursos humanos en el proyecto. Ejemplos de estos modelos estáticos son los diagramas PERT y Gantt, que ya se han descrito en el capítulo anterior. A pesar de estas limitaciones, los modelos estáticos suponen una de las herramientas de visualización y gestión vigentes que se siguen usando intensivamente para la gestión de proyectos [WF02].

Por otra parte, la propia naturaleza de los proyectos en Ingeniería del Software es incremental [NG88], esto es, la adquisición de información en el proceso de

desarrollo del software tanto en las etapas iniciales, como en el desarrollo, e incluso en el mantenimiento del software, se construye principalmente de una manera progresiva, según se va teniendo la información del proyecto, se van desarrollando las tareas de éste, o se van finalizando ciertos procesos. Esta naturaleza incremental requiere de una dedicación lineal a las tareas de gestión durante todo el ciclo de vida del proyecto. Pero en realidad las tareas de gestión en los proyectos suelen quedar relegadas a las fases iniciales de determinación de las tareas del proyecto: se definen los requisitos, se describe la tarea, se establecen los recursos y costes y se realiza una planificación. Esta realidad choca frontalmente con los procesos de gestión que se proponen en el PMBOK [Ins08].

3.3. PARMA

PARMA es una propuesta que trata de solventar, utilizando un enfoque de herramientas informáticas, las limitaciones en la visualización de la información, centrándose en la visualización de los recursos humanos, las actividades ligadas a los procesos y los resultados de ejecución de las actividades. El modelo propuesto trabaja sobre el supuesto de que se puede representar mayor cantidad de información de proyecto debido a que no se utilizan símbolos iconográficos, presentes en otros modelos de representación. Esta característica permite un mayor aprovechamiento del espacio limitado por el tamaño de pantalla o página, y la representación de mayor cantidad de información, aprovechando el espacio disponible y sin eliminar claridad en la representación. El enfoque tomado para el desarrollo se basa en la *Team Role Responsibility Matrix* [Hum99] y la *Responsibility Assignment Matrix* [Ins08], donde se determinan las asignaciones de roles y responsabilidades de los miembros del proyecto. PARMA extiende esta visión, añadiendo la gestión de la información del proyecto. El enfoque matricial para la representación de elementos del proyecto ya ha sido utilizado por otros autores para desarrollar herramientas para la ayuda a la gestión [Lid09, Pol93].

Todos estos modelos y representaciones visualizan únicamente ciertos elementos de un proyecto, obviando el resto de información del proyecto. La principal aportación del modelo propuesto es la ampliación de la visión matricial para la representación

de la mayor cantidad de información del proyecto, incluyendo todos esos aspectos que se representaban en las citadas matrices, como la planificación o las dependencias.

3.4. Componentes de PARMA

El modelo de representación PARMA se basa en la utilización dos componentes para representar la información del proyecto: la Matriz de Representación (o Matriz PARMA), y las Unidades de Información (IU).

3.4.1. Matriz PARMA (Matriz de Representación)

La Matriz PARMA define la estructura jerarquica de uno o varios proyectos, la organización para dichos proyectos, y las asignaciones de tareas del proyecto. Estas asignaciones se corresponden con las Unidades de Informacion (IU). El grado de finalización de estas IU, representadas por un circulo, se determinan segun la evolucion de la tarea asociada, y varian en su representacion desde que no estan iniciadas (∘) hasta su completa finalizacion (•), utilizando la escala de grises para determinar la evolucion intermedia. En la figura 3.1 se observa un ejemplo de representacion de esta matriz.

Las filas de la matriz determinan la estructura del proyecto, mientras que la organización se define por las columnas de la matriz. Los cruces entre filas y columnas determinan las posibles asignaciones de tareas que se pueden realizar. Las asignaciones se realizan sobre los niveles inferiores de la estructura del proyecto (generalmente sobre tareas), y según se van completando estos niveles inferiores, quedan determinados los niveles de información superiores, hasta llegar al nivel del proyecto. De esta manera, se tiene en todo momento monitorizado que tareas se han realizado, por quien, y cual es la evolución del proyecto.

Esta matriz determina, sin indicar los roles que desempeña cada componente del proyecto (tal como se realizaba en la Matriz de Asignación de Responsabilidades

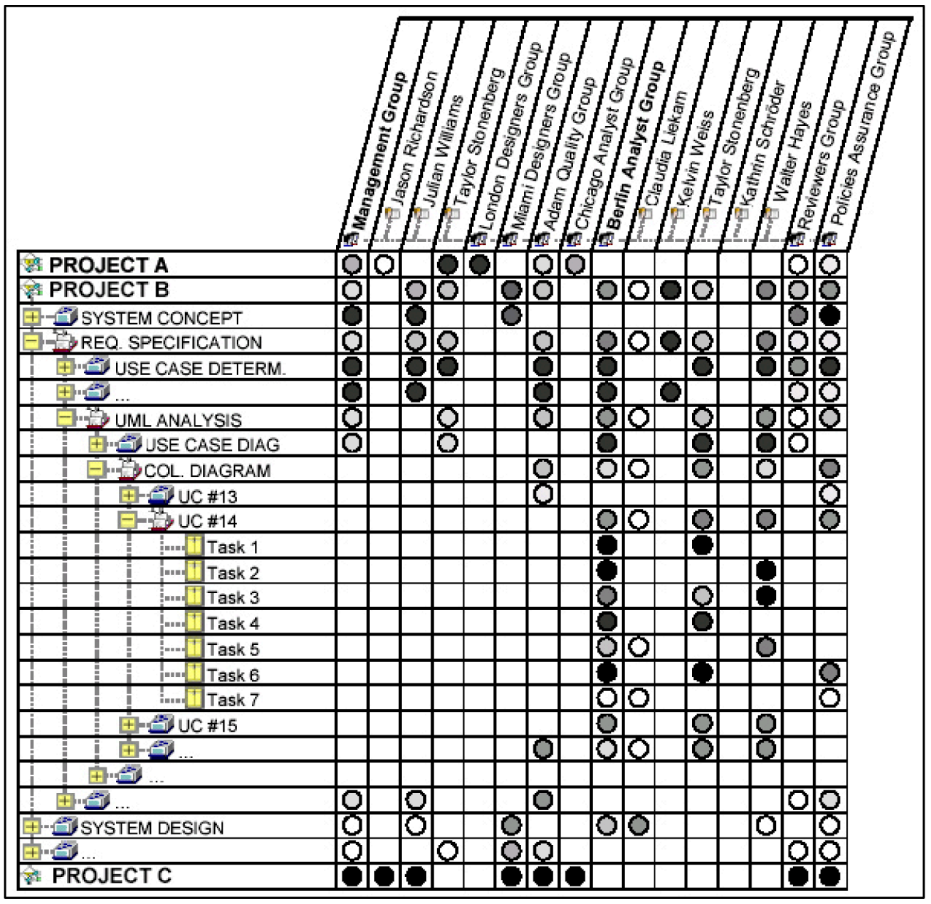


FIGURA 3.1: Un ejemplo de una representación de la matriz de representación PARMA.

[Ins08]), las asignaciones a realizar a través de las Unidades de Información, que contienen la información de la tarea correspondiente a desempeñar.

3.4.2. Information Units (IU)

Las Unidades de Información contienen pares *(parámetro, valor)*, donde el parámetro determina la etiqueta de la información contenida en el valor. Estos parámetros son comunes en todas las IU, mientras que el valor varía entre tareas. En la figura 3.1 se observa un ejemplo supuesto de IU, diseñado a efectos de presentación.

Los parámetros que pueden ser definidos en las IU son: Descripción; Responsables; Duración; Fechas de Inicio/Fin; Coste; Incidencias; Errores; Entregables obtenidos;

Código Fuente; Documentación; Dependencias; Criterios Previos; Calidad; Hitos; etc. Todos estos parámetros se definen para obtener una representación lo más aproximada a la realidad del proyecto (y sus tareas), y pueden estar definidos por una o varias métricas que logren este objetivo. El conjunto de parámetros que pueden utilizarse en este sentido está definido por el conjunto de métricas definidas en [DS00].

Además de estos parámetros, el usuario puede definir otros tantos parámetros como sean necesarios para que la representación de la información del proyecto represente el mundo real. Estos parámetros permiten al usuario realizar de una manera más eficiente y efectiva sus tareas correspondientes, pero también permite llevar un control sobre cualquier tipo de desviación del proyecto, ya sea de tiempo, de coste, de errores, o cualquier otro riesgo que pueda afectar al desarrollo del proyecto.

3.5. Ciclo de Vida PARMA

PARMA es un modelo de representación basado en la naturaleza incremental del desarrollo de los proyectos de Ingeniería del Software [NG88]. Para poder representar dicha naturaleza incremental, PARMA se guía por un ciclo de vida propio, donde la información se va representando en el modelo según ésta se va obteniendo del proceso de desarrollo. Para representar esta información, se ha seleccionado la representación matricial, ya que esta tiene la mayor ganancia de información, al no utilizar barras ni símbolos iconográficos que se utilizan en otros modelos de representación.

El ciclo de vida de PARMA y el proceso de desarrollo del software van ligados, por lo que tenemos que el ciclo de vida de PARMA comienza cuando comienza el proceso de desarrollo, y finaliza cuando finaliza el proyecto.

En el ciclo de vida se identifican seis procesos principales y dos procesos de apoyo para la construcción del modelo de representación de la información. Los procesos

principales son: Determinación de la Estructura del Proyecto (PSD); Determinación Organizativa del Proyecto (POD); Asignación Usuario-Tarea (UTA); Determinación de la Información del Proyecto (PID); Ejecución del Proyecto (PEX); y Finalización del Proyecto (PFI). Los dos procesos de apoyo son: Refinamiento de la Información (IRE); y Acciones Correctivas (CAC). La división en estas etapas viene dado por la unión del ciclo de desarrollo de software y la obtención progresiva de la información del proyecto. En la figura 3.2 se observa la representación gráfica del ciclo de vida.

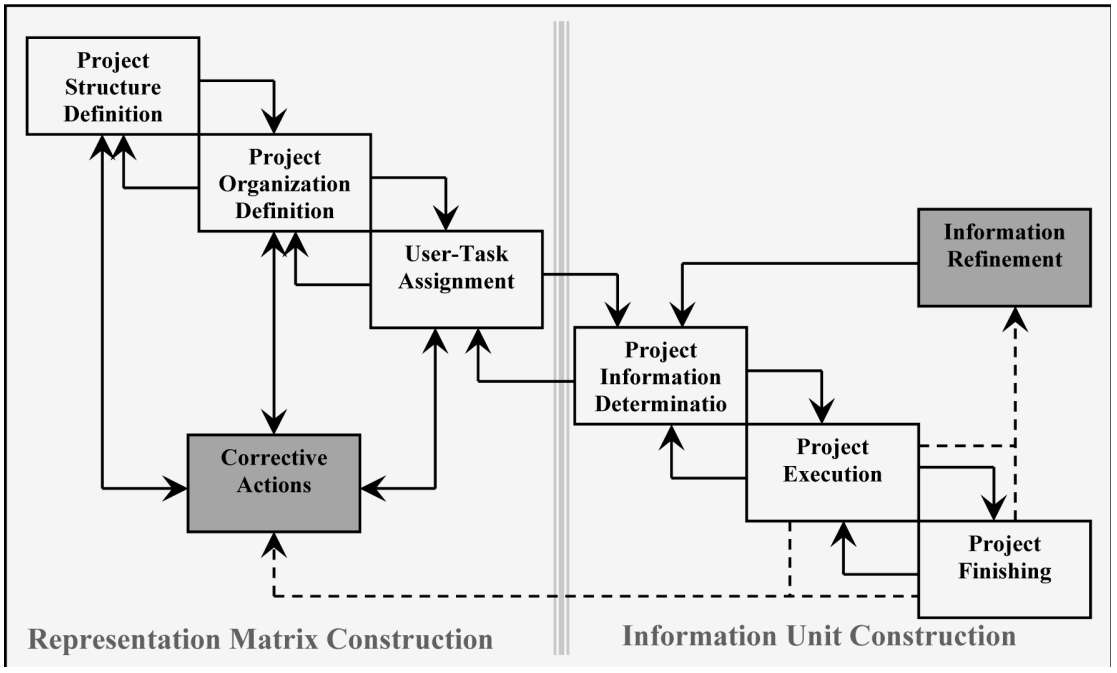


FIGURA 3.2: Ciclo de Vida para la Representación Visual de PARMA.

3.5.1. PSD: Definición de la Estructura del Proyecto (*Project Structure Definition*)

En PSD se encuentran todos los elementos que forman parte de la estructura del proyecto. La estructura del proyecto se puede obtener parcialmente de la descomposición del trabajo a realizar, donde se realiza una división del proyecto a nivel de proceso, refinando a partir de ahí, hasta llegar a nivel de tarea. La división de las tareas dependerá del enfoque que se utilice para la descomposición [Dep05b]. El objetivo principal del proceso PSD es determinar una estructura que alcance

un nivel de definición de unidad de trabajo (tarea), para así poder tener un mayor control sobre la asignación y el control del proyecto. Esta estructura queda representada en la Matriz de Representación en las filas, tal como se observa en la figura 3.1. Por otra parte, la estructura del proyecto es totalmente dinámica, pudiéndose modificar, eliminar, añadir nuevas tareas, en cualquier momento del desarrollo del proyecto.

3.5.2. POD: Definición de la Organización del Proyecto (*Project Organization Definition*)

Para el proyecto es fundamental mantener una correcta estructura organizativa. En este proceso se realiza la definición de la estructura organizativa, que queda determinada por el conjunto de personas que van a trabajar dentro del proyecto. Para dinamizar el modelo, las personas podrán trabajar en uno o varios grupos de trabajo dentro del proyecto (siguiendo una organización matricial, por ejemplo), pudiendo desempeñar uno o varios roles dentro de cada grupo, de manera similar a la conocida Matriz de Responsabilidad [NG88, Hum99]. Al igual que con la estructura del proyecto, la organización es dinámica en el sentido que puede actualizarse, eliminar o añadir personas dentro de la configuración de la Matriz PARMA.

La organización (personas o grupos) queda representada en las columnas, al igual que sucede en la Matriz de Responsabilidad. De esta manera, quedan definidos los encabezados de columna con la organización para el proyecto, y los encabezados de fila con la estructura del proyecto, que forman la base de la Matriz PARMA. En la figura 3.1 se observa cómo se representa la organización en la Matriz PARMA.

3.5.3. UTA: Asignación de Tareas de Usuario (*User Task Assignment*)

Establecida la estructura y organización del proyecto, se establecen las asignaciones a nivel de tarea-persona en la Matriz PARMA. Estas asignaciones, representadas por círculos (∘), se realizan mediante las Unidades de Información (IU), donde se

establecen los roles desempeñados en cada tarea por las personas que participan en el proyecto. La figura 3.1 muestra la representación de las asignaciones.

La asignación de tareas es un punto fundamental del proceso, ya que en este punto se puede comenzar a estudiar el control, la carga de trabajo, la disponibilidad, la planificación, los riesgos y los factores que pueden hacer que el proyecto fracase o finalice con éxito, ya que se determina la capacidad de las personas para poder realizar las tareas que se han asignado. Para realizar esta asignación, se puede tener en cuenta las aptitudes necesidades para desarrollar dichas tareas, la referencia histórica en el desarrollo de las tareas, y cualquier otro factor que determine que la asignación se haga de una manera correcta para el éxito de dicha tarea, y por lo tanto del proyecto.

3.5.4. PID: Determinación de la Información del Proyecto (*Project Information Determination*)

Una vez realizada la asignación, se procede a dotar de información a las IU. Para ello, las IU disponen de ciertos parámetros estáticos que pueden ser rellenados por los responsables de las actividades para determinar qué, cuándo, cómo y quién tiene que realizar las tareas correspondientes. La figura 3.3 muestra un ejemplo de IU.

La información del proyecto correspondiente a los niveles superiores (a nivel de proyecto, fase, proceso o actividad) se va rellenando de dos maneras, según si se puede determinar en base a si existe información suficiente en sus niveles inferiores o no:

- Automáticamente, se rellenan los parámetros que puedan deducirse como conjunción de sus hijos de nivel inferior, como las fechas de inicio y fin, o el estado.
- Manualmente, los parámetros que necesiten determinar con exactitud qué tipo de trabajo se va a realizar. Por ejemplo, la descripción o el código fuente.

Parameter	Value
Identifier	TSK08.23.v4.02.14A-3
Name	UC #14: Task 3
Short Description	First Design Ideas for the Use Case of delivering an item.
Description	According the domain document, the analysis of Use Case in the last system, and the new proposal of system, do a first detailed design following all those guides of the use case of delivering an item.
Responsable 1	Taylor Stonenberg
Role 1	Analyst & Quality Management
Responsable 2	Walter Hayes
Role 2	Designer
Est. Duration	4 hours
Real Duration	3 hours; 6 minutes
Task Start Date	06/25/2004
Task Finish Date	06/26/2004
Est. Cost	$40
Actual Cost	$31
Work Reports	(Not Yet)
Prerequisite 1 Ref.	DOC00.11.44.v12.1
Prerequisite 1 Name	Analysis of Use Case in Last System
Prerequisite 1 Name	Finished (04/04/2004)
Prerequisite 2 Ref.	DOC05.14.v22.A
Prerequisite 2 Name	Client's Proposed System. Final Draft
Prerequisite 2 State	Finished (06/15/2004)
Prerequisite 3 Ref.	DOC02.9.AAK2.v1.0
Prerequisite 3 Name	Domain Document
Prerequisite 3 State	Finished (12/12/2003)
Encountered Errors	(None)
Report of the Product	(Unfinished)
Obtained Product	(Unfinished)
Quality Policies	ENT.ISO.9002
Millestone	07/04/2004
Actual State	75%

FIGURA 3.3: Un ejemplo de una *Information Unit* (IU) de PARMA.

3.5.5. PEX: Ejecución del Proyecto (*Project Execution*)

Cada una de las tareas del proyecto, tiene que pasar por el proceso de ejecución, donde se lleva un control del trabajo realizado (con informes de trabajo personal), de la adecuación con la planificación y/o presupuesto establecido, de los productos obtenidos, tanto en forma de documentos, informes, código fuente o programas, la calidad obtenida, etc. Esta información queda representada en las propias IU, que contendrán tanto los datos planificados, como los datos reales, para llevar un control de que el trabajo se realiza conforme a la planificación, o no. En este punto, la dirección puede llevar un control de la evolución del proyecto, gestionar esta

información, tomar medidas correctivas si procede (ver proceso CAC), y ampliar y/o actualizar la información (ver proceso IRE).

3.5.6. PFI: Finalización del Proyecto (*Project Closing*)

Finalizada la ejecución de las tareas, se procede al almacenamiento de la información, analizando el trabajo realizado, completando la información de sus predecesores, y si procede, tomando medidas según va finalizado la ejecución de las tareas, actividades, procesos, fases, y/o proyectos. Este proceso determina la completitud de la información de la IU, siguiendo así con la ejecución de otras tareas, el análisis de la tarea actual, procesos de refinamiento de la información (IRE) o acciones correctivas (CAC).

3.5.7. IRE: Refinamiento de la Información (*Information Refinement*)

El proceso de refinación de la información se basa en necesidades durante el desarrollo de las tareas o la finalización de éstas (ver figura 3.1) que determinen la actualización o inserción de información en las IU, para la ejecución de una manera más informada de una o varias tareas a las que pueda afectar este proceso.

3.5.8. CAC: Acciones Correctivas (*Corrective Actions*)

Si se hubiera detectado durante la ejecución o finalización de las tareas alguna desviación frente a la planificación, modificaciones en la configuración del personal o si se detectan tareas faltantes (o sobrantes), se debe realizar una actualización de la Matriz PARMA (la estructura, organización y/o asignación) para reflejar esta situación en el proyecto. Como las acciones correctivas afectan a uno o varios usuarios, y por dependencias, a varias tareas, entonces habrá que notificar a los afectados.

3.6. Visualización de PARMA

A la Matriz de Representación, además de la estructura y la organización, se le puede añadir una tercera dimensión: el tiempo. Con ello, no se trata de obtener una vista tridimensional de la representación de un modelo (se demuestra que, aunque los modelos tridimensionales dan una mejor visión espacial, los modelos bidimensionales carecen de ambigüedad [JC99]), sino de poder trabajar con varias vistas de la misma información.

Tomando como base estos tres ejes, tenemos que Tiempo, Organización y Estructura se pueden combinar para obtener distintas vistas. Las diferentes vistas se pueden observar en la figura 3.4, y son las siguientes:

- Vista Estructura–Organización: Esta vista se utiliza para especificar la información del proyecto, denominada genéricamente Matriz de Representación o Matriz PARMA, donde se determina la estructura, la organización y las asignaciones en el proyecto. Esta vista se utiliza para tener una visión general de la estructura, la organización y las asignaciones del proyecto, así como llevar un control sobre las gestión, planificación y seguimiento del proyecto en cualquier punto de éste.
- Vista Estructura–Tiempo: En esta vista se muestra es la planificación temporal de la ejecución del proyecto. Esta vista se corresponde con el diagrama Gantt del proyecto.
- Vista Organización–Tiempo: Esta vista se centra en el usuario de PARMA, en el sentido de que visualiza la información de aquellas tareas, actividades o procesos donde el usuario toma parte activa en el desarrollo. Esta vista temporal proporciona así mismo una visión organizativa y de planificación del usuario, de tal forma que se pueden mostrar incompatibilidades de horarios tras la realización de las asignaciones, así como las dependencias existentes sobre las tareas propias del usuario.

Cada una de estas vistas da un punto de vista de la misma información de un proyecto, mirado desde varios ángulos, de tal manera, que tanto para la dirección,

los desarrolladores, y los clientes puede verse personalizada la información que sea necesaria en cada momento.

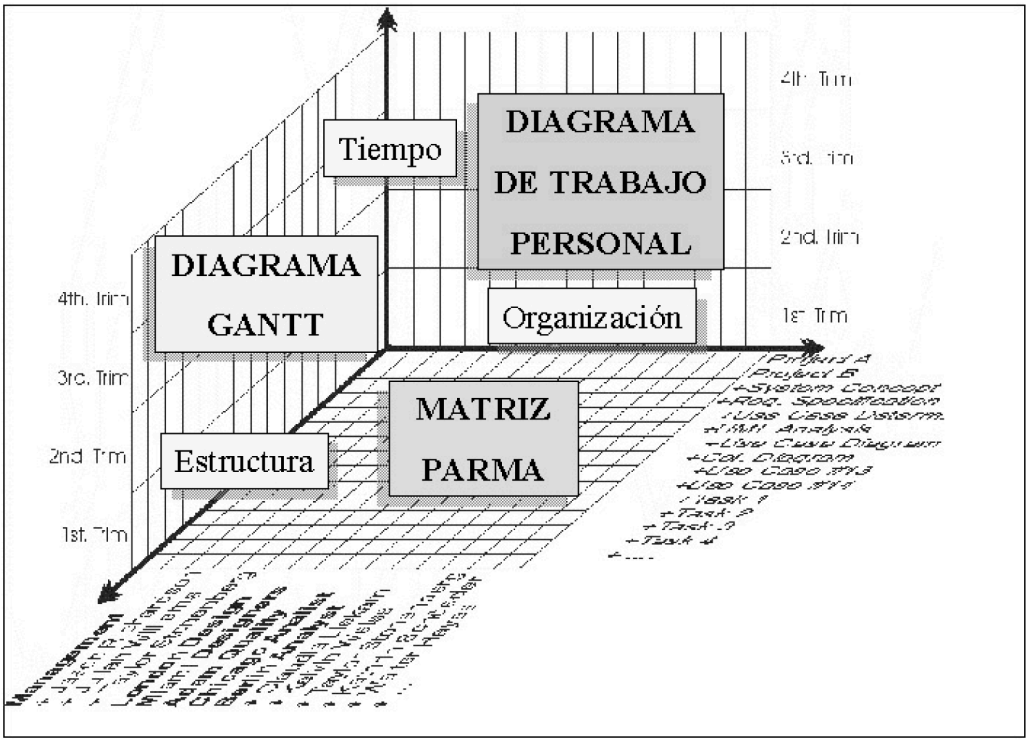


FIGURA 3.4: Las distintas visualizaciones de la representación PARMA.

3.7. Interfaz PARMA

En este punto se propone la interfaz para la visualización de la información de PARMA. Tomando como base la pantalla de un ordenador, la interfaz se divide en tres partes, tal como se puede observar en la figura 3.5.

- Visualización de la Matriz PARMA. La ventana principal muestra la Matriz de Representación PARMA, en cualquiera de las vistas descritas en la sección 3.6. Ocupa la mayor parte de la pantalla debido a que esta matriz, como se ha mencionado anteriormente, es el principal componente del modelo PARMA, y por lo tanto da una idea de la estructura, organización, planificación y evolución del proyecto.

- Visualización de las IU. La información de las IU se visualiza en la ventana lateral adyacente a la matriz, y muestra la información de la asignación seleccionada en la matriz. En esta ventana se muestran todos los parámetros establecidos para dicha IU, determinando la información que se pueda mostrar teniendo en cuenta la limitación del espacio disponible.

- Visualización de Información Adicional. En la ventana inferior se muestra la in-formación extendida de las IU que no pueden mostrarse completamente en la ventana de visualización de las IU. Esta ventana actúa como una extensión a la visualización de las IU, en aquellos parámetros cuya información es más extensa que la disponibilidad de espacio en la ventana correspondiente.

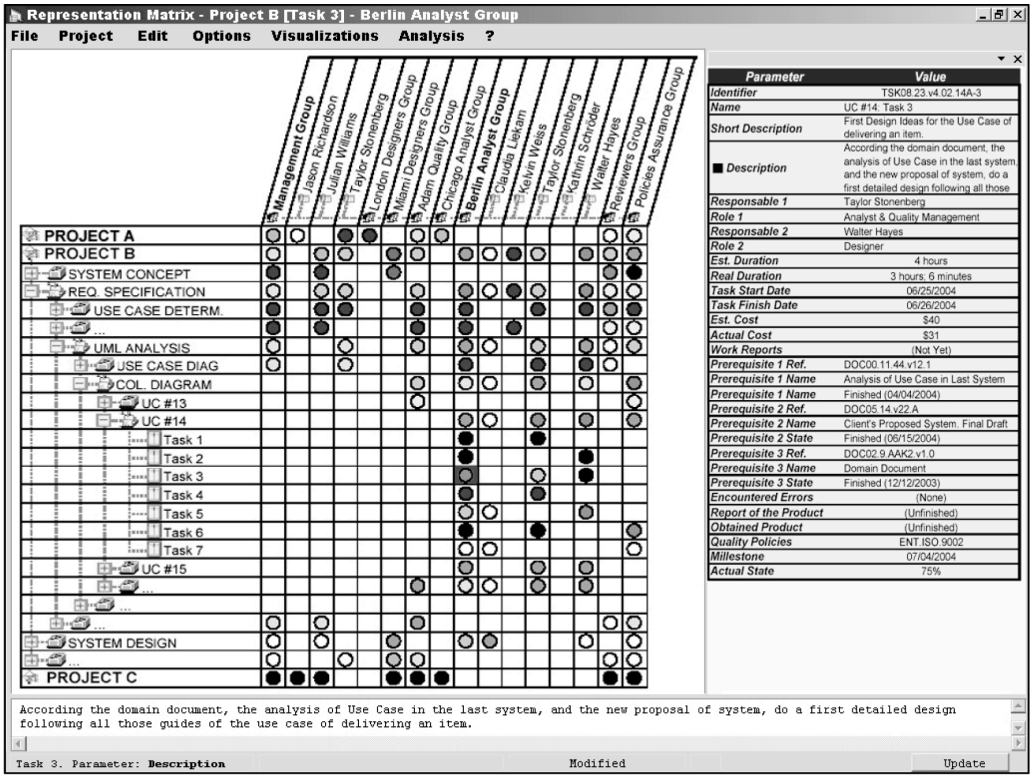


FIGURA 3.5: Interfaz de la representación de PARMA.

3.8. Conclusiones

Basándose en el estudio realizado en el capítulo anterior, en este capítulo se ha propuesto un modelo de representación denominado *Project–Activity Representation MAtrix* (PARMA). Este modelo de representación trata de incidir en aquellos componentes que han aparecido de manera reiterada en los modelos de representación estudiados en el capítulo 2.

El modelo PARMA es una propuesta de modelo de representación de la información para proyectos de Ingeniería del Software que representa matricialmente la estructura, organización y asignaciones (mediante la Matriz de Representación), y la información asociada del proyecto (mediante parámetros en las Unidades de Información). De esta manera, sin depender de la secuenciación de las tareas, controla, gestiona y visualiza la representación, planificación y evolución del proyecto. Para ello utiliza un ciclo de vida que permite añadir y actualizar la información del proyecto incrementalmente.

La representación matricial y la ausencia de símbolos iconográficos, que simplemente restan espacio para la representación de información, permiten que se pueda mostrar en una única interfaz mayor cantidad de información, y por lo tanto, se tenga un mayor poder de decisión para acciones correctivas (también soportadas por PARMA). Además esta información del proyecto puede ser utilizada posteriormente para realizar análisis y obtener conclusiones sobre el desarrollo del proyecto. Por otra parte, se dispone de tres vistas que permiten distintos puntos de vista a los participantes en el proyecto. Finalmente, el modelo PARMA es un modelo de representación incremental y flexible, aplicable a otras áreas de aplicación distintas a la Ingeniería del Software.

El desarrollo de este capítulo surge de la necesidad de realizar un análisis de los métodos de representación utilizados en la actualidad para la representación de proyectos, donde se han estudiado los principales parámetros de un proyecto para la visualización de la máxima información disponible, así como cuáles son los métodos de representación que más intuitivamente pueden percibirse por el usuario. Con PARMA se ha pretendido presentar un modelo que no tenga las

carencias asociadas a otros modelos, y que además represente la máxima cantidad de información correspondiente del proyecto.

Capítulo 4

Diagrama Gantt Extendido

4.1. Introducción

En los proyectos software se producen con cierta frecuencia problemas que afectan al desarrollo, principalmente derivados de una incorrecta aplicación de las herramientas, métodos y mecanismos existentes, como ya se ha podido observar en los capítulos anteriores. Por una parte, desde el punto de vista del proceso, no se utilizan metodologías que se han demostrado eficientes, ni se adecúa el proceso al proyecto, sino más bien justo al revés, es el proyecto el que se adapta al proceso de desarrollo elegido. Por otra parte, desde el punto de vista de la gestión, la problemática proviene de una deficiente aplicación de las herramientas y métodos de los que se dispone.

El conjunto de herramientas de gestión viene dado por la aplicación de buenas prácticas, la utilización de recomendaciones de los estándares, o tener en cuenta el conocimiento extraído en proyectos anteriores. Pero, sin duda, uno de los aspectos fundamentales de la gestión de los proyectos software es precisamente que se gestione de una manera efectiva, esto es, que sea un proyecto organizado, planificado, controlado y dirigido. Para ello, una artefacto muy útil para capturar los datos del proyecto es precisamente el proceso de planificación, ya que proporciona al gestor del proyecto una visualización de la estimación del plan del proyecto, y a los desarrolladores una forma de reconocer la importancia de las actividades que están

ejecutando. Hay varias formas de generar el plan del proyecto, aunque las herramientas más utilizadas son los diagramas de red (por ejemplo, PERT/CPM), los escenarios *what-if* o la distribución equitativa de recursos para evitar situaciones de sobreasignación [Ins08].

Por lo general, un gestor de proyectos utiliza la actividad como el nivel inferior de definición a la hora de dividir el proceso de desarrollo, ya que las actividades proporcionan la unidad atómica de trabajo que puede dividirse y asignarse. Esta división determina la importancia del proceso de desarrollo (cómo se organizan las actividades en el proyecto), pero también de cómo se gestiona éste. Sin embargo, este enfoque se centra más en cómo realizar la ejecución de las tareas programadas, que en quién realiza dichas tareas, por lo que los recursos humanos se consideran como si fueran simples recursos, un componente del proyecto a consumir [Jur99].

Para tener en consideración a las personas es necesario tener un control efectivo sobre la realización de las actividades asignadas, pero también sería adecuado la utilización de una representación apropiada de la información del proyecto, las actividades y las personas. En este artículo se propone una extensión al diagrama Gantt, adaptándolo a un enfoque centrado en las personas, ya que si bien las actividades y sus dependencias son importantes, también lo es representar a las personas que trabajan en el desarrollo del proyecto.

4.2. Características de las visualizaciones

Una representación de proyecto es útil si los implicados en el proyecto la entienden y utilizan. Pero también es necesario que la representación sea sencilla, clara, que los componentes representados se visualicen de manera simple y legible, y que la información esté bien organizada. Pero aún más importante es que la representación suponga una buena abstracción del mundo real.

El diagrama Gantt y los diagramas de red (PERT/CPM) han sido durante décadas las herramientas más apropiadas para representar la información del proyecto. El abanico de representaciones es amplio, y van desde representaciones especializadas, como por ejemplo, Asbru en entornos médicos [SMJ98], LineOfBalance

(LoB) en entornos repetitivos [ATS99], o UML para el diseño software [Gro10], hasta representaciones genéricas, como los diagramas Gantt y PERT/CPM en la planificación [Ker09], o las redes de Petri y las ontologías en la representación de conocimiento. Pero muchos de estos modelos adolecen de ser bien demasiado genéricos, o bien demasiado especializados. Así, las características deseables de una representación visual deberían ser las siguientes:

- Contexto y contenido claro. El uso para el cual fue desarrollada debe tener una equivalencia dentro del contexto, y el contenido debe mostrar únicamente la información necesaria, pero también la suficiente como para ser de utilidad.
- Simplicidad. La información debe ser simple de entender, y que esta información pueda ser manejada y controlada, tanto por el cliente como por el desarrollador.
- Organización. La representación debe presentar la información de una manera organizada, esto es, debe existir una estructura clara de la información y esta información seguir siempre la misma jerarquía en todos los niveles del refinamiento.

Con estas variables, está claro que la complejidad de un proyecto debe quedar inequívocamente simplificado y organizado para representar los elementos esenciales del proyecto que favorezcan gestionarlo eficientemente.

4.3. Diagrama Gantt Extendido

El desarrollo de ciertas tareas especializadas en los proyectos software requiere, por lo general, personas expertas en dichas tareas (o con un conjunto determinado de habilidades). Sin embargo, cuando se realiza la planificación y la asignación, la información sobre quien va a ejecutar cada tarea no aparece representada. Para facilitar la labor del gestor del proyecto, sería útil que la información sobre asignaciones apareciera de manera integrada con la representación de la planificación de las actividades. Para ello, se utilizan generalmente los diagramas Gantt,

generalmente combinados con diagramas de red, ya que ambas representaciones proporcionan las características deseables descritas en la sección 4.2. La representación de la asignaciones es recomendable, debido a:

- La necesidad de comunicación entre los implicados en el proyecto.
- La aparición de nuevos procesos del desarrollo, donde es más importante las personas que el proceso o los productos.
- Las dependencias entre actividades únicamente determinan la ordenación temporal y lógica de las actividades.

Si queremos incluir los recursos humanos dentro del diagrama Gantt es necesario definir el conjunto de componentes que va a utilizar dicha representación, así como las nuevas interacciones surgidas de la extensión del diagrama Gantt. En la Tabla 4.1 se muestran los elementos representados en la propuesta extendida del diagrama Gantt. En esta representación desaparecen las dependencias entre actividades, ya que éstas son sustituidas por las comunicaciones entre las personas (ver sección 4.3.2)

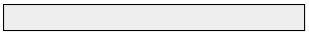
Representación	Descripción
	Grupo de Actividades
	Actividad
	Hito
	Recursos Humanos
	Comunicaciones (entre recursos humanos)

Tabla 4.1: Representación de símbolos del diagrama Gantt extendido

Sin embargo, la configuración del diagrama Gantt extendido también debe determinar las posibles interacciones entre actividades y recursos humanos. A estas interacciones se le han denominado *comunicaciones*, ya que establecen enlaces entre las personas que participan en el proyecto. A continuación se describe con

mayor detalle dónde y cómo encajan tanto los recursos humanos como sus comunicaciones dentro del diagrama Gantt.

4.3.1. Situación de los Recursos Humanos

Según su definición, una actividad es el mínimo componente del trabajo que puede asignarse a una persona [Ins08]. Por lo tanto, la relación entre recursos humanos y actividades es directa, es decir, una persona ejecuta un conjunto de actividades, y una actividad es ejecutada por una o varias personas. A nivel de representación, dichas actividades deben ser adecuadamente representadas y situadas dentro del diagrama, ya que su representación determina la planificación y distribución del trabajo a realizar, pero también establece la aplicación del proceso de desarrollo seleccionado al proyecto software particular. Por ello, en esta propuesta se representa a los recursos humanos como el siguiente nivel de refinamiento de la estructuración del proyecto. Esta representación permite mostrar en un único diagrama una visualización conjunta de la estructura de descomposición del proyecto, la asignación de actividades a las personas, y las tareas asignadas a cada persona.

4.3.2. Comunicaciones entre Recursos Humanos

La representación de los recursos humanos permite refinar las dependencias, pudiendo detallar a la vez las dependencias y las comunicaciones. Así, en nuestra propuesta desaparecen las dependencias entre actividades, y su funcionalidad es sustituida por las comunicaciones. Las comunicaciones entre recursos humanos representan tanto la secuenciación de actividades, como las necesidades de colaboración, entrega de productos de trabajo intermedios, o cualquier otro tipo de comunicación que sea necesario para el desarrollo adecuado de las tareas asignadas. Se pueden distinguir dos tipos de comunicaciones:

- *Comunicación entre actividades*. Son las relaciones de dependencia que se establecen entre dos actividades diferentes. En este tipo de comunicaciones, la relación consiste en un acuerdo para el intercambio de información entre

una persona que finaliza una tarea (dentro de una actividad) y otra persona que necesita el resultado del trabajo realizado en la actividad dependiente, por lo que también se determina la secuenciación de tareas a realizar.

- *Comunicación dentro de una actividad*. Este tipo de comunicación consiste en el intercambio necesario de información entre dos o más personas que están ejecutando la misma actividad, siendo el objetivo la finalización exitosa de dicha actividad.

En la Tabla 4.2 se pueden observar las posibles combinaciones de comunicaciones que pueden darse en la representación de recursos humanos. En esta tabla únicamente se han descrito las posibles interacciones entre dos personas, aunque puede extenderse a comunicaciones entre varias personas. También define un tipo especial de relación en las comunicaciones dentro de una actividad, denominada *comunicación colaborativa*, que permite definir una colaboración estrecha entre dos personas trabajando en común para la ejecución de un conjunto de tareas. Este hecho permite mostrar en la representación propuesta, técnicas como la programación por pares, una técnica utilizada en programación ágil [Mar02].

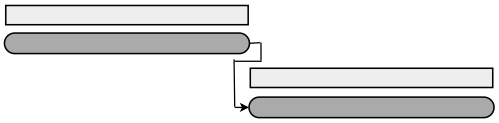
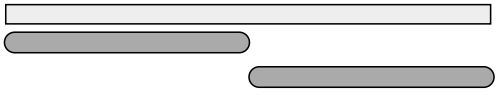
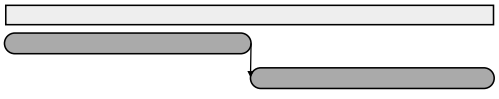
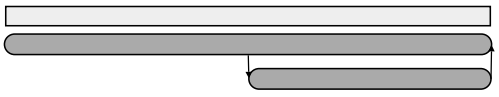
Representación	Descripción
	Comunicación entre actividades
	No hay comunicación
	Comunicación dentro de actividad
	Comunicación Colaborativa

Tabla 4.2: Representación de las comunicaciones en el diagrama Gantt extendido

4.4. Características del Diagrama Gantt Extendido

La representación extendida del diagrama Gantt proporciona unas características adiciones frente al diagrama Gantt tradicional, que se presentan a continuación.

4.4.1. Mayor visualización de la información

El modelo propuesto representa mayor cantidad de información sobre el proyecto que cualquier otra representación sin perder las características deseables de simplicidad, claridad y organización. Por ejemplo, los modelos mixtos como *DesignNet* [LH89] pueden representar mayor número de componentes (como entregables o productos), pero este modelo basado en las redes de Petri y en gráficos *AND/ OR*, no proporciona una visión integrada de la información, y consecuentemente se convierte en un modelo difícilmente extensible a proyectos grandes. En UML [Gro10], las representaciones se centran más en el diseño software que en el proceso del desarrollo, aunque este modelo proporciona una de las mejores soluciones adaptadas al proceso del diseño, ya que proporcionan al desarrollador una manera metódica de mejorar los diseños software.

4.4.2. Visualización de las asignaciones

La extensión del diagrama Gantt con los recursos humanos permite al *project manager* manejar información sobre la asignación, programación, planificación y seguimiento, pero también permite a los desarrolladores observar las posibles interacciones presentes en las tareas asignadas. De hecho, las asignaciones están presente en muy pocas representaciones. Un ejemplo es el modelo RAM (*Responsibility Assignment Matrix*) [RT05], que dispone de manera matricial las actividades en las filas y la organización de personas en las columnas.

La visualización de las asignaciones proporciona una mejora para el control de la planificación, pero también permite realizar un seguimiento del proyecto software, e

incluso para la extracción de conocimiento y la minería de datos, una vez finalizado el proyecto.

4.4.3. Optimización del Proceso

La representación de actividades, recursos humanos y comunicaciones en un mismo diagrama permite la detección de posibles sobreestimaciones y/o solapamientos. Este hecho da lugar a que tanto el proceso de desarrollo, la planificación, o las asignaciones puedan optimizarse para evitar estos problemas.

Por una parte, la optimización puede tener lugar en la secuenciación de tareas, ya que al haberse definido las interacciones a nivel de recursos humanos, se puede visualizar más claramente las dependencias a nivel de tareas, y no a nivel de actividad. Así, por ejemplo, un producto finalizado en una actividad se puede enviar directamente a una actividad dependiente, sin esperar a que la actividad en curso esté finalizada.

Por otra parte, es aconsejable tratar de optimizar las asignaciones y comunicaciones en la medida de lo posible, sobre todo para evitar situaciones de sobreestimación y solapamientos entre los recursos humanos.

4.5. Un ejemplo del Diagrama Gantt Extendido

Para ilustrar la representación de recursos humanos en el diagrama de Gantt, en esta sección se presenta un ejemplo de utilización para un proceso de desarrollo ágil. El proceso se basa en una iteración de una metodología ágil para el desarrollo de un juego de preguntas y de respuestas. En este proceso se distinguen seis fases o grupos de actividades: *Planificación de la iteración*, *Diseño de las unidades de prueba*, *Programación*, *Refactorización*, *Prueba/Verificación* e *Integración*.

Una vez analizadas y verificadas las actividades a ejecutar en cada una de las fases, se realiza la planificación y secuenciación de las actividades en una escala temporal, que da como el resultado el diagrama que se observa en la Figura 4.1.

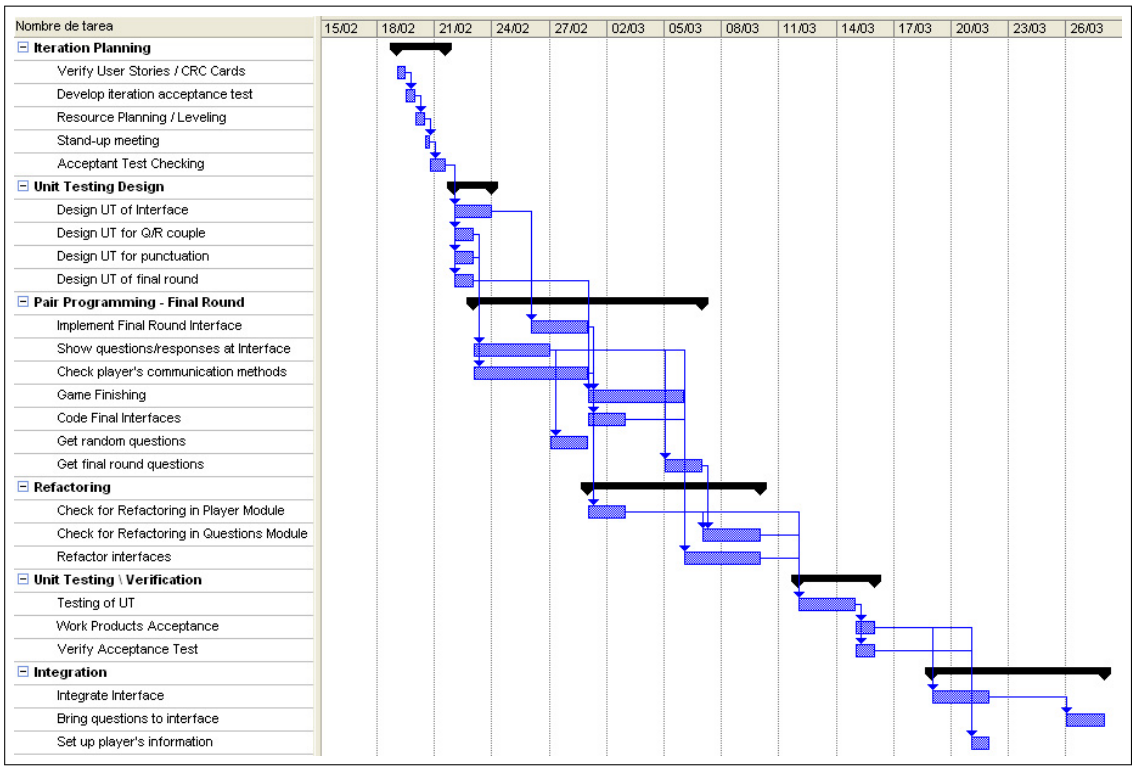


FIGURA 4.1: Diagrama Gantt tradicional para un proceso de desarrollo ágil

En este diagrama Gantt se representa la planificación de las actividades y sus dependencias, que determinan el orden lógico entre la proceso de desarrollo y sus dependencias.

Para extender el diagrama Gantt tradicional con la representación de los recursos humanos es necesario transformar las dependencias entre actividades de la figura 4.1 en dependencias entre recursos humanos, o lo que es lo mismo, en *comunicaciones*. Esto da lugar al diagrama que puede observarse en la figura 4.2.

Para mostrar más detalladamente la nueva representación se ha seleccionado el conjunto de actividades de *Programación*, para mostrar los distintos tipos de comunicaciones presentados en este capítulo, tal y como se expone a continuación:

- Comunicación entre actividades (dependencias entre actividades). Entre las actividades *Code final interfaces* y *Get final round questions* se puede observar una dependencia de este tipo, que puede darse bien por secuenciación

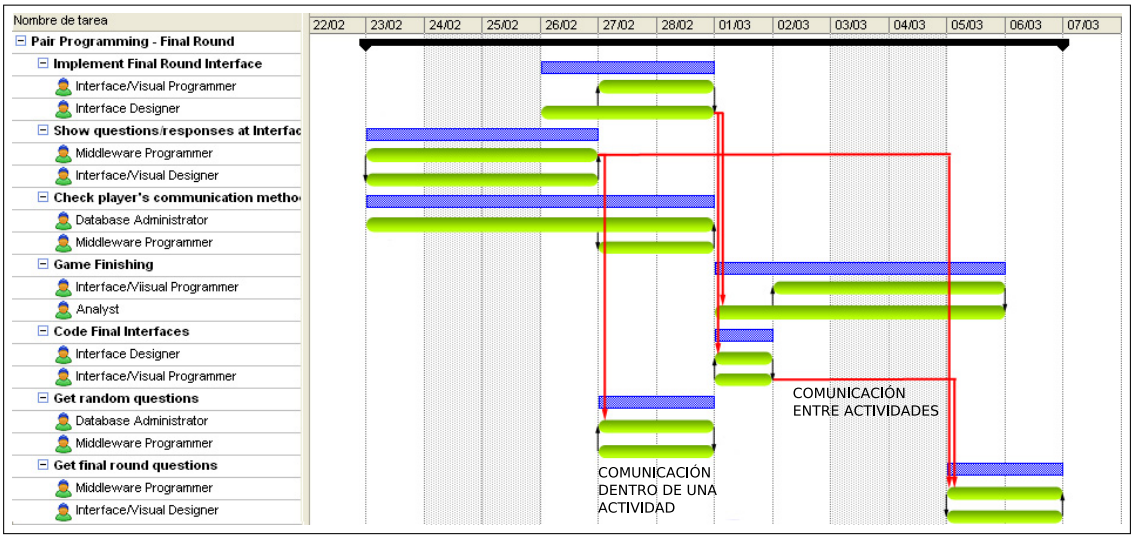


FIGURA 4.2: Diagrama Gantt extendido para la fase de *Programación* de un proceso de desarrollo ágil

lógica de las actividades, pero que también puede incluir el intercambio de productos de trabajo, informes o cualquier otro tipo de entregables.

- Comunicación dentro de la actividad. Este tipo de comunicaciones se producen cuando dos o más personas trabajan dentro de una misma actividad. En la figura 4.2 se pueden observar estas dependencias en la actividad *Get random questions*, donde tanto el analista como el programador visual deben trabajar conjuntamente para la finalización de la actividad. Cuando dicha comunicación implica colaboración (por ejemplo, en el caso de la programación por pares de la actividad *Code final interfaces*) se le denomina *Comunicación colaborativa*.

El diagrama Gantt extendido muestra gráficamente la integración de recursos humanos dentro del conjunto de actividades de la fase de *Programación*, representándose los recursos humanos, y habiéndose transformado las dependencias por comunicaciones. En la figura anterior también puede observarse cómo se muestran los dos tipos de comunicaciones definidos anteriormente: entre actividades, y dentro de actividades.

Al comparar el diagrama Gantt tradicional (Figura 4.1) y el diagrama Gantt extendido (Figura 4.2) se pueden observar los siguientes hechos:

- La planificación de las dos últimas tareas hacen uso de recursos compartidos. Por ejemplo, el administrador de bases de datos está asignado para la realización de dos actividades a la vez. Esto no podía ser detectado en un diagrama Gantt tradicional.
- Usando esta representación, la responsabilidad de la comunicación queda determinada por el sentido de las flechas. Esto quiere decir que para un recurso humano, si se determina que es el origen de una *comunicación*, debe proporcionar algún tipo de entregable al destino de dicha *comunicación*.
- El desarrollo de una actividad puede dividirse en varias fases, como en el caso de la actividad *Game finishing* donde es el analista el responsable de comenzar el desarrollo de la tarea, y posteriormente el programador visual se incorpora, para obtener como resultado una comunicación colaborativa.

Con todo esto, el diagrama Gantt extendido proporciona un refinamiento de la información mostrada, permitiendo a la asignación directa a los recursos humanos, y determinando las necesidades de comunicación entre las personas participantes en el proyecto, y no a nivel de tarea, como en el diagrama Gantt clásico.

4.6. Conclusiones

En este capítulo se han mostrado las carencias que ofrecen los principales modelos de representación para la planificación, sobre todo debido a que estos modelos se centran más en el propio proceso de desarrollo del software (en cómo se ejecutan las actividades) que en quién desarrolla las tareas asociadas al proyecto (los recursos humanos).

La solución propuesta en este capítulo es la extensión del diagrama Gantt para incluir a un nuevo componente fundamental: los recursos humanos. La necesidad de representar este componente viene dada por la importancia de las personas dentro del proceso de desarrollo software, pero también por la necesidad de representar las asignaciones y comunicaciones dentro de un modelo de representación.

Sin embargo, no es necesario desarrollar un nuevo modelo de la representación desde cero, sino que, basándose en las características del diagrama Gantt, se han añadido a este modelo de representación los recursos humanos, dando lugar a una extensión de la representación que proporciona simplicidad, claridad, y organización, características necesarias para una adecuada visualización de la información.

Esta propuesta supone un nuevo enfoque para la planificación y seguimiento de los proyectos software, y abre nuevas vías para definir las asignaciones sin que éstas sean impuestas por las dependencias entre actividades, sino que dichas asignaciones representen a su vez tanto las dependencias, como las comunicaciones.

Este trabajo debería complementarse en el futuro con la implementación de esta propuesta, donde pueda validarse de una manera adecuada el trabajo, y donde se puedan demostrar de manera aplicada los beneficios reales del diagrama Gantt extendido respecto a otros modelos de representación, en un contexto de proyectos reales. En la actualidad realizar dicha experimentación es compleja, debido a que la información de proyecto suele ser por lo general información sensible y bajo restricciones de confidencialidad.

4.7. Objetivos alcanzados en la Primera Parte

Con este capítulo se da por finalizada la primera parte de esta tesis, centrada en un estudio profundo de los diferentes modelos de representación para la gestión de proyectos. En estos tres capítulos se ha realizado una investigación de las representaciones más utilizadas en el contexto de la gestión de proyectos, y en base a dicho estudio, se han obtenido una serie puntos débiles que han servido para el desarrollo de sendas propuestas, tanto en el capítulo 3 con PARMA como en este capítulo con el diagrama Gantt extendido. Tras realizar este estudio se puede afirmar que los modelos de representación para la gestión de proyectos actuales proporcionan una visualización parcial de la información que impide obtener una visión global del proyecto en un momento dado. Cuando el tamaño del proyecto crece y/o cuando nos encontramos en un entorno de desarrollos tecnológicos, con las variables inherentes de variabilidad e intangibilidad, el control sobre la gestión de dicho proyecto software se hace complejo, incluso con las herramientas software

de gestión actuales [MGL11]. Desde el punto de vista del doctorando, las propuestas que se han presentado en los dos últimos capítulos, cubren parcialmente el problema de visualización.

Sin embargo, cuando se trata de proyectos software, uno de los componentes importantes es cómo se representa el conocimiento para la gestión. Para ello, en los siguientes capítulos se va a tratar de las representaciones del conocimiento, es decir, cómo se modelan los procesos asociados a la gestión de proyectos software, haciendo especial hincapié en aquellas propuestas que están estableciendo un estándar *de facto*, bien sea por la definición de los propios procesos asociados, como de las herramientas que utilizan para representar dicho conocimiento y valerse de él.

Capítulo 5

Representaciones del Conocimiento

5.1. Introducción

En la primera parte de esta tesis se han desgranado los diferentes modelos de representación visual existentes para la gestión de proyectos en el ámbito de la ingeniería del software. Sin embargo, la visualización proporciona un enfoque parcial del problema de la gestión de proyectos software. La representación visual solo proporciona una conceptualización sobre la complejidad de los proyectos [GA08]. El problema de la pérdida de información en ese proceso de simplificación ha sido parte del estudio realizado en la primera parte de la tesis.

En esta segunda parte nos centramos en el problema de la complejidad de la gestión de los proyectos. Para ello, se va a hacer uso de las herramientas de gestión del conocimiento para modelar la información de los procesos de gestión. Este capítulo se centra sobre las diferentes representaciones que permiten el modelado de esos procesos, enfocándose la atención sobre aquellos más cercanos a la conceptualización de proyectos software. Así mismo, también resulta necesario conocer el conjunto de lenguajes que permiten expresar el modelado.

5.2. Representación del Conocimiento

Los Sistemas Basados en el Conocimeinto o *Knowledge Based Systems* (KBS) proporcionan una fuerte base para la explotación de datos, el apoyo a la decisión y el aprendizaje automático.

En la actualidad es necesaria la coordinación y cooperación entre las distintas aplicaciones y herramientas que toman parte durante el desarrollo de un proyecto. Para ello, es necesario que todas las aplicaciones puedan entenderse entre sí, esto es, hablar el mismo lenguaje para comunicarse entre ellas. Por lo tanto, el objetivo es alcanzar una interoperabilidad entre las aplicaciones que permitan una representación conjunta del proyecto y todos los factores que le rodean. Durante muchos años se han propuesto distintas representaciones que expresaban términos y conceptos asociados a ciertas áreas del conocimiento.

Sin embargo, durante la última década, y basándose principalmente en dichas representaciones propuestas para el modelado del conocimiento, la investigación se está moviendo hacia la creación de conocimiento con contenido semántico. Este conocimiento semántico se ha desarrollado principalmente gracias a las ontologías, que proporcionan *"una especificación explícita y formal de una conceptualización consensuada"* [SBF98]. Una ontología define los términos y conceptos (con significado) necesarios para describir un área de conocimiento, así como las relaciones que existen entre ellos.

El objetivo de este capítulo es, por una parte, realizar un estudio de los lenguajes de modelado que se utilizan para el capturar el conocimiento sobre un determinado dominio, en nuestro caso la representación de un proyecto software, y por otra parte, analizar los principales modelos de representación que participan en la definición de los procesos de gestión. Este análisis se utilizará para detectar el conjunto de entidades fundamentales que son necesasrias para la creación de un modelo para la representación del proyecto.

5.3. Lenguajes de modelado

Para modelar la información, existen varios enfoques que engloban a la gran mayoría de sistemas de representación de la información. Los diferentes lenguajes de representación están representados en alguno de los enfoques de modelado que se presentan a continuación [OPSE$^+$10]:

- **Lenguaje de marcas**. Este enfoque define una estructura de datos jerárquica consistente en un conjunto de etiquetas que definen atributos y contenido. En particular, el contenido de las etiquetas se define recursivamente utilizando otro conjunto de etiquetas. Las representaciones basadas en los lenguajes de marcas generalmente utilizan un lenguaje derivado de *Standard Generic Markup Language* (SGML) [Con96], el metalenguaje de todos los lenguajes que marcas. El lenguaje más común que se adopta como lenguaje de marcas es *eXtensbile Markup Language* (XML).

- **Modelado Gráfico**. Una de las herramientas de modelado general es el Lenguaje Unificado de Modelado o *Unified Modelling Language* (UML), que tiene un enfofue orientado a sus modelos gráficos denominados diagramas UML, que comprenden trece diagramas que especifican estructura, comportamiento o interacción [Gro10]. La orientación de sus diagramas permite definir estructuras para definir modelos de información. Este enfoque gráfico es especialmente adecuado para el modelado de aplicaciones orientadas a estructuras de base de datos, que puedan derivar en modelos Entidad-Relación (ER) que puedan ser cargados en bases de datos relacionales.

- **Modelos Orientados a Objetos**. El modelado orientado a objetos permite explotar las bondades de este enfoque, que son el encapsulamiento y la reutilización. Delegando la abstración a nivel de objeto, se ocultan la información interna del objeto, proporcionando una serie de métodos que definen la API para dicho objeto.

- **Modelos basados en Lógica**. En los sistemas de programación lógica, la información se representa a través de hechos del mundo real (*facts*), expresiones sobre dichos hechos, y reglas. La mayoría de los lenguajes de modelado

de información se basan en sistemas de lógica de primer orden. De manera general, la lógica define las condiciones a través de las cuales una expresión o hecho puede derivarse (un proceso que se conoce como razonamiento o inferencia), a partir de un conjunto de hechos o expresiones. Para describir las condiciones se aplican un conjunto de reglas formales. El modelado de la información se basa en las reglas que se definan, y el contenido del modelo, expresado a través de expresiones lógicas, varía en función de la ejecución de las reglas. El sistema se nutre de hechos o expresiones que se añaden al sistema, bien directamente como entrada, o bien infiriendo a través del sistema de reglas definido.

- **Modelos ontológicos**. Según la definición formulada por Gruber, una ontología se define como *"una formalización de una conceptualización"* [Gru93]. Desde un punto de vista semántico, las ontologías permiten la descripción de modelos para un cierto dominio usando formalismos explíticos. Esta información explícita es utilizada para la representación y el razonamiento sobre dicha información. Los modelos semánticos representan un enfoque actualmente vigente que permiten la representación de la información, proporcionando herramientas como el mapeado y *matching* de la información, la reutilización y la capacidad de inferencia lógica.

De todos estos enfoques, en la actualidad los modelos ontológicos u ontologías proporcionan un campo de investigación que está siendo aplicado en la gran mayoría de dominios, incluyendo la ingeniería del software. Esto se debe a la potencia que proporciona el modelado de la información en conjunción con la posibilidad de inferencia sobre la información.

Desde el punto de vista formal, una ontología para un cierto dominio se refiere a la terminología (vocabulario del dominio), todos los conceptos esenciales en dicho dominio, su clasificación (taxonomía), sus relaciones (incluyendo todas las jerarquías y restricciones) y los axiomas de dominio [GDD09]. Así, una ontología define lo siguiente:

- Vocabulario: diccionario, glosario (expresado en lenguaje natural), semánticamente independiente del contexto (*smartphone*), unívoco y compartido entre los expertos de dicho dominio.
- Taxonomía: clasificación o categorización jerárquica de las entidades en un determinado dominio (p.ej. un *smartphone* es un disposivo móvil)
- Relaciones: relacionan lógicamente los términos del vocabulario (un *smartphone* tiene la capacidad de llamar a otros teléfonos móviles)
- Axiomas: Restricciones sobre el dominio, cardinalidades y reglas que permiten el razonamiento sobre los hechos descritos en la ontología (inteligencia artificial)

El uso de las ontologías para la representación de la información está siendo generalizado en la gran mayoría de áreas de conocimiento. Dentro de ese creciente interés, es necesario destacar los trabajos realizados en la definición semántica para la gestión conceptual de proyectos [RdA06] [AAH$^+$06] [TYD$^+$09]. Dentro de la Gestión de Proyectos se está empezando a ver la necesidad de plasmar la adquisición de conocimiento implícito a través de herramientas, debido a que afectan directamente a la gestión y desarrollo de los proyecto [**?**].

Sin embargo, el error más generalizado consiste en utilizar las ontologías para el modelado de entidades como si fuera una base de datos. En este sentido, lo importante de las ontologías no consiste en obtener datos estructurados que modelen un cierto conocimiento, sino que dicho conocimiento es compartido entre la comunidad experta. El conocimiento representado en una ontología está en constante evolución y la captura de información asociada a la simple taxonomía y definición de relaciones se completa con la posibilidad de definir axiomas y reglas sobre los datos y realizar consultas e inferencias sobre el contenido de la ontología.

5.4. Lenguajes de modelado semántico

Los modelos semánticos se encuentran actualmente en plena vigencia como uno de los temas a investigación en muchas de las áreas de conocimiento. Esto se debe

en gran medida a que proporciona una representación expresiva del conocimiento y a que permite realizar consultas y razonamiento sobre dicho conocimiento [WDC+04]. En la actualidad no son pocos los proyectos de investigación y aplicaciones reales que hacen uso de las capacidades que proporcionan las ontologías. Una de las áreas de aplicación donde las ontologías forman parte como herramienta para la investigación de dominio es en medicina, donde se pueden encontrar proyectos como OBO [Cla11b] o NCI Thesaurus [Ins11], que tienen destacados avances gracias a la conceptualización semántica. Otro ejemplo de éxito es WordNet [Uni11], que sirve de repositorio para realizar mapeado y *matching* de ontologías. En este último caso, WordNet está sirviendo de herramienta a muchos investigadores para aprendizaje automático, clasificación, minería de datos y apoyo a la decisión [KS03, Noy09]. Otro de los beneficios que proporcionan los modelos semánticos es la interoperabilidad entre sistemas que utilicen el mismo lenguaje de representación, sobre todo si se ha producido el mapeado de información. Pero quizá la característica más representativa de los lenguajes semánticos es permitir definir la consulta y el razonamiento automático de manera expresiva, permitiendo obtener nueva información inferida que puede ser utilizada por las aplicaciones.

Durante la última década, el W3C ha estado desarrollando especificaciones para definir lenguajes que den soporte a los enfoques semánticos. *Resource Definition Framework* (RDF) es la primera especificación que ha dado soporte a lo que hoy conocemos como la Web Semántica. Con el objetivo de incrementar la expresividad y la capacidad de razonamiento, se siguieron desarrollando lenguajes, ahora recomendaciones por el W3C, que explotan el modelado y permiten el razonamiento, como *Ontology Web Language* (OWL) [BvHH+04], o más recientemente OWL 2 [Gro09b].

5.4.1. Resource Description Framework (RDF)

Las tecnologías semánticas se pueden ver como un *framework* definido por capas, cuyas capas inferiores proporcionan los formatos de intercambio de datos, tanto a nivel sintáctico como semántico, mientras que las capas superiores definen ontologías completas, incluyendo axiomas y reglas, en caso de ser necesario (ver figura 5.1). En particular, RDF es un lenguaje que fue originalmente creado para

la representación de información de recursos disponibles en la Web [KC04]. Con la generalización del concepto de un recurso Web, RDF también puede utilizarse para representar información sobre recursos que no pueden ser obtenidos directamente a través de la Web, como propiedades o literales. RDF proporciona un *framework* para expresar la semántica de la información de tal manera que pueda ser intercambiada entre diferentes aplicaciones sin pérdida de significado. Para identificar recursos, RDF utiliza identificadores denominados URI (*Uniform Resource Identifier*). Cualquier recurso puede ser descrito usando simplemente propiedades y relaciones.

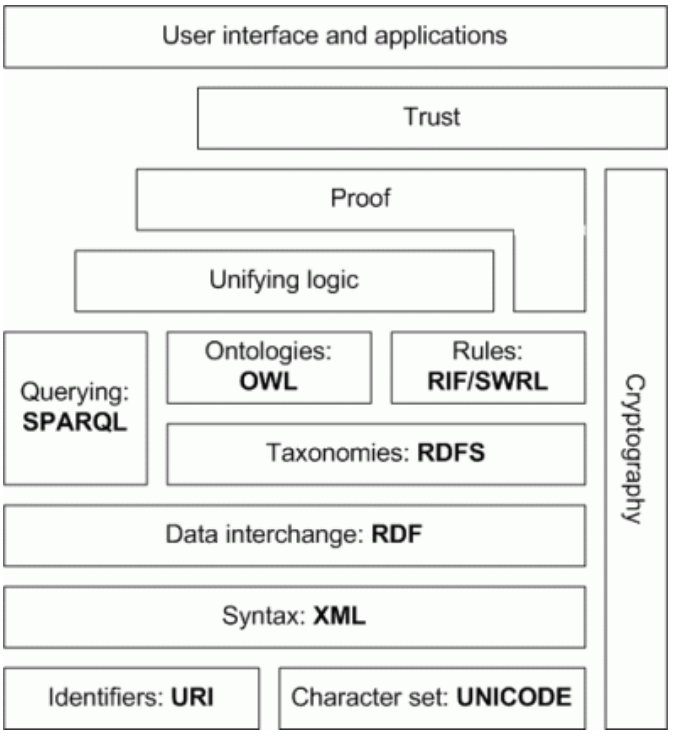


FIGURA 5.1: Arquitectura de Capas definida para la Web Semántica

De una manera más formal, RDF representa tripletas que se pueden interpretar gráficamente como grafos dirigidos, donde se define un sujeto (un recurso identificado por una URI), un predicado (la relación o gráficamente el arco) y el objeto (que puede ser un recurso o un literal). Por ejemplo, para una expresión como *"Dave Beckett is the editor of the resource http://www.w3.org/TR/rdf-syntax-grammar"*, se puede representar como el grafo que se puede observar en la figura 5.2.

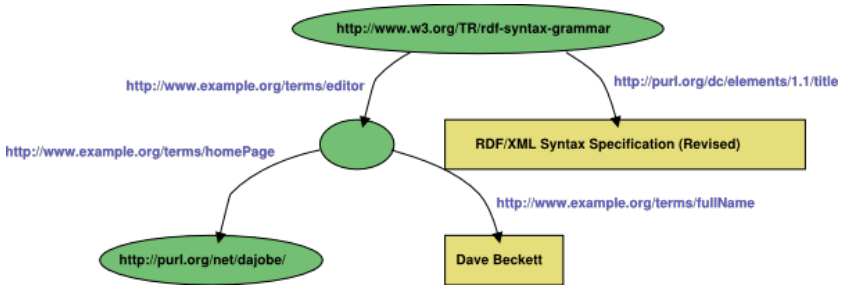


FIGURA 5.2: Ejemplo de representación de grafos en RDF

Para la codificación de las expresiones en RDF en formato máquina, RDF utiliza un proceso de serialización basado en XML [BPSM+08]. Las propiedades RDF pueden ser atributos de un recurso, o bien puede representar una relación entre dos recursos. Sin embargo, RDF no proporciona un mecanismo para describir estas propiedades ni mecanismos para describir las relaciones entre las propiedades y los recursos. Para ello, se definieron en su momento varias extensiones como *RDF-Schema* (RDF-S) [BGM04], un lenguaje de descripción de vocabulario donde se definían clases y propiedades que permitían la definición de clases, propiedades y otros recursos, como el rango y dominio de las propiedades.

5.4.2. Web Ontology Language (OWL)

El nivel inmediatamente superior sobre RDF para la arquitectura de la web semántica debía ser un lenguaje de definición de ontologías que pudiera describir formalmente el significado de la terminología utilizada para los "recursos Web" definidos anteriormente. El *Web Ontology Language* (OWL) [BvHH+04] resultó ser la respuesta para proporcionar un vocabulario expresivo para la declaración de propiedades y clases, y los axiomas sobre elementos, como cardinalidad, simetría, equivalencia, etc.

Una de las características que es inherente a OWL es la utilización de las bases de la Lógica Descriptiva (*Description Logics* –DL–) para la representación de conocimiento en un cierto dominio de aplicación [BCM+07]. DL permite definir descripciones de conceptos y relaciones entre dichos conceptos.

OWL se divide en tres sublenguajes, según las restricciones y expresividad que se quisiera definir:

- OWL–Lite. Determina un conjunto mínimo de elementos del lenguaje, y por lo tanto su expresividad es limitada.
- OWL–DL. Proporciona la máxima expresividad pero sin perder completitud computacional (todas las aserciones son computadas) y decidibiliad de los motores de razonamiento (todas las computaciones terminan en un tiempo finito).
- OWL–Full. Extiende la expresividad definiendo nuevos elementos del lenguaje, pero pierde decidabilidad.

OWL utiliza como base RDF (*Resource Description Framework*), que permite la definición de todos esos elementos del lenguaje OWL descritos en su especificación [PSHH04]. OWL DL proporciona una expresividad en lógica descriptiva $\mathcal{SHOIN}^{(\mathcal{D})}$, que permite expresar transitividad de roles ($\mathcal{S}$), jerarquía de roles ($\mathcal{H}$), nominales o enumerados ($\mathcal{O}$), propiedades inversas ($\mathcal{I}$) y restricciones de cardinalidad ($\mathcal{N}$), Además, permite la definición de propiedades de datos y sus tipos ($^{(\mathcal{D})}$).

En la actualidad, OWL 2 [Gro09b] se han convertido en la recomendación del W3C para la descripción de conocimiento con contenido semántico. OWL 2 extiende la especificación descrita para OWL 1, proporcionando claves, un mayor rango de propiedades de datos, ampliando la información de cardinalidad, proporcionando propiedades asimétricas, reflexivas y disyuntivas, y proporcionando la posibilidad de añadir anotaciones.

5.5. Modelos conceptuales para la gestión de proyectos

Una vez conocidos los lenguajes de modelado, nos centramos en el análisis de los modelos que permiten definir los procesos que describen a un proyecto. Así, se han

analizado las siete representaciones más significativas para el modelado de procesos y planes de proyectos, habiendo seleccionado aquellas que mayor aplicabilidad tienen para la representación de procesos orientados a la gestión de proyectos software. Las representaciones a estudio son las siguientes: BMO (*Business Management Ontology*), el formalismo Act, ⟨I-N-OVA⟩, SPAR (*Shared Planning and Activity Representation*), O-Plan, PSL (*Process Specification Language*) y SPEM 2.0 (*Software & Systems Process Engineering Metamodel specification*). Estas representaciones se pueden categorizar a su vez en lenguajes de representación del conocimiento (como el Act, PSL, O-Plan o ⟨I-N-OVA⟩), y en modelos u ontologías de representación de procesos y planes (BMO, SPAR y SPEM).

A continuación se muestran todas estas representaciones del conocimiento, describiendo para cada una de ellos sus principales características, el contexto de utilización y una descripción de cada uno de los modelos.

5.5.1. Business Management Ontology

BMO (*Business Management Ontology*) es un modelo de información basado en una ontología abierta y altamente semántica. Entre las entidades que modela BMO se encuentran los recursos humanos (*personas*) y materiales (*assets*), la organización, los roles, las competencias y privilegios de roles, las tareas, los documentos de negocio, o entidades de información a largo plazo (políticas, estándares, etc.). [Jen03][Jen04]

BMO utiliza un enfoque ontológico, esto es, la utilización de los procesos de negocio proporciona tanto un valor sintáctico (proporcionado por el vocabulario de conceptos y términos de los procesos de negocio) como un valor semántico, que dota de significado a través de propiedades y relaciones entre dichos conceptos y términos.

Para definir BMO se hace uso de una serie de estándares abiertos que proporcionan y aseguran una adecuada definición de la ontología sobre el modelo de negocio, entre los que se encuentran BPMN (*Business Process Modeling Notation*), ebXML BPSS (*Business Process Specification Schema*) y UBL (*Universal Business Language*) [Jen03].

Para definir la ontología completa sobre el proceso de negocio, y que ésta pueda ser extensible a distintas áreas de conocimiento, BMO se basa en la utilización de varias ontologías. De este conjunto de ontologías se utilizarán únicamente aquellas partes que sean fundamentales para definir el modelo de negocio a modelar. En la actualidad, BMO está formada por unas 40 ontologías [Jen04], que se pueden categorizar en tres grupos o capas:

- *Core business ontology layer* (BMO–Core). Define los principales conceptos y términos comunes a todas las áreas de conocimiento, y que proporcionan la base o núcleo de cualquier negocio.
- *Industry specific ontology layer*. Cada área de conocimiento necesita extender y adaptar los conceptos neutros de la ontología núcleo BMO–Core.
- *Organization specific ontology layer*. Cada organización puede definir sus propios conceptos y términos para modelar de manera más realista la actividad del negocio.

En este caso, la unión de la ontología-núcleo (BMO–Core) con las demás ontologías forma el modelo de dominio necesario para definir los procesos, tareas, recursos, planes, y documentos asociados al negocio del área de conocimiento que quiera representarse.

Esta ontología proporciona también una taxonomía asociada al modelado de procesos de negocio, esto es, un lenguaje (vocabulario) y una estructura jerárquica, con las relaciones entre términos y conceptos. La base de conocimiento viene dada por la instanciación de dicha ontología para el modelado de negocio.

En la actualidad, BMO comprende más de 650 clases, que están disponbiles a través de [Gmb04] . BMO permite al analista:

- definir procesos de negocio privados
- definir procesos de negocio públicos o colaborativos
- definir entidades de negocio

- definir objetos de negocio
- definir servicios que implementen actividades del proceso

5.5.2. Formalismo *Act*

Act es un modelo formal para la representación del conocimiento de planes de proyectos, y tiene dos objetivos fundamentales: la generación de planes complejos; y la ejecución reactiva de planes en entornos dinámicos [MW97a]. El componente básico de este formalismo es el *Act*, que describe los planes a ejecutar y las acciones a realizar. Un *Act* puede ser dividido jerárquicamente hasta encontrar un nivel de definición suficiente para planificar/ejecutar las acciones a realizar. El propósito final de un *Act* es, o bien, satisfacer una meta o condición, o bien, responder a algún tipo de evento que ha modificado el entorno. Un *Act* se describe en función de un conjunto de propiedades o *slots* que describen las condiciones necesarias para la ejecución del plan: objetivos, condiciones de entorno, recursos, etc. Los slots que pueden definirse para un *Act* son:

- *Name*. Nombre del Act e identificador único.
- *Comment*. Documentación descriptiva asociada al slot.
- *Cue*. Describe el propósito final del Act.
- *Precondition*. Describe las restricciones del entorno que deben cumplirse para que pueda ejecutarse el Act.
- *Setting*. Determina metapredicados tipo Test adicionales para la aplicación en Acts.
- *Resources*. Determina los recursos va a necesitar el Act durante su ejecución.
- *Properties*. Determina pares propiedad/valor que no pueden expresarse mediante las propiedades anteriores.

Las propiedades *Cue*, *Precondition*, *Setting* y *Resources* se denominan *Gating slots*, y especifican mediante expresiones objetivo (*Goal expressions*), condiciones que

deben ser cumplidas para la ejecución del Act. La representación gráfica de un Act se realiza mediante los *plots*, que muestran de manera gráfica un conjunto de acciones y subobjetivos parcialmente ordenados.

Las expresiones objetivo describen requisitos en el proceso de planificación y ejecución, y los estados que pueden alcanzarse. Estas expresiones objetivo se definen a través de metapredicados, que describen metas y hechos del sistema, y pueden ser del tipo *Achieve*, *Achieve-by*, *Achieve-All*, *Wait-Until*, *Test*, *Conclude*, *Retract*, *Require-Until* o *Use Resource* [MW97b]. Los cuatro primeros (*Achieve*, *Achieve-by*, *Achieve-All* y *Wait-Until*) determinan acciones para cumplir un objetivo (*goal*) bajo ciertas restricciones (en el caso de los tres últimos). Los metapredicados *Test* se utilizan como prueba condicional para verificar si un Act puede ejecutarse. El metapredicado *Conclude* describe cambios en el mundo causados por la ejecución de una acción. Los metrapredicados del tipo *Retract* especifican en los plots acciones de borrado de fórmulas en la base de datos del sistema. El metapredicado *Require-Until* determina condiciones activas durante toda la ejecución.

5.5.3. <I-N-OVA>

El modelo <I-N-OVA> es un lenguaje para la representación de tareas, planes, procesos y actividades, que puede ser convertido a un lenguaje expresado en lógica de primer orden. El modelado de actividades, procesos y planes se realiza a través de restricciones sobre el comportamiento [Tat95]. Este modelo de restricciones tiene como objetivo la ejecución en un contexto de agentes que controlen las variables del entorno de ejecución.

En <I-N-OVA> los planes se especifican a través de un subconjunto de actividades que modelan el comportamiento del mundo real. Dichas actividades pueden transformarse en eventos, si el entorno no está controlado por agentes. La construcción del modelo se realiza de manera incremental, de tal manera que en cada nivel de refinamiento se pueda definir completamente el mundo a modelar. Para especificar las actividades de este modelo, se utilizan un conjunto de restricciones, que pueden ser de los siguientes tipos:

- Implícitas o *Issue* (I). Proporciona las restricciones evidentes para el modelado del entorno sobre el que se va a desarrollar la ejecución.
- Actividades o Nodo (N). En este conjunto se encuentra la única restricción que proporciona una modificación sobre la topología del modelo. Así, la restricción *'include activity'* (incluir actividad) es la única que puede añadir nodos al modelo.
- Restricciones detalladas o de Orden / Variables / Auxiliares (OVA). Son el conjunto de restricciones que no son implícitas ni que añaden nodos (o actividades). En este sentido, las restricciones pueden ser de tipo Temporal u de Orden (O), cuando es necesario definir una restricción sobre la duración o el orden de ejecución de las tareas, de tipo Variable (V), cuando entran en juego factores del entorno, y por lo tanto, dichas restricciones deben aplicarse sobre eventos, o del tipo Auxiliar (A), que nos permite definir el resto de restricciones (sobre recursos, autoridad, estados, espaciales, etc.).

Este modelo permite la integración de distintas perspectivas en la representación de planes y procesos. El modelado utilizando únicamente restricciones sobre el entorno se complementa perfectamente con otras representaciones de planes y procesos, como por ejemplo, O-Plan.

5.5.4. O-Plan

O-Plan es un formalismo para la representación de planes y tareas, independiente del dominio [CTB91]. El conocimiento se representa a través del Formalismo de Tarea (*Task Formalism*). Dicho conocimiento corresponde a una descripción detallada de las posibles actividades (acciones o eventos) u operaciones dentro del dominio o área de aplicación. El Formalismo de Tarea se especifica a través de un esquema, que recoge las diferentes posibilidades y refinamientos posibles de las actividades y planes a ejecutar. Las actividades pueden ser acciones, que estarán controladas por un agente, o eventos, que surgirán del dominio de aplicación. Estas actividades podrán dividirse hasta llegar a un nivel denominado primitivo o atómico, que no acepta un nivel de refinamiento o división mayor.

Para describir el Formalismo de Tarea, O-Plan utiliza el esquema, que describe acciones, eventos y las reglas que definen la división jerárquica. Los cuatro tipos de esquemas que hay son normal (modela acciones), procesos (modelan eventos) o esquemas de tarea (que modelan tareas en general) y metaesquema, que proporciona una composición entre los tres tipos anteriores.

El formalismo de tarea proporciona una descripción general jerárquica especificando el conjunto de posibles actividades a realizar dentro de dicho dominio de aplicación, y describiendo como esas tareas pueden expandirse dentro de conjuntos de subactividades con sus correspondientes restricciones. Los planes se crean utilizando el conjunto adecuado de actividades (y subactividades), donde así mismo se determinan las restricciones de dependencia, de recursos o temporales.

Además de la representación de tareas, planes, actividades procesos, otro conocimiento que se captura dentro del formalismo de tareas son los efectos y condiciones de las acciones, relaciones jerárquicas, requisitos temporales, autoridad, necesidades de recursos, etc.

5.5.5. SPAR

La información sobre los procesos, actividades, recursos, planes, gestión debe ser un recurso que debe ser interoperable con varias aplicaciones que se utilizan durante el desarrollo de un proyecto, que permita la cooperación y coordinación entre las distintas organizaciones existentes. SPAR (*Shared Planning and Activity Representation*) es un modelo compartido para la representación de planes, procesos o actividades, de tal manera que el conocimiento sobre la organización pueda ser compartido y utilizado de manera efectiva. SPAR proporciona una estructura cuyo objetivo principal es la representación de las entidades principales y sus relaciones para la creación y utilización de representación de planes y actividades [Tat98]. El modelo SPAR se basa principalmente en KRSL (*Knowledge Representation Specification Language*) para el modelado de planes y actividades, donde se detallan las siguientes entidades:

- Plan. Corresponde a una especificación de una actividad para el cumplimiento de uno o más objetivos.
- Especificación de una actividad. Determina o describe una o más actividades.
- Agentes. Puede ejecutar actividades o alcanzar objetivos.
- Objetivos. Se basan en la evaluación de uno o varios criterios.
- Estado. Una actividad puede cambiar su estado, que puede ser evaluado como cumplido o no.

5.5.6. PSL

PSL (*Process Specification Language*) es un lenguaje para la representación neutral de datos de productos para procesos de fabricación [SGCL99]. El principal objetivo de PSL es la búsqueda de un lenguaje común que permita la interoperabilidad entre las distintas aplicaciones que se utilizan en los procesos de fabricación. Así, aplicaciones basadas en la gestión de proyectos (Primavera, MS Project), planificación de la producción, workflow, BPR (*Business Process Reengineering*) o la planificación del proceso podrían compartir una información común sobre el proceso [CGSL03].

Además, PSL proporciona un motor de razonamiento utilizando un conjunto de cuatro entidades (que forman parte de PSL–Core): actividades; ocurrencias de actividades, puntos de tiempo, y objetos. Las actividades pueden tener múltiples ocurrencias, aunque también pueden no ocurrir.

La ontología PSL engloba un conjunto de teorías de primer-orden que se divide en tres partes:

- *PSL–Core*. Es el conjunto de axiomas basado en primitivas semánticas intuitivas.
- *PSL outer core*. Forman una extensión a PSL–Core, tanto a nivel axiomático (definiendo nuevas entidades como Extensión a Subactividad, Extensión a Ocurrencia de Actividad o Extensión a Estados), como a nivel de relación.

- *PSL extensions*. Proporcionan módulos adicionales a PSL, sobre todo para ganar en expresividad y poder modelar las diferentes casuísticas que pueden darse en los procesos de fabricación.

La principal aportación de PSL es que proporciona un lenguaje independiente del dominio, esto es, se crean modelos de proceso para cada una de las aplicaciones utilizadas en el proceso de fabricación, para después transformarlos (a través de wrappers específicos para cada apliación) al lenguaje PSL, que permita la interoperabilidad entre las distintas aplicaciones, y realizando el proceso inverso, esto es, parseando desde el lenguaje PSL a un lenguaje comprensible por la aplicación. Esto permite un intercambio de información entre las distintas aplicaciones utilizadas en el proceso de fabricación, pero también disponer de un lenguaje expresado en lógica de primer orden que permite expresar planes, procesos, actividades y restricciones.

5.5.7. SPEM 2.0

El metamodelo de procesos de ingeniería software y de sistemas o *Software & Systems Process Engineering Meta-model* (SPEM) permite el modelado de procesos software usando estándares del OMG como MOF (*Meta-Object Facility*) y UML (*Unified Modelling Language*), haciéndo posible la representación de procesos software utilizando herramientas compatibles con UML [RGS10].

Cuando se utiliza el estándar SPEM 2.0, los procesos pueden definirse bien utilizando el perfil UML, o bien el metamodelo. Un perfil UML define una serie de metáforas para la representación de conceptos asociados a los proyectos software. Desde el punto de vista del meta-modelo, los procesos SPEM incluyen la semántica del metamodelo MOF y es posible automatizar la transformación entre diferentes representaciones, usando MOF como base para MDA (*Model Driven Architecture*).

La especificación SPEM define un total de siete paquetes que permiten la definición de los procesos de desarrollo software y de sistemas y de sus componentes (ver Fig. 5.3). Estos procesos son [OMG08]:

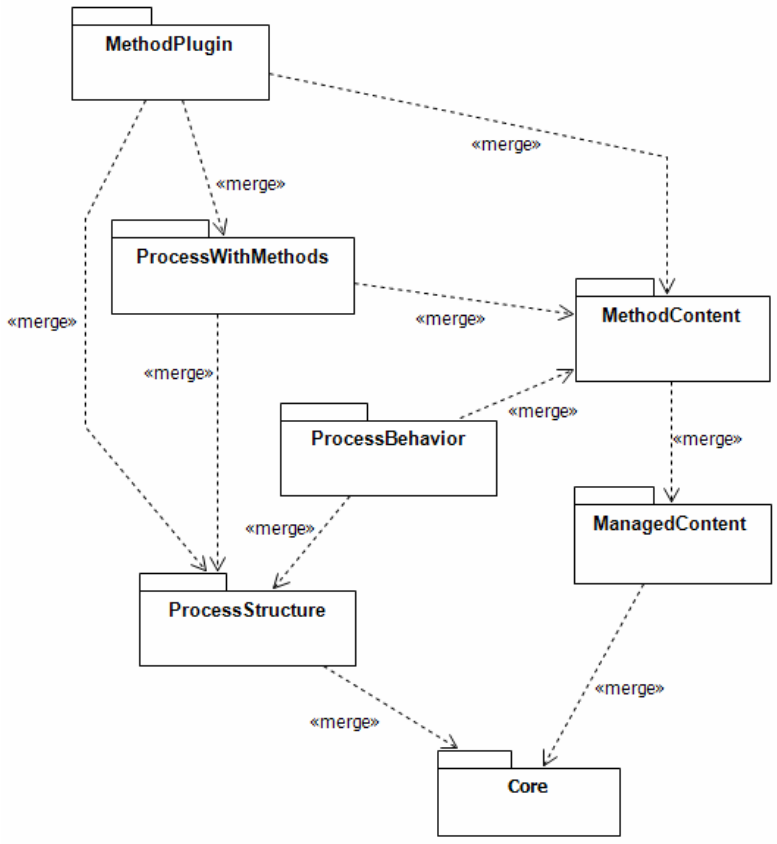


FIGURA 5.3: Estructura del metamodelo SPEM 2.0

- ***Core***. El paquete del metamodelo *Core* contiene aquellas clases y abstraciones del metamodelo que constituyen la base para las clases de todos los demás paquetes del metamodelo.

- ***Process Structure***. Este paquete define la base para la creación de todos los modelos de procesos.

- ***Process Behaviour***. Los conceptos definidos en el paquete *Process Structure* representan un proceso como una estructura estática, permitiendo la concatenación de actividades y la definición de dependencias entre ellos. El paquete del metamodelo *Process Behaviour* permiten la extensión de este metamodelo con modelos de comportamiento.

- ***Managed Content***. Los procesos de desarrollo no se representan únicamente como modelos, sino que son complementados con documentación y gestión descrita en lenguaje natural.

- ***Method Content***. El paquete del metamodelo *Method Content* proporciona aquellos conceptos relacionados con los usuarios y las organizaciones, para construir una sistema de desarrollo basado en el conocimiento independiente de cualquier definición de proceso o proyecto de desarrollo. En este paquete se añaden los conceptos para la definición del ciclo de vida y los elementos que proporcionan una base para documentar el conocimiento sobre los métodos de desarrollo software, las técnicas y usos concretos de buenas prácticas.
- ***Process With Methods***. El paquete del metamodelo *Process With Methods* define nuevas estructuras y redefine estructuras ya definidas para la integración de los procesos definidos en los conceptos del paquete del metamodelo *Process Structure* con instancias sobre los conceptos del paquete del metamodelo *Method Content*.
- ***Method Plugin***. El paquete del metamodelo *Method Plugin* determina conceptos para el diseño y la gestión de librerías o repositorios de métodos y procesos.

Los elementos básicos del paquete del metamodelo *Method Content* incluyen:

- Tareas: Unidades de trabajo atómicas que se componen de una serie de pasos.
- Roles: Conjunto de competencias, habilidades y responsabilidades asociadas a una persona o grupo de personas.
- Productos de Trabajo. Artefactos, entregables o resultados.
- Elementos de guía. Proporcionan información adicional relacionada con otros elementos.
- Categorías. Se utilizan para organizar y crear jerarquías de elementos.
- Asociaciones. Asocian los elementos definidos en el metamodelo.

Por una parte, a los elementos que son instanciados dentro de un proceso en particular se les añade el sufijo *"Use"*. Así, podemos tener *Task Use* para la definición

de una tarea, *Work Product Use* para la definición de un artefacto, o *Role Use* para la definición de un rol a utilizar dentro de un proceso. Hay que tener en cuenta que la notación se refiere a la utilización de un elemento genérico en una definición de un proceso, y por lo tanto, no pertenece a ningún proyecto en concreto.

Por otra parte, tenemos una serie de procesos que se definen dentro del paquete *Method Content* y que se combinan para la definición de actividades y procesos. Los elementos básicos de un proceso son:

- Definición de Trabajo. Es un concepto abstracto que generaliza cualquier tipo de actividad a realizar.
- Elemento de Trabajo. Define una generalización abstracta para todos los demás tipos de elementos de procesos, como parámetros del proceso, realizadores del proceso, elementos de descomposición del trabajo y secuenciación del trabajo.

SPEM estandariza y formaliza la manera de representar los procesos software en relación tanto a su parte estática como dinámica, como actividades, roles, tareas o resultados [RGS10].

5.6. Elementos para el modelado del conocimiento en la gestión de proyectos software

Por una parte, el modelado de los proyectos software proporciona una ayuda para el diseño y mejora de los procesos de desarrollo. No obstante, dichos procesos no quedan definidos de una manera completa y/o estructurada.

Por otra parte, la interoperabilidad entre aplicaciones permitiría proporcionar una visión integrada de la información y el intercambio de información entre las distintas aplicaciones que se pueden utilizar durante el desarrollo y gestión de un proyecto software. La principal problematica que plantea la actual representación de los procesos software y de cómo se gestionan éstos, es que la información raramente está integrada, sino que generalmente está diseminada entre las distintas

aplicaciones y servicios que proporcionan una visión parcial de la información del proyecto.

Sin embargo, gracias a los lenguajes de representación de conocimiento y ontologías expuestos en la sección 5.4, se está alcanzando el objetivo de interoperabilidad entre las distintas áreas de conocimiento. No obstante, aún no se dispone de una ontología o modelo de representación de conocimiento que permita expresar completamente el proceso de gestión de un proyecto software.

Para poder describir un lenguaje para la representación en la gestión de proyectos software es necesario determinar el conjunto de entidades que forman parte del proyecto y participan de forma activa en él. Por ello, se debería analizar el conjunto de entidades y parámetros que mayor influencia tienen en las aplicaciones utilizadas para la gestión de proyectos, teniendo en cuenta las estrategias de modelado y los modelos analizados en la sección anterior.

Para saber los principales requisitos a la hora de definir un modelo para la gestión de proyectos software, nos basamos en las principales áreas de conocimiento descritas en el *Project Management Body of Knowledge* (PMBOK) [Ins08]. En la Tabla 5.1 se describe la asociación realizada entre cada área de conocimiento y las entidades fundamentales que se pueden derivar.

Área de Conocimiento PMBOK	**Entidad Fundamental**
4. Project Integration Management	Proceso y Objetivos
5. Project Scope Management	Actividades y Productos
6. Project Time Management	Objeto de tiempo
7. Project Cost Management	Objeto de coste
8. Project Quality Management	Actividades
9. Project Human Resource Management	Recursos
10. Project Communications Management	Comunicaciones
11. Project Risk Management	Actividades
12. Project Procurement Management	(ninguno)

Tabla 5.1: Entidades para definir una representación del conocimiento para la gestión de proyectos software

Tras analizar los procesos definidos en el PMBOK, se obtienen un conjunto de nueve entidades que pueden modelar la gestión de un proyecto. Estas entidades

son: *proceso*; *actividad*; *objeto de tiempo*; *objeto de coste*; *recursos*; *comunicaciones*; *productos*; *objetivos*; y *objetos*. Los primeros ocho artefactos modelan todos los componentes que se consideran fundamentales para la gestión de un proyecto software. El artefacto *objeto* se ha reservado para definir el conjunto de componentes que no se pueden categorizar en ninguna de las entidades fundamentales, pero que son necesarias para el desarrollo y gestión del proyecto, junto con las entidades fundamentales.

Así, por ejemplo, los riesgos de un proyecto generalmente no toman parte activa de un proyecto, sino que únicamente se valoran cuando surgen. Su modelado como *objeto* permite tenerlo en cuenta durante el proceso de desarrollo y gestión, aunque su definición, control y verificación corresponderá con un conjunto de actividades que quedarán determinadas y planificadas. Así mismo, sucede con los contratos, documentos, informes, gráficos, planes del proyecto, formularios, estándares, plantillas y demás entregables que forman parte material del proyecto.

5.6.1. Procesos

Un proceso software es un conjunto de prácticas secuenciales (o iterativas) que son funcionalmente coherentes y reutilizables para la gestión, organización e implementación de proyectos software [WK00]. Si se desea controlar un proyecto es necesaria la verificación de que el proceso se ha especificado de una manera correcta y adecuada. Además, la información sobre el proceso también proporciona una medida para la mejora de dicho proceso.

5.6.2. Actividades y Tareas

Las actividades o tareas constituyen el núcleo unitario de cualquier proyecto. Para cada actividad o tarea es necesario disponer en cada momento de la máxima información posible, incluyendo descripción, objetivo, prioridad, estimaciones, dependencias con otras actividades, etc. Las actividades generalmente implementan los distintos procesos: desarrollo, gestión, aseguramiento de la calidad, verificación, etc.

5.6.3. Objetos de Coste y Tiempo

Una de las tareas que se realiza en las fases iniciales de cada proyecto es la planificación, ya que permite realizar un cálculo estimado del proyecto. Por ello, se han definido las entidades de objeto de coste y objeto de tiempo, que permiten definir las distintas planificación y presupuesto para un proyecto.

Estos objetos provienen de la definición de planes realizadas en las ontologías a estudio previamente analizadas, como PSL, O-Plan o BMO. Esto permite definir las actividades en función de parámetros temporales y de coste, lo que determina, por una parte, la definición de una secuenciación de las actividades, y por otra, la posibilidad de ir refinando las planificaciones y presupuesto, según avance el proyecto o se definan de manera detallada las actividades.

5.6.4. Recursos y Comunicaciones

Sin recursos es imposible realizar un proyecto. En proyectos software, cuando se habla de recursos generalmente suele referirse a los recursos humanos, ya que los recursos materiales se dan generalmente por supuesto y suelen estar cubiertos o planificados como coste al inicio del proyecto (servidores, equipos, etc.).

Es necesario que cada recurso quede definido durante el proyecto, ya que sus parámetros (especialización, habilidades, formación, participación en proyectos pasados, rendimiento, etc.) proporcionan una información valiosa para la determinación de las asignaciones, la planificación, la organización o la duración de las actividades.

Así mismo, a la vez que se determinan los recursos, es necesario dotar de vías de comunicación a éstos. La definición de comunicaciones debe basarse en el estudio de las prácticas habituales en el desarrollo de los proyectos software, pero también de cara a pensar en mejoras de estas vías de comunicación, que apoyen una mejora en la eficiencia de los proyectos.

Por ello, una parte fundamental de un proyecto es la organización y las vías de comunicación que pueden establecerse, ya que una organización inadecuada tiene

mayor probabilidad de retrasar la duración del proyecto, mientras que las comunicaciones inadecuadas están más relacionadas con una interpretación inadecuada de los requisitos, y la implementación incorrecta de las funcionalidades asociadas.

5.6.5. Objetivos y Productos

Un proyecto se crea principalmente para crear un producto y servicio único. Para ello, es neceario definir un conjunto de objetivos que permita verificar el avance del proyecto, y en su caso, darlo por terminado (y obtener el producto objetivo).

5.7. Conclusión

En los proyectos software existen en la actualidad un pequeño conjunto de herramientas que permitan la colaboración y coordinación entre las distintas aplicaciones que forman parte del desarrollo y de la gestión. Sin embargo, cada vez más, las empresas desarrolladoras de software desean que la información de proyectos y gestión esté integrada, teniendo una visión única de la información que puedan utilizar todas las aplicaciones. Pero esta interoperabilidad aún no es posible en la mayoría de los casos, ya que no existe aún una versión unificada de lenguaje de intercomunicación.

Por otra parte, las actuales ontologías de alto nivel son demasiado genéricas para que puedan proporcionan un lenguaje altamente semántico y especializado, que sirva de comunicación entre los distintos sistemas y aplicaciones.

En este capítulo se ha realizado un análisis de las principales representaciones y ontologías para proyectos en distintas áreas de conocimiento y de los lenguajes de modelado que permiten expresar dichas representaciones. Así, se pueden diferenciar tanto representaciones genéricas –por ejemplo, BMO–, como soluciones particulares en áreas de conocimiento concretas –como los procesos de fabricación en PSL–. Sin embargo, se ha visto que hay una necesidad de ampliar la definición que se realiza de un proyecto, ya que ciertas entidades como las comunicaciones, los productos y los recursos quedan definidos de una manera demasiado somera.

Sin embargo, es destacable la aportación que realiza SPEM 2.0, donde se definen gran parte de los procesos necesarios para el desarrollo software. También se han explorado los enfoques de modelado y los lenguajes que proporcionan ese modelado de la representación de la información.

Para ello, tomando como base estas representaciones y ontologías, se ha definido el conjunto mínimo de entidades necesarias para poder definir el desarrollo y gestión de un proyecto software, basándose para ello en las principales áreas de conocimiento descritas en el PMBOK [Ins08].

Capítulo 6

PMO: Un sistema basado en el conocimiento para la gestión de proyectos

6.1. Introducción

El capítulo actual utiliza los resultados obtenidos del capítulo anterior y define un modelo ontológico que permite la representación de las entidades que afectan a la gestión del proyecto. La necesidad de definir modelos que puedan capturar los procesos de la gestión de proyectos supone un campo de investigación en el que se observa un creciente interés debido al crecimiento de las tecnologías semánticas.

En este capítulo se presenta *Project Management Ontology* (PMO), una ontología de dominio centrada en establecer un modelo formal de los procesos, actividades, herramientas y técnicas específicas de la gestión de proyectos. PMO proporciona una descripción completa de los términos fundamentales y características inherentes al manejo de la información asociada a la gestión, seguimiento, control y dirección de los proyectos, así como de los procesos, relaciones, restricciones y aserciones sobre los datos de proyectos. En el Anexo A y Anexo B se pueden encontrar las documentación asociada de las ontologías descritas en el lenguaje OWL. La

documentación se ha generado utilizando la herramienta *SpecGen* [SBF11] para mostrar las clases y propiedades descritas usando el lenguaje semántico OWL.

6.2. Herramientas de representación de conocimiento

Aunque las herramientas de las que disponen los gestores de proyectos son acordes con las necesidades de proyectos, en muchas ocasiones éstas no proporcionan un dimensionamiento adecuado para la gestión de proyectos software, ya que no se obtiene un modelado acorde al conjunto de variables que se deben tener en cuenta en este tipo de proyectos.

Factores como la productividad, la detección de defectos de software, el modelado de requisitos o las métricas para la trazabilidad del sistema, siempre desde el punto de vista del gestor del proyecto, no son tomados en cuenta, y por lo tanto, los proyectos no se terminan en tiempo, presupuesto o funcionalidad requeridos.

En la actualidad existe un creciente interés sobre la adecuada gestión del conocimiento en las distintas áreas de conocimiento. De hecho, en el mundo del desarrollo software, es fundamental gestionar adecuadamente dicho conocimiento, tanto el que se da de manera explícita como el tácito o implícito. El primero se define como "una herramienta de gestión para aprovechar la manipulación del conocimiento de la organización" [SSP99]. Este conocimiento táctico se encuentra físicamente en unos sistemas que incluyen tecnologías como intranets, herramientas grupales, listas de distribución, repositorios de conocimiento, bases de datos, etc. El segundo, más difícil de capturar, se basa más bien en la experiencia, guiado por el contexto, y por lo general, reside en los individuos.

En los proyectos software, dicho conocimiento es fundamental por muchas razones: conocimiento histórico, lecciones aprendidas, explotación de datos, toma de decisiones, seguimiento del proyecto, metodologías utilizadas, estimación y planificación, asignación de recursos, etc. Por ello, es necesario capturar y gestionar el conocimiento disponible en un formato y representación adecuados. Esto puede realizarse para la mayoría de áreas de conocimiento, donde pueden definirse por

expertos un conjunto de conceptos, aserciones, reglas e inferencias sobre dicha información. Ésta información puede guardarse utilizando alguna de los lenguajes de representación de conocimiento existentes (ver sección 5.3 y 5.4).

La representación del conocimiento tiene varias funciones: sirve como modelo de la realidad, establece hechos sobre el mundo real, determina un conjunto de aserciones para inferir y razonar sobre dicho modelo, proporciona una organización adecuada del conocimiento, y facilita un lenguaje que expresar el conocimiento humano [GDD09].

Una manera razonable para determinar dicho conocimiento es la utilización de ontologías, que proporcionan *"una representación explícita de un conocimiento compartido de los conceptos importantes en un cierto dominio de interés"* [Kal01], esto es, una representación declarativa de conceptos, estructuras de datos, relaciones, aserciones, reglas y restricciones que representar un modelo abstracto y simplificado de la realidad. Estas ontologías tienen dos características importantes: proporcionan una representación explícita de un modelo conceptual sobre un dominio determinado; y dicho modelo establece una representación compartida y consensuada sobre el conocimiento disponible en dicho dominio.

En la actualidad, existen muchas ontologías de dominio y sistemas basados en el conocimiento que abarcan un amplio conjunto de áreas de conocimiento en diversos dominios. Se pueden encontrar información sobre dichas ontologías en varios repositorios de ontologías [oP10, Cla11a, Pag04, Cla11b]. Sin embargo, el desarrollo de representaciones formales y explícitas para la gestión de proyectos aún está en la fase de definición y propuesta.

6.3. Project Management Ontology (PMO)

La gestión de proyectos se define como el conjunto de herramientas, técnicas, conocimiento y habilidades aplicadas a un proyecto para cumplir un conjunto de requisitos, estándares, especificaciones y objetivos que llevan a completar dicho proyecto. La gestión por proyectos se llevan a cabo en varios ámbitos, incluyendo entre ellos a su aplicación en arquitectura e ingeniería, industria química, desarrollo

software o en el ámbito de la investigación. En todos estos ámbitos, es de vital importancia la gestión de proyectos, ya que proporciona un conjunto guiado de procesos que permiten realizar un control y evaluación de las tareas a desarrollar para conseguir el producto o servicio final.

Para gestionar los proyectos, se dispone de varias herramientas centradas en capturar un parte de la información del proyecto. Sin embargo, estas herramientas se utilizan de una manera aislada –válida únicamente para las personas que utilizan dichas herramientas–, o como un conjunto integrado de aplicaciones. Por lo tanto, es complicado compartir este conocimiento con otras organizaciones o incluso, dentro de una misma organización que utilice distintas aplicaciones para la gestión de proyectos, incluso si existe la posibilidad de importar/exportar la información. Por ello, es necesario definir una representación del proyecto que pueda ser utilizada tanto por las distintas aplicaciones y que permita la interoperabilidad entre éstas. En este sentido, creemos necesario establecer una representación del conocimiento que permita capturar y gestionar adecuadamente la información del proyecto. Para este propósito, lo más adecuado es la utilización de ontologías de dominio.

Para desarrollar esta ontología, es necesario plantearse las distintas opciones para capturar dicho conocimiento: (i) hablar con expertos en la gestión de proyectos, (ii) capturar el conocimiento directamente de las aplicaciones de gestión de proyectos, o bien, (iii) obtener de alguna fuente de información un modelo descriptivo y consensuado del conocimiento. La primera opción es difícil de lograr, ya que parte de dicho conocimiento es tácito y por lo tanto, no hay forma de modelarlo adecuadamente. En la segunda opción, el modelo de desarrollo seguido por los desarrolladores está centrado en la aplicación, por lo que el modelo de datos únicamente tiene los parámetros suficientes para que la aplicación funcione, sin que esté definido el conocimiento para la gestión de proyectos de una manera completa. Finalmente, para la tercera opción, existe una gran cantidad de literatura, tanto en forma de libros como en forma de artículos de investigación. Entre todos ellos, se ha seleccionado *Project Management Body of Knowledge* (PMBOK) [Ins08], ya que proporciona una documentación exhaustiva en el área de la gestión de proyectos que ha sido desarrollado por expertos en el área y en áreas adyacentes. Y por lo tanto supone una definición experta de procesos consensuada.

Para capturar y gestionar dicho conocimiento, se ha desarrollado *Project Management Ontology* (PMO), un conjunto de ontologías que definen el conjunto de los principales procesos, conceptos y relaciones de la gestión de proyectos. Este sistema basado en el conocimiento proporciona la primera representación formal de conocimiento en el dominio de la gestión de proyectos. Para su desarrollo, se ha considerado fundamental crearlo de manera modular y estructurada, de tal manera que otras ontologías o sistemas basados en el conocimiento puedan unirse a PMO para crear una ontología específica de la gestión de proyectos en el dominio aplicado. PMO está compuesta de:

- Un taxonomía que define la estructura de un proyecto. Esta taxonomía proporciona una jerarquía base común a todos los proyectos en su estructuración, así como en la definición de los principales términos. Esta taxonomía está formada principalmente por relaciones del tipo *isA* (subclases) o *has* (composición), que define esencialmente las partes en que se puede dividir y estructurar un proyecto y sus componentes.

- Un completo vocabulario que define conceptos específicos de la gestión de proyectos, y que pueden ser aplicables a la mayoría de áreas de conocimiento. Este vocabulario ha sido obtenido del glosario de términos del PMBOK [Ins08].

- Un sistema basado en el conocimiento (KBS) que contiene el conocimiento experto de cómo gestionar un proyecto de manera adecuada. Esto incluye la estructura, el contenido semántico, y los procesos e instancias necesarias que proporcionan una guía para la dirección del proyecto.

- Un conjunto de *slots* o propiedades asociadas a uno o varios de los conceptos presentes en la taxonomía o el vocabulario.

- El conjunto de relaciones entre conceptos definidos tanto a nivel de taxonomía, como entre los distintos componentes de PMO (utilizando las propiedades *owl:equivalentClass* y *owl:sameAs*). Estas relaciones están basadas en la propia definición de los conceptos.

Se puede observar la estructura básica de PMO en la figura 6.1. Cada uno de los componentes mostrados en esta figura representa una ontología. En PMO actualmente se han definido cinco ontologías: PM–Cost, PM–Process, PM–Planning, PM–Organization, y la ontología núcleo PM–Core.

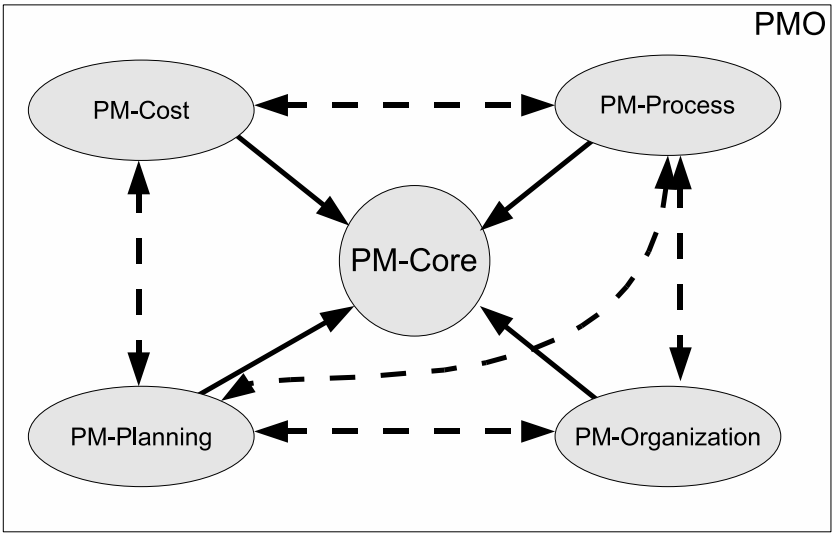


FIGURA 6.1: Componentes de la Arquitectura de la Ontología PMO.

En esta figura, se pueden diferenciar dos tipos de relaciones: las indicadas con línea continua, y las indicadas con punteado. Las flechas continuas indican una correspondencia directa entre clases que representan un mismo concepto utilizando la propiedad de OWL *owl:sameAs*, como por ejemplo los conceptos que representan *Actividad* y *Paquete de Trabajo*. Las flechas punteadas representan conceptos relacionados entre sí a través de los conceptos definidos en la ontología PM–Core.

Para el proceso de desarrollo, se ha seguido la metodología recomendada por Noy y McGuiness [NM01]. Este proceso de desarrollo consiste en un conjunto de pasos propuestos por los autores para la creación de ontologías de dominio: (i) determinación del dominio y ámbito de la ontología; (ii) consideración de reutilización de ontologías ya existentes; (iii) enumeración de los conceptos y términos clave; (iv) definición de la estructura de clases y atributos de cada una de las clases; (v) definición de las restricciones sobre los atributos; y (vi) completar la ontología, poblándola con instancias o individuos.

Para crear una vista gestionable del dominio de gestión de proyecto, se ha dividido PMO en varios componentes u ontologías, cada uno de ellos proporcionando

una visión parcial de una parte del conocimiento en la gestión de proyectos. Esta estructura se ha definido en base a las distintas partes diferenciadas en las que se compone el area de conocimiento (coste, planificación, riesgos, requisitos, aseguración de la calidad, etc.). PMO tiene los siguientes componentes:

- ***PM–Core***. La ontología núcleo está formada por el conjunto de conceptos, relaciones, axiomas y atributos que forman la base para la gestión de proyectos. En esta ontología se incluyen conceptos como proyecto, fase, entregables, productos, o actividades.
- ***PM–Process***. Esta ontología representa el conjunto de procesos y grupos de procesos de recomendada aplicación para la gestión de un proyecto. Esta ontología proporciona principalmente el conocimiento recogido en el PMBOK [Ins08].
- ***PM–Organization***. Esta ontología proporciona la estructura organizativa del proyecto, definiendo los conceptos de equipo, persona, atribuciones, habilidades, asignaciones, etc. El enfoque tomado para el desarrollo de esta ontología ha sido la división desde el punto de vista de la gestión para la organización que desarrolla el proyecto, el desarrollo de los equipos, y los actores que forman parte de dicho proyecto.
- ***PM–Cost***. Esta ontología incluye todos aquellos conceptos, atributos y relaciones relacionados con la gestión de costes, tanto monetarios como de esfuerzo. También incluye conceptos relacionados con las estimaciones y presupuestos.
- ***PM–Planning***. Finalmente, esta ontología desarrolla toda la parte de gestión de la planificación, calendario y seguimiento de un proyecto.

Aún están en proceso de desarrollo ontologías adicionales que completan las partes de conocimiento no incluidas en esta primera versión de PMO, ya que se consideraron en un primer momento como secundarias en la definición del domino. Entre estas partes del conocimiento no desarrolladas están los riesgos, los requisitos, la aseguración de la calidad y la gestión de las comunicaciones. Una vez desarrolladas, este conjunto secundario de ontologías pasarán a formar parte de PMO,

utilizando la unión de ontologías (*ontology merging*), con el objetivo de completar el dominio de conocimiento.

6.3.1. Ontología Núcleo: *PM-Core*

PM-Core es la ontología núcleo de PMO. Todas las demás ontologías en PMO están directamente relacionadas con *PM-Core*, como se puede observar en la Figura 6.1. *PM-Core* contiene el vocabulario y la estructura básica para la definición de un proyecto genérico, en cualquier ámbito de aplicación. El principal concepto definido en la ontología es el *Proyecto*, que tiene asociado una serie de atributos inherentes, como su nombre, descripción, procesos a utilizar, criterios de calidad o hitos. Así mismo, un proyecto puede ser parte de un *Programa* o un *Portafolio*. En el proceso de división hacia artefactos más manejables, se pueden definir distintos tipos de *Componentes de proyecto* (clase abstracta): *Fases* (o subfases), *Entregables*, o *Paquetes de trabajo*, siguiendo las recomendaciones de estructuración del trabajo del proyecto [Ins06]. Cada uno de estos componenetes pueden combinarse entre sí mediante las relaciones definidas en la propia ontología, para formar la *Estructura de Descomposición del Trabajo* (WBS).

En la figura 6.2, se puede observar la jerarquía de clases definida en *PM-Core*. Tal y como se define en [Ins06], "*una Estructura de Descomposición del Trabajo es una descomposición jerárquica orientada a ser entregada del trabajo a ser ejecutado por el equipo del proyecto para cumplir los objetivos del proyecto y crear los entregables solicitados*". Los elementos terminales son principalmente entregables o paquetes de trabajo, que se dividen a su vez en actividades. Las actividades representan los elementos gestionables más pequeños que pueden asignarse directamente a un equipo del proyecto o una persona.

6.3.2. Ontología de Procesos de Gestión de Proyectos: *PM-Process*

PM–Process es la ontología que contiene el modelo de procesos de gestión del PMBOK [Ins08]. En esta ontología se incluyen aquellos grupos de procesos en los

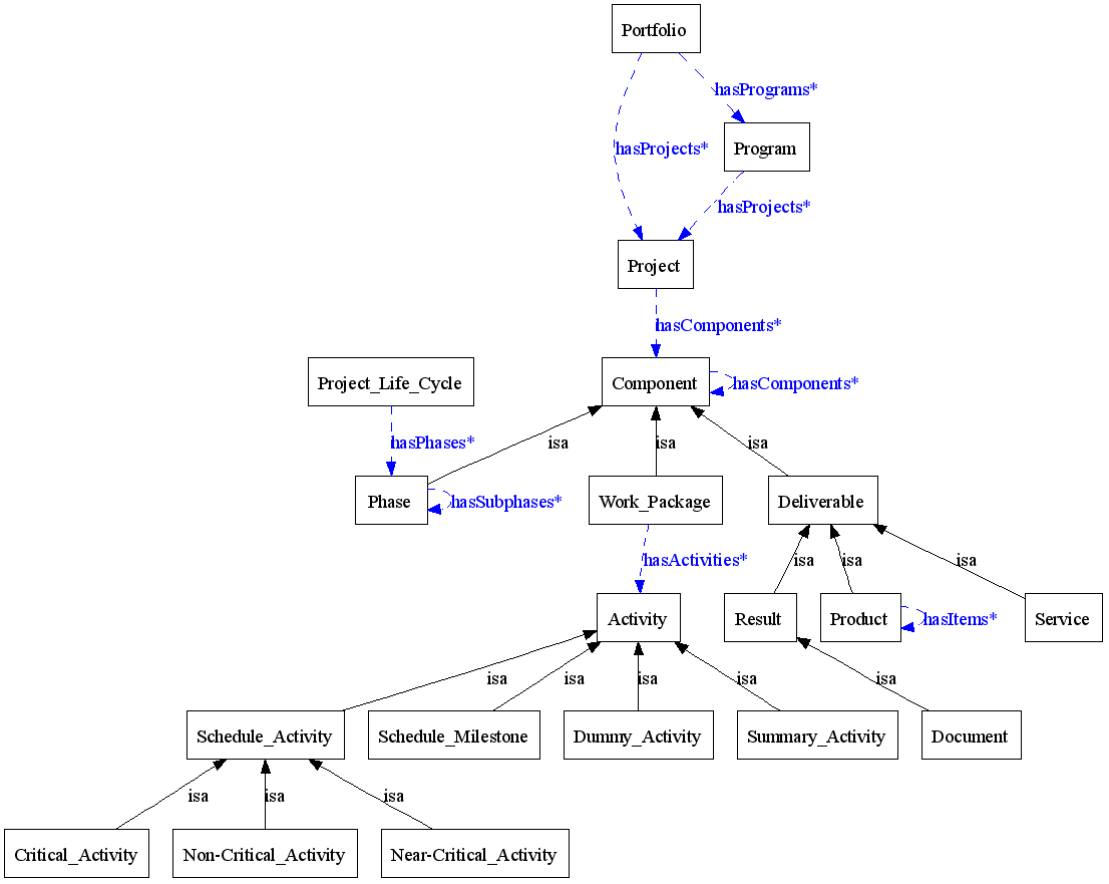


FIGURA 6.2: Vista simplificada de la jerarquía de clases de *PM-Core*

que está dividido el cuerpo del conocimiento en la gestión del proyecto: procesos de iniciación, planificación, ejecución, control y monitorización, y finalización.

Estos grupos de procesos son necesarios para cualquier proyecto, ya que proporciona un proceso guiado, siendo necesario que se ejecuten siguiendo la misma secuencia para todos los proyectos, independientemente del ciclo de vida utilizado o el proceso de desarrollo software seleccionado. Cada uno de estos grupos de procesos contiene una serie de procesos de gestión interrelacionados entre sí, tal y como se define en el PMBOK. Esto incluye una descripción completa del proceso de gestión, así como las entradas, salidas, herramientas y técnicas necesarias para llevar a cabo dicho proceso. El equipo de gestión del proyecto será el encargado de seleccionar el subconjunto adecuado de estos procesos para llevar a cabo el conjunto de actividades del proyecto.

Sin embargo, *PM-Process* no está restringido a la definición de una serie de clases,

relaciones y propiedades para almacenar el conocimiento sobre los procesos de gestión, sino que también proporciona un sistema basado en el conocimiento, ya que la ontología se ha completado con un conjunto de instancias que definen los procesos con la información obtenida del PMBOK [Ins08].

Este conocimiento tiene el objetivo de proporcionar una guía a la hora de definir el conjunto de actividades presentes en *PM-Core*, pero también de adecuar la gestión de estas actividades al proyecto específico. Esta ontología representa únicamente los procesos de gestión, por lo que no incluye otros procesos que no son específicamente procesos de gestión, por ejemplo, procesos software, que podrían estar definidos en otra ontología.

6.3.3. *PM-Organization*

En un sentido estricto, la organización y los recursos humanos no es una parte constituyente de la gestión de proyectos, aunque sí que es una pieza fundamental para el éxito del proyecto. De hecho, algunos procesos de gestión están directa o indirectamente relacionados a la organización (por ejemplo, todos aquellos procesos de la Gestión de Recursos Humanos del Proyecto presentes en el PMBOK).

Durante el desarrollo de esta ontología se ha utilizado parte el trabajo ya desarrollado en las ontologías de organización disponibles en [LSH+99], que describen las principales características de una organización, incluyendo metas, jerarquía, roles, puestos desempeñados, personas, equipos, etc. La utilización del conocimiento disponible en estas ontologías ha sido de gran utilidad para crear y adaptarlas en *PM-Organization* a un enfoque de gestión de proyectos.

En *PM-Organization* se han definido los conceptos básicos de una organización, como organización, persona, empleado, puesto o habilidades, y se ha extendido añadiendo conceptos más relacionados a la gestión del proyectos, como equipo de proyecto, miembro del equipo, gestor/director o equipo virtual. Finalmente se han redefinido o agregado nuevos términos conducentes a su adaptación a las actividades de gestión, como por ejemplo, habilidades de gestión, responsabilidades de gestión, competencias o comunicaciones.

Esta ontología es una base del conocimiento que puede unirse a *PM-Core* para proporcionar una definición completa del proyecto, tanto desde el punto de vista structural como desde el organizativo. La estructura de la taxonomía asociada a *PM-Organization* puede observarse en la figura 6.3.

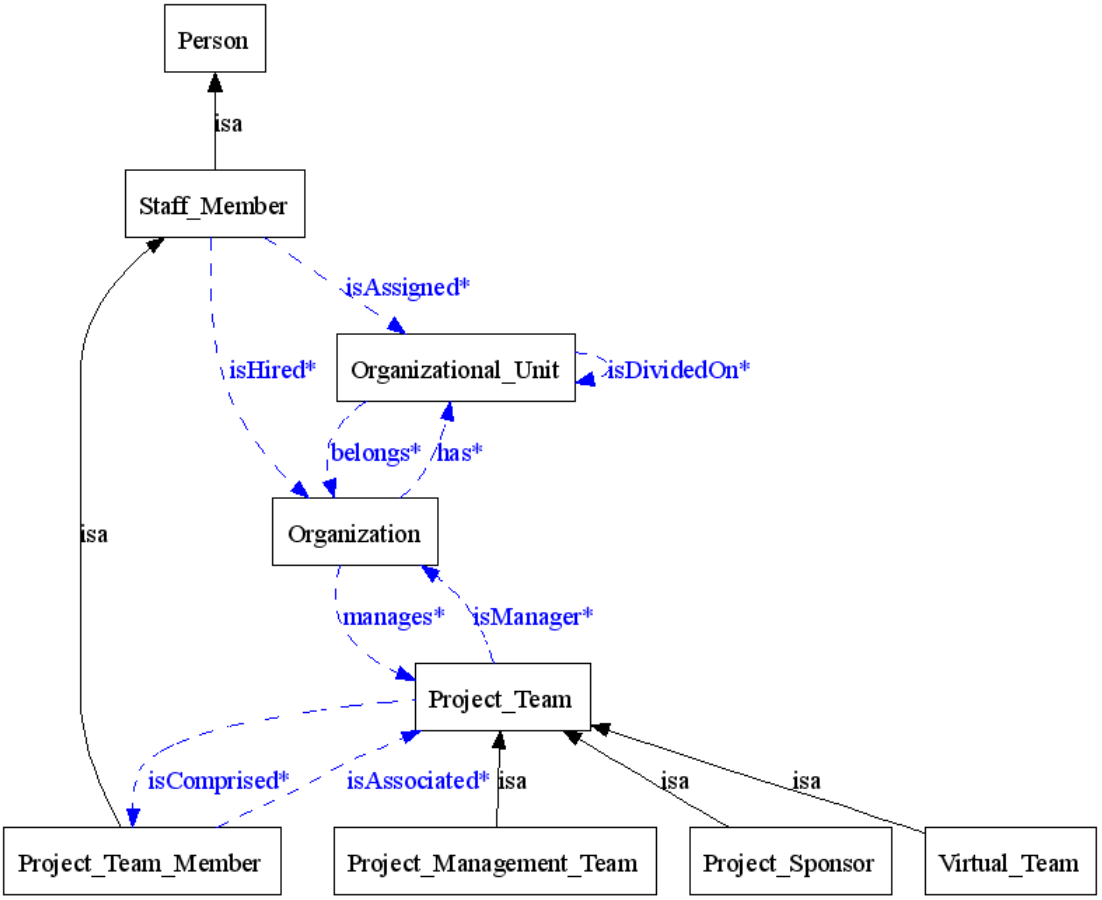


FIGURA 6.3: Estructura de la jerarquía de clases de *PM-Organization*

6.3.4. *PM-Planning*

La ontología *PM-Planning* contiene el conocimiento necesario para la representación de la planificación, calendario, estimaciones, asignación, seguimiento, control, gestión de cambios y finalización de las actividades del proyecto.

La unidad utilizada en esta ontología es principalmente el tiempo, aunque también están consideradas otras unidades de medida, como los recursos y el esfuerzo. Debido a que ya existen representaciones del tiempo en forma de ontologías, se

ha considerado conveniente la utilización de *OWL Time Ontology* [HP06] para ayudar al desarrollo de la ontología *PM-Planning*. De hecho, todas las medidas de duración e intervalos de tiempo han sido definidas en función de esta ontología. En *PM-Planning* se pueden diferenciar dos tipos de medidas:

- Las propiedades que están directamente relacionadas con la planificación. Entre estas propiedades se incluyen aquellos valores necesarios para las estimaciones y seguimiento del proyecto, como por ejemplo, *Fecha de Inicio Temprana*, *Fecha de Inicio Tardía*, *Fecha de Finalización Temprana* y *Fecha de Finalización Tardía*, utilizados en el Método del Camino Crítico [Ker09], o simplemente otras propiedades directamente asociadas al cumplimiento de plazos de las actividades (por ejemplo, *Fecha Actual*, *Fecha de Finalización Actual*, *Fecha de Finalización Estimada*, etc.).
- Otros conceptos que definen estructuras de planificación y estimación más complejas como las Líneas de Base.

En *PM-Planning* se ha decidido incluir la definición temporal para distintos husos horarios y calendarios, ya que los proyectos, especialmente aquellos que trabajan con equipos virtuales, son desarrollados generalmente en distintas localizaciones. *PM-Planning* está muy relacionada con *PM-Cost* a través de *PM-Core*, ya que ambas comparten varios de los términos de alto nivel, como por ejemplo, la *Línea de Base* o la *Secuencia del Camino Crítico*, entre otros.

6.3.5. *PM-Cost*

El componente *PM-Cost* representa el conjunto de conceptos relativos al presupuesto, planificación, estimación y control de los costes. Se ha diferenciado *PM-Cost* de *PM-Planning* ya que ambos componentes representan diferentes términos: el coste del proyecto determina generalmente un valor monetario del proyecto, mientras que el calendario engloba los aspectos temporales del proyecto, comúnmente más utilizados para medir la progresión del proyecto. De hecho, las

organizaciones generalmente tienen dos aplicaciones separadas para las estimaciones de coste/presupuesto y el calendario. Sin embargo, ambas ontologías determinan una información fundamental en la gestión de proyectos. De hecho, determinar buenas estimaciones del presupuesto y del calendario son factores fundamentales para el éxito del proyecto [WF02].

La ontología *PM-Cost* está principalmente expresada en términos de coste monetario dada una cierta moneda (por ejemplo, si el proyecto se está realizando conjuntamente entre Europa y EE.UU., los equipos del proyecto trabajarán tanto en Euros como en Dólares). Debido a que el proyecto generalmente está supeditado a un presupuesto definido, las unidades a utilizar para expresar el coste no deberían cambiar durante dicho proyecto, o al menos no deberían estar sujetas a los cambios monetarios que se produzcan. Este hecho está capturado en la ontología, donde se establecen distintas unidades monetarias, y un tipo de datos con funciones embebidas para hacer transparente el coste del proyecto.

6.4. Integración de PMO

En la actualidad, una de las principales características de las ontologías es facilitar la interoperabilidad con otra información, ontologías o aplicaciones. Durante el proceso de desarrollo de PMO, se ha decidido desarrollar una definición genérica del área de conocimiento de la gestión de proyectos, de tal manera que pueda hacerse extensible a otras áreas de conocimiento diferentes. De hecho, se puede afirmar que esta característica es fundamental para la integración del conocimiento de la gestión de proyectos con otras áreas de conocimiento, como la ingeniería del software o la arquitectura. Así, PMO ha sido desarrollada utilizando un enfoque modular, para poder facilitar la integración con otras formas de conocimiento (ontologías, bases de datos, sistemas de gestión del conocimiento, aplicaciones, etc.).

Por una parte, todos los componentes de PMO están altamente integrados. Este hecho puede observarse en la Figura 6.1, donde las dependencias entre todos los componentes de PMO están relacionados de una u otra manera. Por ejemplo, la ontología *PM-Process* puede ser fácilmente definida independientemente de las demás

ontologías, pero los conceptos y propiedades presentes en ella están fuertemente relacionados en la manera de distribuir el trabajo al equipo del proyecto (presente en *PM-Organization*). De la misma manera, se pueden encontrar relaciones similares en las demás ontologías de PMO. Actualmente, se está desarrollando nuevos componentes para capturar mayor información en áreas asociadas a la gestión de proyectos, como los riesgos, las comunicaciones o la calidad del proyecto.

Por otra parte, es recomendable integrar este sistema basado en el conocimiento con otras ontologías existentes, para extender y enriquecer el ámbito de actuación de PMO. Esto se puede realizar mediante procesos de unión y mapeado de ontologías [ES04]. Supongamos que se desea unir PMO con una ontología que modela el conocimiento en Ingeniería del Software. Se puede realizar un mapeado de ambas ontologías siempre que los conceptos estén descritos de similar manera o se puedan encontrar conceptos equivalentes en ambas ontologías. Por ejemplo, en *PM-Process* se ha definido el concepto de *Proceso*, que tiene una subclase denominada *Procesos de Gestión*, que especifica los procesos que se recomiendan para una gestión eficiente de los proyectos. En otra ontología, asociada al conocimiento en la Ingeniería del Software, es altamente probable que nos encontremos con una serie de conceptos que definan los procesos software. Éstos procesos software dependerán del ciclo de vida seleccionado, pero también de las políticas aplicadas en la organización, la experiencia del gestor del proyecto, etc. Se puede afirmar que si dicha ontología o representación de conocimiento existe, podría definirse una mapeado entre ambas ontologías para la unión del conocimiento en la gestión de proyectos al desarrollo del proyecto software.

En la Figura 6.4, dadas dos ontologías, PMO y otra que represente el conocimiento sobre los procesos software, ambas pueden ser unidas utilizando un mapeado del conocimiento. De hecho, en la actualidad existen varios trabajos abiertos sobre el mapeado de ontologías [KS03, CSH06], que incluyen entornos de trabajo para el mapeado, métodos, herramientas, traductores, mediadores y técnicas.

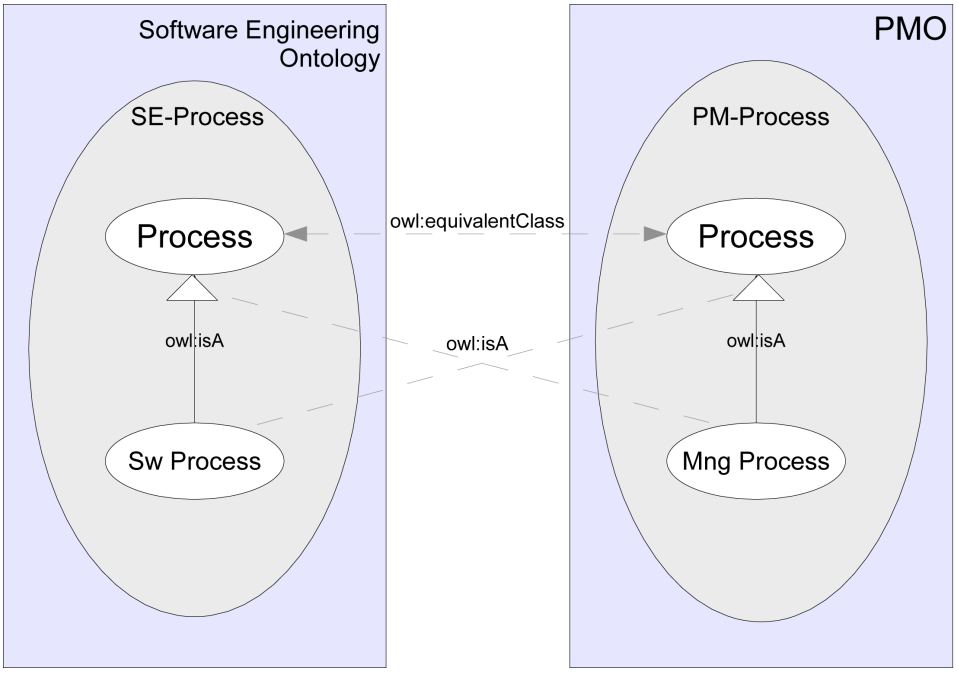


FIGURA 6.4: Un ejemplo de unión de dos ontologías de áreas de conocimiento diferentes

6.5. Conclusiones

En este capítulo se ha presentado *Project Management Ontology*, un sistema basado en el conocimiento para la captura y gestión del conocimiento en el área de la gestión de proyectos. Esta representación del conocimiento proporciona una visión semántica de la información del proyecto utilizando el enfoque de gestión.

Este trabajo propone un conjunto de ontologías básicas que son necesarias para comenzar a capturar y salvar la información de gestión de los proyectos en un formato independiente de las aplicaciones utilizadas, facilitando así mismo el proceso de interoperabilidad e intercambio de información. Aunque el proceso de desarrollo de ontologías es duro y bastante largo, la necesidad de proporcionar y obtener el conocimiento disponible sobre la gestión de proyectos es fundamental para gestionar de una manera cada vez más eficiente. De hecho, en la actualidad, existe una gran cantidad de grupos de investigación dedicados a la gestión del conocimiento.

Así mismo, en este capítulo se han mostrado las principales características de *Project Management Ontology* (PMO). PMO está compuesta de varias ontologías,

donde cada una desarrolla una parte del conocimiento de la gestión de proyectos. Esta estructuración por partes de conocimiento, ha permitido trabajar individualmente con cada una de ellas (dividiendo el problema de la representación del conocimiento en partes más pequeñas), lo que ha facilitado su posterior proceso de integración. Así mismo, en la actualidad se está tratando de unir esta ontología con ontologías de otras áreas de conocimiento.

Capítulo 7

Razonamiento sobre una ontología de gestión de proyectos

7.1. Introducción

En el capítulo anterior se ha presentado una ontología que define los principales artefactos para la representación de la gestión de proyectos. Esta ontología (o mejor dicho, conjunto de ontologías) proporciona esa base de conocimiento experto suficiente para instanciar un proceso de desarrollo software desde el punto de vista de la gestión.

Uno de los problemas de la gestión de proyectos, y más concretamente de los proyectos software, es la constante necesidad de información a distintos niveles de gestión. En este capítulo se estudian los lenguajes que permiten realizar razonamiento sobre la información contenida en la ontología, para poder proporcionársela a los distintos niveles de gestión. Así mismo, es interesante no sólo la inferencia, sino la creación de nuevo conocimiento a partir del existente, usando un sistema de reglas que permitarn realizar dicho razonamiento. Esto permitirá la validación de la ontología PMO, presentada en el capítulo anterior.

7.2. Inferencia sobre ontologías

El uso de las ontologías para la creación de sistemas basados en el conocimiento está siendo generalizado en la gran mayoría de áreas de conocimiento. Dentro de ese creciente interés, es necesario destacar los trabajos realizados en la definición semántica para la gestión conceptual de proyectos [RdA06] [AAH+06] [TYD+09]. En capítulos anteriores ya se han estudiado diversas propuestas formales para proporcionar dicha conceptualización. En línea con *Project Management Ontology* [RBD08], detallada en el capítulo anterior, se están proponiendo nuevas ontologías que explotan la capacidad semántica de su conceptualización en el área de la gestión de proyectos [SA09] [SCA09].

La ontología PMO, descrita en el capítulo anterior (en el Anexo A se puede encontrar la documentación asociada a la ontología núcleo y en el Anexo B, la información derivada de la ontología *PM-Organization*), se va a utilizar para aplicar reglas que demuestren su validez para realizar razonamiento. Esta ontología está expresada utilizando el lenguaje OWL [BvHH+04], lenguaje recomendado por el W3C. Para el razonamiento, éste se puede realizar en OWL a nivel de clase, propiedad o individuo. Ejemplos de razonamiento sobre clases son la pertenencia a clases, equivalencia entre clases, consistencia, clasificación de la información, obtención de propiedades adicionales usando la transitividad y equivalencia, etc.

Para realizar inferencias sobre ontologías definidas en OWL, el W3C propone utilizar como estándar el lenguaje SWRL (*Semantic Web Rule Language*) [HPSB+04], basado en RuleML [BPT+10]. SWRL extiende la definición en OWL proporcionando reglas basadas en lógica, y por lo tanto, proporciona una mayor expresividad a la hora de realizar razonamientos sobre OWL. En SWRL las reglas están definidas en términos de antecedentes y consecuentes. La combinación de OWL y SWRL para la definición de reglas sobre una ontología se ha demostrado útil a la hora de obtener cierta información que se pueda considerar razonada sobre un modelo semántico [RGS10].

El lenguaje SWRL se propuso para complementar a OWL en el razonamiento semántico a través de reglas [DHC07]. SWRL permite realizar razonamiento sobre individuos definidos en base a un modelo definido en OWL. Según define el W3C,

SWRL extiende el conjunto de axiomas de OWL con reglas Horn, combinando los sublenguajes OWL DL, OWL Lite y RuleML. Estas reglas Horn se definen genéricamente como una implicación entre un antecedente (*body*) y su consecuente (*head*), de tal manera que si se cumplen las condiciones especificadas para el antecedente, entonces también se producen las condiciones especificadas para el consecuente. Usando esta implicación, las reglas se pueden definir a través de la siguiente sintaxis:

`antecedente` $\Rightarrow$ `consecuente`

El ejemplo clásico para el uso de reglas se basa en el parentesco, definido de la siguiente manera:

`hasParent(`$?x_1$`,`$?x_2$`)` $\wedge$ `hasBrother(`$?x_2$`,`$?x_3$`)` $\Rightarrow$ `hasUncle(`$?x_1$`,`$?x_3$`)`.

De esta manera, se hace cierta la expresión `hasUncle(`$?x_3$`)`, para cualquier individuo $?x_1$ que tenga un padre $?x_2$, y donde $?x_2$ tenga un hermano $?x_3$.

A efectos de simplicidad y para presentar un razonamiento sencillo sobre la ontología, se va a utilizar el motor de reglas Jess sobre el entorno de desarrollo de ontologías Protégé *eriksson.jess*. Otra alternativa posible para realizar el razonamiento hubiera sido utilizar Pellet, aunque no proporciona un valor añadido a la demostración que se pretende realizar.

7.3. Desarrollo de ontologías para ingeniería del software

El desarrollo de ontologías para las subáreas de conocimiento definidas en ingeniería del software requiere llegar a un consenso en la definición de un vocabulario común compartido para investigadores, profesionales y desarrolladores. Una ontología que pretenda capturar el conocimiento dentro de los proyectos software debería ser capaz de definir unívocamente los procesos software, actividades, planificación, presupuesto, riesgos, requisitos, pruebas, aseguramiento de la calidad, gestión del proyecto, organización, herramientas, métodos y metodologías.

La importancia de haber desarrollado una ontología del dominio de la gestión de proyectos en el capítulo anterior es que se proporciona un modelo semántico que puede ser utilizado como representación para expresar la información del proyecto.

A pesar del avance en las tecnologías semánticas, en proyectos software la utilización de estas tecnologías se ha aplicado de un modo limitado, por lo que aún hay pocos trabajos de investigación que hayan propuesto de manera consensuada y compartida el desarrollo de ontologías específicas para la ingeniería del sofware. Las iniciativas de desarollo de ontologías están principalmente centradas en ciertos experimentos limitados a un cierto dominio dentro del área, como por ejemplo, en el aseguramiento de la calidad [WADD03] o en las métricas software [GBC+06]. Parte de esta tesis se ha centrado en enfocar este problema para definir explícitamente los procesos de gestión de proyectos que afectan a la ingeniería del software [RBDA07].

7.4. Ontología de Proyecto

La ontología de proyecto tiene una serie de conceptos base que están mapeados en la ontología *PM-Core*, mientras que el resto de conceptos definidos para la gestión del proyecto están mapeados a través de extensiones, como por ejemplo la ontología relativa a organización (*PM-Organization*). Debido a que no existe una manera única de representar conceptos desde el punto de vista del gestor de proyecto, pueden existir diferentes formas de representar la misma información, tal y como se ha podido comprobar durante el desarrollo de esta tesis.

Para realizar inferencia sobre los datos se va a utilizar la ontología núcleo *PM-Core*. Esta ontología núcleo proporciona el conjunto de conceptos, relaciones, axiomas y atributos que forman la base para la definición de una estructuración de proyecto. Se incluyen conceptos como proyecto, fase, entregables, productos, o actividades. Por otra parte, para la inferencia se va a utilizar la ontología *PM-Organization*, que define la estructura organizativa del proyecto, definiendo los conceptos de equipo, persona, atribuciones, habilidades, asignaciones, etc. La descripción de ambas ontologías se ha definido en el capítulo anterior. Así mismo, en los anexos

A y B se puede visualizar la documentación obtenida a partir de las clases de la ontología núcleo y la de organización, respectivamente.

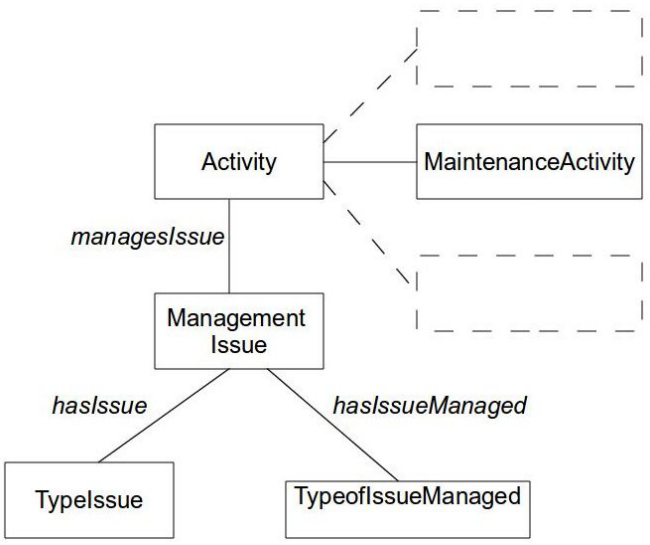


FIGURA 7.1: Extensión de la ontología núcleo para agregar conceptos de gestión

El desarrollo de estas ontologías se han construído con la idea de proporcionar una conceptualización específica como objetivo. Para claridad de comprensión, se han implementado sobre estas ontologías una serie de reglas en SWRL que permiten no sólo la consulta sobre los datos, sino la creación de nuevo conocimiento utilizando SWRL. Para ello, ha sido necesario extender la ontología núcleo (*PM–Core*) con una serie de relgas que hemos denominado *PM-Core extension* (ver Figura 7.1).

7.5. Extensión sobre ontologías con reglas

La extensión de la ontología núcleo se basa en diferentes tipos de aplicaciones que puede cubrir el concepto *Activity*. En este caso, se está modelando una serie de problemas con los que trabajar en el contexto del proyecto.

La ontología descrita en la figura 7.1 puede ser interpretada, de una manera más concisa de la siguiente manera: en cada proyecto pueden surgir diferentes problemas o interrupciones, que se modelan como `Management Issues`. Cada uno de los problemas que puedan surgir es etiquetado como `Management Issue`, que

puede ser gestionado por una `Activity`. En el diagrama anterior, esto se modela a través de la relación `Activity` ***`Manages Issue`*** `Management Issue`. Cada uno de los `Management Issue` se puede clasificar dependiendo del tipo (usando `Type Issue`), y así mismo, hay un subconjunto de problemas que pueden ser gestionados eficazmente a través de la entidad `Type of Issue Managed`. El concepto `Management Issue` está relacionado con `Type Issue` a través de la relación ***`hasIssue`***, que a su vez también está relacionado con la entidad `Type of Issue Managed` a través de la relación ***`hasIssueManaged`***. En este párrafo se han definido conceptualmente una extensión sobre la ontología desarrollada que puede ayudar a resolver problemas de gestión, a través de la definición de nuevas tareas que entrarían dentro del proceso de desarrollo.

Para este capítulo se propone un conjunto de reglas para inferir conocimiento en base a los individuos definidos en función del modelo la ontología. Usando las definiciones de conceptos existentes en PMO, se han seleccionado aquellos que son susceptibles de poder realizar un razonamiento. Esta parte es la generación de informes de problemas, cambios o gestiones de eventos. Se han de definir una serie de reglas SWRL básicas en la extensión a la ontología núcleo, que permita la representación de las restricciones sobre las tareas de gestión y que nos permita incluir nuevo conocimiento a la ontología. Otro conjunto de reglas SWRL son aquellas que permiten restablecer un conjunto de valores específicos. Estas consultas y reglas muestras cómo es posible crear un subsistema de razonamiento sobre los conceptos ya definidos en la ontología.

A continuación se describen parte de las reglas que han sido incluídas como Extensión de *PM-Core*. Estas reglas pueden ser clasificadas como reglas de consulta y reglas de gestión del razomiento.

7.5.1. Reglas de Consulta

Una de las formas de hacer uso de SWRL es obtener conocimiento a través de consultas que se realizan sobre una ontología, de tal manera que cada expresión sirve para conocer información que se infiere con los datos ya existentes. Ejemplos de reglas de consulta son:

- Una consulta simple:
  `organization:Project_Team_Member(?p)` ⇒ `sqwrl:select(?p)`.

  Esta regla obtiene todos los miembros de un equipo para un proyecto específico.

- En esta segunda regla de consulta, el sistema obtiene todas las peticiones de cambio que han sido inicadas por un miembro específico del equipo (*JaneDoe*)

  `pmo:Approved_Change_Request(?r)` ∧
  `pmo:change_request_initiator(?r, p1:JaneDoe)` ⇒ `sqwrl:select(?r)`

7.5.2. Reglas de Razonamiento

La diferencia con las reglas de consulta es que a partir de la información se infiere de manera semiautomática nuevo conocimiento. Por ejemplo, para la ontología que se está utilizando podemos definir las siguientes reglas de razonamiento:

- Regla 1:

  `pmo:ManagementIssue(pmo:problemaSerio)` ∧
  `pmo:MaintenanceActivity(?a)` ⇒
  `pmo:hasIssue(pmo:problemaSerio, pmo:revisionGeneralProyecto)` ∧
  `pmo:hasIssueManaged(pmo:problemaSerio, pmo:reactivo)` ∧
  `pmo:managesIssue(?a, pmo:problemaSerio)`

  Esta regla realiza una serie de operaciones si existe un problema serio como problema detectado a nivel de gestión (`problemaSerio`) y además existe una actividad de mantenimiento (`MaintenanceActivity ?a`), que simplemente traslada la gestión del problema a través de `managesIssue`.

- Regla 2

  `pmo:ManagementIssue(pmo:problemaNimio)` ∧
  `pmo:hasIssueManaged(?i, pmo:reactivo)` ∧

`pmo:MaintenanceActivity(?a)` ⇒ `pmo:managesIssue(?a, ?i)`

La segunda regla consulta si hay algún problema denominado `problemaNimio` y si dicho problema también es del tipo `reactivo` (esto también puede gestionarse debido que se produce `hasIssueManaged`) y si hay una actividad de mantenimiento (`MaintenanceActivity`), entonces la actividad `?a` se encarga de la gestión del problema `?i` (`managesIssue`)

- Regla 3 `pmo:ManagementIssue(pmo:personnelProblem)` ∧
  `pmo:hasIssueManaged(?i, pmo:internal)` ∧
  `pmo:ManagementActivity(?a)` ⇒ `pmo:managesIssue(?a, ?i)`

De manera similar, la tercera regla funciona parecido a la regla anterior, pero utiliza las instancias `personnelProblem` and `internal`.

7.5.3. Entorno de ejecución

Para ejecutar las reglas de razonamiento definidas anteriormente, es necesario un contexto de ejecución. Para ello, utilizaremos la pestaña SWRL, que es una extensión a Protégé-OWL que permite la ejecución de reglas utilizando alguno de los motores de razonamiento disponibles en el *framework* [OTN+07].

En la figura 7.2 se puede observar el resultado de la ejecución de la primera regla en Protégé. Así mismo, en las figuras 7.3 y 7.4 se puede observar el resultado de la ejecución de las reglas de gestión en el entorno Jess-Protégé.

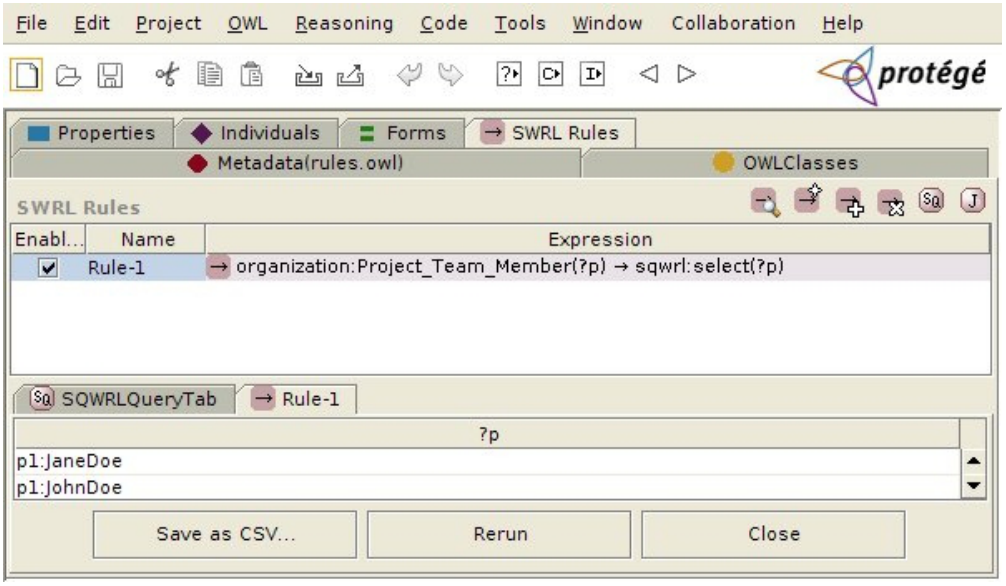


FIGURA 7.2: Entorno de ejecución de una regla simple en SWRL (*Regla 1*)

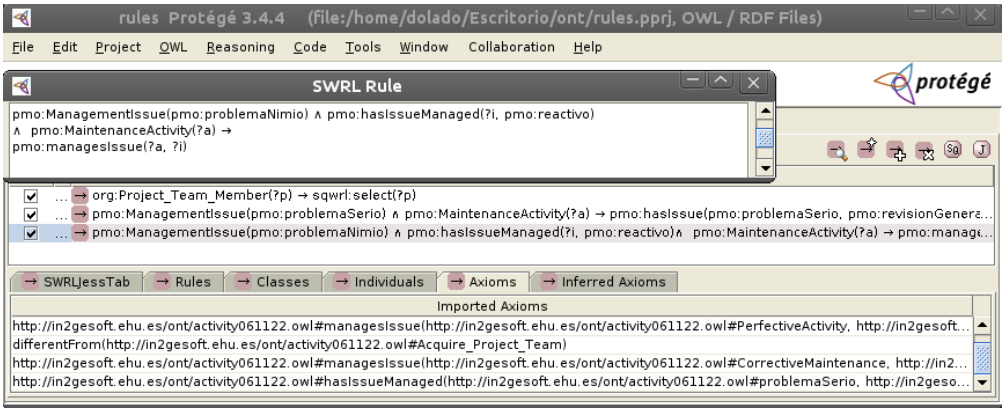


FIGURA 7.3: Entorno de ejecución de una regla compleja en SWRL (*Regla 2*)

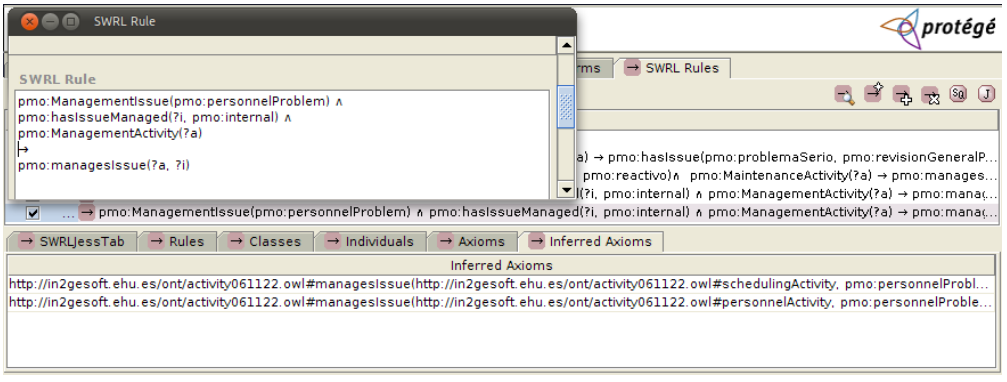


FIGURA 7.4: Entorno de ejecución de otra regla compleja en SWRL (*Regla 3*)

7.6. Conclusiones

En este capítulo se ha mostrado, a través de diferentes ejemplos, cómo se puede realizar inferencias sobre los datos utilizando los modelos de representación presentados en el capítulo anterior, y cómo extender información sobre la gestión del proyecto, utilizando dicho modelo conceptual. Uno de los principales problemas de la gestión de proyectos, y más específicamente en la gestión de proyectos software, es la constante necesidad de información a diferentes niveles de gestión. El concepto de *ontología* permite definir ideas y conceptos a diferentes niveles.

La combinación de una ontología expresada en OWL y el lenguaje de reglas SWRL puede proporcionar información adicional sobre un proyecto, mejorando de esta manera la capacidad de gestión. En este capítulo se han demostrado dos tipos de reglas que se pueden construir a partir de SWRL: una para la consulta; y otra para el razonamiento.

En el área de la gestión de proyectos es necesario la definición de un mecanismo que permita la organización y formalización de este tipo de reglas de razonamiento. Mientras el estudio conceptual de la estructura del proyecto ha sido detalladamente analizada en esta tesis, principalmente por el esfuerzo investigador realizado en el área de la gestión de proyectos en la última década, aún queda pendiente hacer *inteligente* a los sistemas basados en el conocimiento que describen a los proyectos.

7.7. Objetivos alcanzados en la Segunda Parte

En esta segunda parte de la tesis, se ha explorado los distintos lenguajes de modelado y las representaciones del conocimiento sobre el área de procesos para poder obtener ideas sobre cómo se han desarrollado otras representaciones que modelaban procesos de negocio. Utilizando esta base, se ha realizado un análisis de los procesos definidos en el PMBOK *pmi.pmbok*. A partir de ese conocimiento consensuado de los procesos definidos, se ha podido realizar un proceso de modelado, que hemos denominado *Project Management Ontology*. Esa ontología, que ha sido implementada utilizando el lenguaje de modelado OWL, recomendado por la W3C

para la representación de conocimiento, ha sido utilizada en este último capítulo para realizar inferencia sobre los datos, tanto a través de consultas como a través de ejecución de reglas de razonamiento, creando nuevo conocimiento a partir del ya existente.

Con esta segunda parte se ha alcanzado el objetivo de plantear un modelo de representación que permitiera la definición de un proyecto utilizando modelos semánticos, y poder realizar razonamiento sobre la información de proyecto. Para ello, se ha proporcionado una extensión que define una serie de reglas que permiten la generación semiautomática de nuevo conocimiento a partir del existente. Para ello se ha utilizado SWRL, un lenguaje de definición de reglas, que permite la definición de expresiones del tipo `antecedente` $\Rightarrow$ `consecuente`. Usando estas reglas en un *framework* de ejecución como Protégé–Jess, se ha podido demostrar la inferencia de nueva información a partir de la información de proyecto de la que se dispone.

Capítulo 8

Conclusiones

8.1. Introducción

La Gestión de Proyectos supone una parte importante del proceso de desarrollo software. Desafortunadamente, hay aún demasiados problemas en los proyectos actuales. Dichos problemas incluyen desvíos en presupuesto y calendario, resultados –productos o servicios finales– que no cumplen las características y/o calidad especificadas [Gro09a]. Para resolver estos problemas es necesario centrarse en los factores críticos de éxito, como por ejemplo, la definición de canales adecuados de comunicación o la realización de planificaciones y presupuestos realistas. Estos factores, sin embargo, simplemente evidencian la causa de los fallos en los proyectos, pero no definen herramientas, metodologías o métodos concretos para evitar que se produzcan estas causas. En esta disertación se han propuesto soluciones específicas a los actuales procesos de gestión, centrándose en dos aspectos inherentes del desarrollo software: (i) la representación de la información del proyecto, y (ii) la visualización de dicha información.

La Ingeniería del Software se puede considerar como una disciplina formal, donde desde hace muchos años se han definido conceptos y principios fundamentales [CGM+89]. Por desgracia, los principios teóricos, conceptos y metodologías no se aplican de una manera adecuada en el desarrollo software, y por lo tanto, aún

existen una gran cantidad de problemas, desviaciones y fallos, que hace que los proyectos fracasen [Gro09a].

Actualmente, el desarrollo software utiliza la gestión de proyectos como herramienta para el control, la dirección y el aseguramiento de los objetivos del proyecto, utilizando para ello las técnicas de seguimiento, monitorización controlo de riesgos, aseguramento de la calidad y otro conjunto de técnicas y procesos que ayuden a alcanzar los objetivos del proyecto cumpliendo los requisitos, con el presupuesto y calendario acordado con el *stakeholder*. No obstante, el área de la gestión de proyectos aún necesita ser también un área de conocimiento madura, tal y como indica Kerzner en sus *16 Puntos para la Madurez de la Gestión de Proyectos* [Ker09]

Sin embargo, son pocos los proyectos que cumplen dichos requisitos, y la mayoría de ellos no cumplen en tiempo o presupuesto. Estos problemas no son nuevos; desde el inicio de la ingeniería del software como disciplina, los problemas que van surgiendo siguen siendo similares, a pesar del avance de herramientas de desarrollo y el uso generalizado de metodologías de gestión [Bro95, Zel78].

Desde el punto de vista del doctorando, la gestión de proyectos es una de las áreas de la ingeniería del software que ha proporcionado los mayores problemas al proceso de desarrollo. De hecho, esos problemas persisten aún en la actualidad. Una de las principales causas de las desviaciones se deben a un control inadecuado de las actividades de gestión, entre las que se incluyen la definición del proyecto, la estimación, la definición y control de riesgos, la organización, los procesos de control y la evaluación, entre otros [Jur99]. Algunos autores proponen solucionar esos problemas centrándose en la definición de las tareas que están directamente relacionadas con los factores críticos de éxito [Jur99, WF02, Gro09a, Ker09].

En entornos multiproyecto, los problemas de gestión son incluso más serios, debido a la complejidad de gestionar varios proyectos a la vez, además de los problemas que se aparecen en cada uno de los proyectos individuales. Ejemplos de los problemas que aparecen en los entorno multiproyecto son la coordinación, el establecimiento de prioridades y la competición por recursos escasos [EA03][EJ03].

En esta tesis se han proporcionado soluciones específicas que se centran en dos de los problemas que adolecen en la gestión de los proyectos software. Por una parte, en la visualización de la información, ya que proporciona una visibilidad del estado del proyecto en un momento dado, basándose en una abstración del proyecto real. Por otra parte, el modelado de los proyectos no proporciona una información estructurada sobre la que se permita trabajar en una definición exahustiva de los procesos de gestión del proyecto.

8.2. Los problemas en el desarrollo software

Se considera que uno de los mayores retos de la ingeniería del software se produce cuando se desarrolla software complejo. Por software complejo se entienden sistemas críticos, simulación y aplicaciones de tiempo real, sistemas de gestión comercial avanzada, sistemas de computación y resolución de problemas o software de toma de decisiones y captura de conocimiento.

A pesar del avance en las herramientas de desarrollo y los métodos y metologías en ingeniería del software, la gestión de los proyectos software es cada vez más compleja. Esta complejidad se traslada a los procesos de gestión y desarrollo, que generan una serie de problemas durante el desarrollo software, como las desviaciones en tiempo y coste, conflictos entre *stakeholders*, el producto que no cumple los mínimos aceptables de calidad o funcionalidad, etc [Atk99]. Debido a que la reiteración en los mismos errores que hacen fallar los proyectos software [Gro09a], algunos autores han propuesto una serie de factores críticos de éxito que pueden ser resumidos en los siguientes [Jur99, WF02, PC87, JI07, Ker09]:

- Objetivos y metas claras. El gestor de proyecto debe ser capaz de detectar, valorar y asegurar los objetivos del proyecto.
- Presupuesto/planificación realista. La estimación de costes y tiempos es crítica, y debe ajustarse a la realidad para evitar comportamientos imprevisibles durante la ejecución del proyecto.

- Soporte de gestores con experiencia. Las organizaciones deberían proporcionar un adecuado soporte a la gestión de los proyectos por parte de las personas con mayor experiencia.
- Comunicaciones. Es necesario proporcionar canales de comunicación adecuados para completar correctamente las tareas colaborativas o interrelacionadas.
- Gestión del cambio/control/seguimiento. Llevar un control estricto sobre el proyecto siempre ha sido uno de los mejores métodos para anticipar posibles desviaciones, y por lo tanto, para la gestión adecuada cuando hay riesgo de que se produzcan.

A medida que avanzan los sistemas software y hardware, se proponen nuevos métodos formales para adaptarse a cambios de contexto. A su vez, también se incrementa la complejidad en la gestión del proyecto. Desafortunadamente, la gestión del proyecto no ha evolucionado lo suficiente como para incluir en las herramientas de gestión las necesidades de estos avances.

Varios autores coinciden en las deficiencias que surgen cuando la complejidad de los proyectos es alta, como [GA08], donde se destaca la incertidumbre, la complejidad inherente y las interacciones. De hecho, muchos profesionales enfocan la complejidad desde un punto de vista simple, lo que incluye también el tratamiento desde el punto de vista de la gestión y la visualización de la representación del proyecto [MGL11]. Alineado con las hipótesis iniciales de este proyecto de tesis, el soporte a la decisión requiere de la capacidad de caracterizar el estado actual del proyecto desde el punto de vista del que toma las decisiones.

Desde el punto de vista del doctorando, es necesario resolver los problemas de gestión actuales para hacer que el software pueda ser cuantificable, mensurable, evaluable y predecible. Este es el objetivo general que está detrás de cualquier disciplina. Para alcanzar este objetivo, es necesario proporcionar soluciones concretas a esos factores críticos de éxito. En esta tesis, se han considerado dos factores críticos principales: las desviaciones en los proyectos; y los canales de comunicación.

Las soluciones propuestas se centran en cómo los distintos participantes en el proyecto visualizan el progreso del proceso de desarrollo. La visualización proporciona

la única herramienta de comunicación entre los participantes en el proyecto, y el punto de vista del doctorando se basa en la tesis de que la solución en las desviaciones y los problemas en la comunicación se pueden resolver trabajando tanto sobre la visualización de la información como sobre el modelado de dicha información.

También es interesante el análisis de la gestión de los proyectos software en los desarrollos actuales. Como se ha expuesto anteriormente, las empresas están cada día más en la línea de gestión coordinada de proyectos, frente a la gestión unitaria de los proyecto. Sin embargo, no todas las compañías pueden adaptar de manera adecuada sus procesos de gestión a un entorno multiproyecto. La mayoría de problemas surgidos en estos entornos multiproyectos están relacionados con la gestión. Elonen y Artto [EA03] detectaron un conjunto de seis posibles causas de problemas relacionados con la gestión: (i) actividades a nivel de proyecto, (ii) actividades a nivel de portafolio, (iii) gestión del negocio orientado al proyecto, (iv) gestión de la información, (v) responsabilidades, roles y asignaciones, y (vi) recursos, competencias y métodos.

8.3. Objectivos alcanzados

En el primer capítulo de esta tesis se enumeraron los objetivos. El objetivo de la investigación era proporcionar una forma integrada de resolver los factores críticos de fallo. Es más, algunos autores incluso aseguran que las estimaciones vagas supone uno de los principales factores de fallo de los proyectos [Gla02]. Nosotros vemos que desarrollo software como una nueva disciplina que necesita dotarse de herramientas, metodologías y modelos adecuados. Esto incluye tanto a la disciplina directamente relacionada con el desarrollo, como a la disciplina en la gestión del desarrollo. Entendemos como disciplina el conjunto de herramientas, metodologías, métodos, modelos y procesos.

Por el momento, la gestión de proyectos software ha sido utilizada como parte de herramientas genéricas y modelos más relacionados con la gestión de proyectos, y no tan enfocados a los proyectos software. Sin embargo, hay una gran diferencia entre los procesos definidos para los proyectos software y los procesos definidos

para otras áreas de conocimiento más maduras. Así, la naturaleza del desarrollo software es diferente debido [RB06]:

- La experiencia es un grado necesario en cada una de las tareas del proyecto.
- El rendimiento que se obtiene puede variar en su duración, incluso cuando se están evaluando tareas similares en entornos similares y con actores con el mismo perfíl.
- Los fallos durante las etapas iniciales del proyecto (en la definición de requisitos o en la fase de diseño) son relativamente comunes. Si no se resuelven pronto, es seguro que van a causar desviaciones en el proyecto.

Por otro lado, otro de los objetivos de esta tesis era proporcionar las herramientas que permitieran gestionar adecuadamente el proceso de gestión, y por lo tanto, que las actividades pudieran ser validadas durante el proyecto. Estas herramientas también debían ser capaces de proporcionar información suficiente a los participantes del proyecto, estableciendo los correspondientes canales de comunicación.

8.4. Contribuciones a la Investigación

El enfoque inicial de esta investigación era el desarrollo de técnicas conceptuales que permitieran representar la información de los diferentes componentes que toman parte en los proyectos software. Los componentes a representar eran las actividades del proyecto, las personas que participan en el proyecto y los productos desarrollados, entre otros. El hecho no era únicamente visualizar gráficamente la información, sino también proporcionar una representación del conocimiento experto sobre el proyecto. El hecho de visualizar la información de un proyecto podría resultar una tarea sencilla a priori: simplemente asignando cada uno de los componentes a un símbolo iconográfico y definiendo las interacciones entre componentes, como en DesignNet [LH89]. DesignNet era capaz de mostrar varios de los elementos principales del software, como productos, entregables, informes, actividades o las personas. Esta representación no fue una mala propuesta, pero ha

sido barrida por otras herramientas tradicionales, como PERT/CPM o el diagrama Gantt. Uno de los errores de DesignNet fue el exceso de información mostrada, con su dificultad para extenderlo a entornos de grandes proyectos.

La primera parte de la investigación presentada en esta tesis se ha centrado en la detección de las características deseables de los modelos de representación, y cómo estas características influencian la aceptación y uso de uno u otro modelo de representación. Puede ser adecuado el uso de ciertos modelos genéricos de visualización de la información en dominios como la construcción, el entorno miliar o el entorno médico, pero en la ingeniería del software se hace necesario desarrollar modelos personalizados adecuados a las características inherentes del desarrollo software. La utilización de modelos genéricos no facilita la representación adecuada sobre los proyectos software. Es más, la información sobre los recursos humanos supone una característica deseable de mostrar, debido a que participan activamente en el desarrollo software.

En la primera parte de la tesis se han propuesto dos modelos de visualización: (i) PARMA, y (ii) el diagrama Gantt extendido.

PARMA –(*Project-Activity Representation Matrix*)– [RBD03] es un modelo que muestra en el espacio de una matriz tanto las actividades como los recursos humanos. La combinación de recursos y actividades son las asignaciones, y estas asignaciones se muestran con un símbolo iconográfico en la matriz. El color de círculo cambia según la evolución de la actividad, desde blanco –que representa que la actividad no ha comenzado–, hasta negro –cuando la tarea ha finalizado–, utilzando la escala de grises para determinar el avance de la tarea. La figura 8.1 muestra un ejemplo de un modelo de representación PARMA. En este modelo se han definido dos partes diferenciadas: (i) el modelo matricial, y (ii) las tarjetas de asignación.

El modelo matricial muestra en una disposición de celdas las asignaciones de las tareas a cada recurso humano. Las columnas corresponden a los recursos humanos que participan en el proyecto, mientras que las filas muestran las actividades a realizar. Esto permite definir asignaciones cuando una actividad se realiza por una persona, o lo que es lo mismo, cuando una persona es asignada para la ejecución de una actividad.

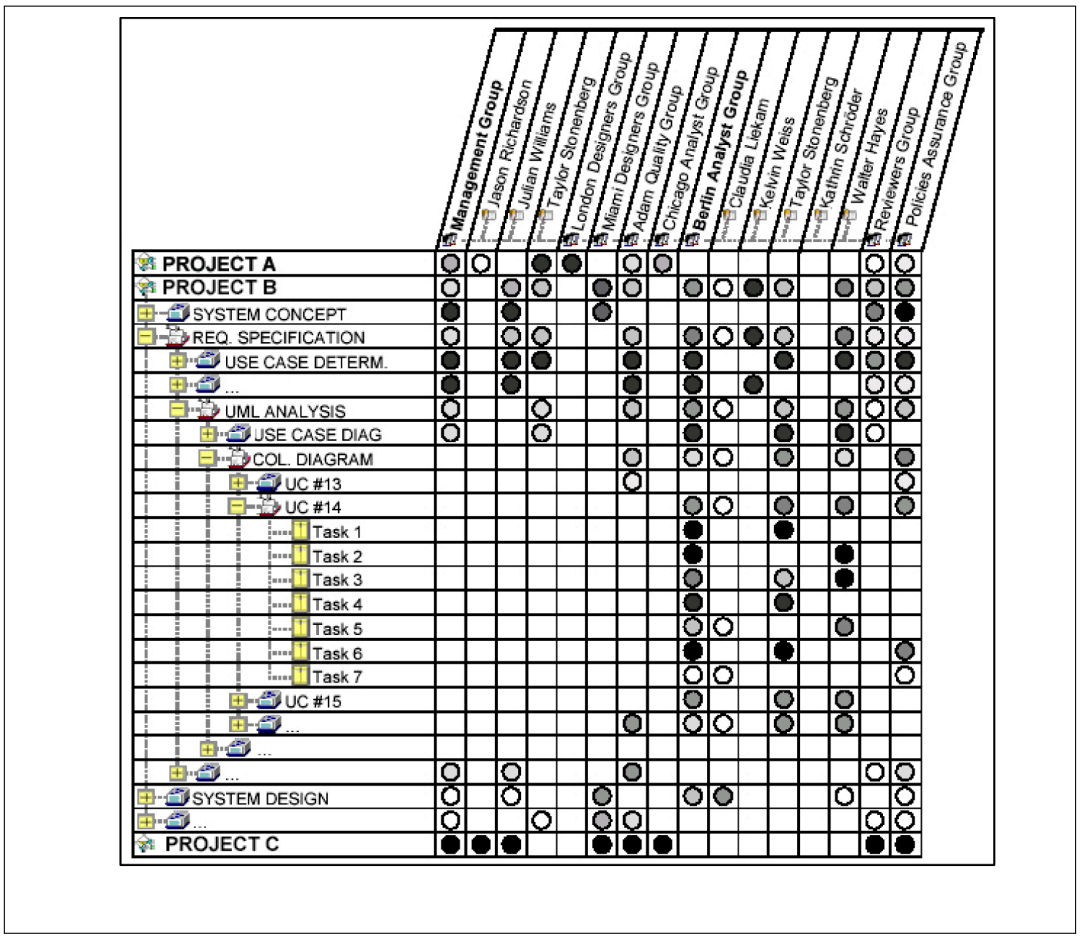


FIGURA 8.1: Una representación simulada del model PARMA

Las tarjetas de asignación muestran toda la información relacionada con la asignación. Concretamente, la información mostrada son tanto parámetros generales (nombre de la tarea, descripción, inicio, etc.), los diferentes niveles de responsabilidad sobre la tarea, estimaciones (de coste y tiempo), dependencias, productos de la actividad, y la necesidad de información de otras actividades o recursos humanos.

Esta propuesta es dificilmente extensible a grandes proyectos, debido a la necesidad de mostrar todas las actividades y los participantes en un proyecto. Otro inconveniente que presenta este modelo es que las asignaciones pueden proporcionar información útil sobre la evolución, aunque esto no facilita la resolución de los problemas asociados al proceso de gestión, debido a que es difícil para los usuarios dar una estimación realista del grado de avance en un determinado punto en el tiempo de la tarea.

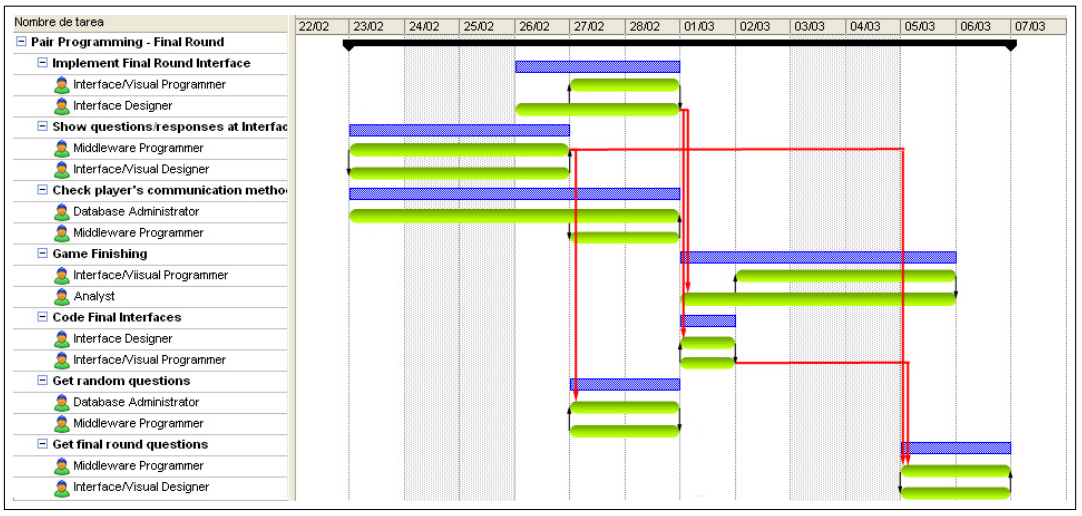


FIGURA 8.2: Un ejemplo de representación del diagrama Gantt extendido

El diagrama Gantt extendido [RBD06] se basa sobre el diagrama Gantt tradicional, pero incluyendo a los recursos humanos como el siguiente nivel de refinamiento de las actividades. Por ejemplo, en la figura 8.2 se muestra cómo esta representación proporciona un nuevo punto de vista de diagrama Gantt. No proporciona únicamente una representación de los recursos humanos como un componente a representar, sino que también determina las diferentes necesidades de comunicación entre los participantes en el proyecto, trasladando las dependencias entre actividades a dependencias entre recursos humanos.

Estas dos propuestas mostradas en la primera parte han supuesto soluciones particulares donde los recursos humanos pasan a formar parte de la información visualizada. Sin embargo, aún se puede continuar la investigación sobre los modelos propuestos. Concretamente, ambos modelos podrían ser desarrollados y probados en entornos de proyectos reales para su validación. Sin embargo, la extensión de esta investigación queda fuera del alcance de esta tesis.

En la segunda parte, se ha estudiado y desarrollado desde el punto de vista de la representación del conocimiento una ontología que representase los procesos de gestión de proyectos. El resultado de este desarrollo ha sido *Project Management Ontology* [RBD08], una ontología que presenta los principales conceptos y relaciones basada en el conocimiento experto obtenido del PMBOK. Esta representación

del conocimiento proporciona una visión semántica de la información del proyecto utilizando el enfoque de gestión.

En esta propuesta se presenta un conjunto de ontologías básicas que son necesarias para comenzar a capturar y salvar la información de gestión de los proyectos en un formato independiente de las aplicaciones utilizadas, facilitando así mismo el proceso de interoperabilidad e intercambio de información. Aunque el proceso de desarrollo de ontologías es duro y bastante largo, la necesidad de proporcionar y obtener el conocimiento disponible sobre la gestión de proyectos es fundamental para gestionar de una manera cada vez más eficiente. De hecho, en la actualidad, existe una gran cantidad de grupos de investigación dedicados a la gestión del conocimiento.

Así mismo, durante la investigación asociada a la representación semántica, se han mostrado las principales características de *Project Management Ontology* (PMO). PMO está compuesta de varias ontologías, donde cada una desarrolla una parte del conocimiento de la gestión de proyectos. Esta estructuración por partes de conocimiento, ha permitido trabajar individualmente con cada una de ellas (dividiendo el problema de la representación del conocimiento en partes más pequeñas), lo que ha facilitado su posterior proceso de integración. Así mismo, en la actualidad se está tratando de unir esta ontología con ontologías de otras áreas de conocimiento.

El principal problema para la validación de una ontología aparece cuando dicha ontología no está suficientemente evaluada, no esté consensuada, o que no se haya comunicado de una manera efectiva al colectivo principalmente interesado en la utilización del conocimiento inherente a la ontología. Por ello, es necesario establecer un marco para que el proceso de desarrollo de la ontología proporcione una verificación y evaluación de su contenido. La validación se ha realizado a través del lenguaje de reglas SWRL [RBRD11]. Para la validación se ha actuado en dos frentes: (i) la capacidad de razonamiento a través de consultas, y (ii) la inclusión de reglas que crean nuevo conocimiento cuando se producen los condicionantes del antecesor de la regla.

Tras este proceso de validación, el mapeado realizado de los procesos de gestión se han demostrado válidos al nivel descrito en los objetivos de la tesis. Sin embargo, sería deseable que los conceptos definidos en PMO puedan ser normalizados en

alguno de los sistemas de conocimiento denominados *Upper-Ontologies* (Dublin Core[cMI11], DOLCE [MBG+07], OpenCyc 2.0 [UOWGS09], SUMO [Pea11]) que proporcionen una definición de los conceptos presentes en la ontología usando conceptos y meta-datos generales y que permitan la estructuración.

Otra de las posibles mejoras, que también queda fuera del alcance de esta tesis, es la futura explotación de los datos (*data mining*) en el software de gestión de proyectos. De esta manera, no sólo se dispondría de las herramientas que proporciona dicho software, sino que se podrían tener variables reales que se obtendrían a través de inferencia sobre los datos para el apoyo a la decisión en los proyectos de ingeniería del software.

Bibliografía

[AAH+06] Sven Abels, Frederik Ahlemann, Axel Hahn, Kevin Hausmann, and Jan Strickmann. PROMONT - A project management ontology as a reference for virtual project organizations. In *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, volume 4277 of *Lecture Notes in Computer Science*, pages 813–823. Springer Berlin / Heidelberg, 2006.

[AMBD04] Alain Abran, James W. Moore, Pierre Bourque, and Robert Dupuis, editors. *Guide to the Software Engineering Body of Knowledge*. IEEE Computer Society Press, 2004. 2004 Version.

[AR06] Nitin Agarwal and Urvashi Rathod. Defining 'success' for software projects: An exploratory revelation. *International Journal of Project Management*, 24(4):358–370, 2006.

[Atk99] Roger Atkinson. Project management: cost, time and quality, two best guesses and a phenomenon, its time to accept other success criteria. *International Journal of Project Management*, 17(6):337–342, 1999.

[ATS99] David Arditi, Onur B. Tokdemir, and Kangsuk Suh. Effect of learning on line-of-balance scheduling. *International Journal of Project Management*, 19(5):265–277, 1999.

[BCM+07] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge Univerity Press, 2 edition, 2007.

[BGM04] Dan Brickley, R.V. Guha, and Brian McBride. Extensible Markup Language (XML) 1.0 (Fifth Edition). `http://www.w3.org/TR/rdf-schema/`, 2004. W3C Recommendation 10 February 2004.

[BI11] Edward W.N. Bernroider and Milen Ivanov. It project management control and the control objectives for it and related technology (cobit) framework. *International Journal of Project Management*, 29(3):325–336, 2011.

[BPSM+08] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, and François Yergeau. Extensible Markup Language (XML) 1.0 (Fifth Edition). `http://www.w3.org/TR/xml/`, 2008. W3C Recommendation 26 November 2008.

[BPT+10] Harold Boley, Adrian Paschke, Said Tabet, Benjamin Grosof, Nick Bassiliades, Guido Governatori, David Hirtle, Omair Shafiq, and Monika Machunik. Schema specification of RuleML 1.0. `http://ruleml.org/1.0/`, 2010.

[Bro95] Frederic P. Brooks. *The Mythical Man-Month (anniversary ed.)*. Addison-Wesley, 1995.

[BvHH+04] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. OWL Web Ontology Language Reference. `http://www.w3.org/TR/owl-ref/`, 2004. W3C Recommendation 10 February 2004.

[CGM+89] D. E. Comer, David Gries, Michael C. Mulder, Allen Tucker, A. Joe Turner, and Paul R. Young. Computing as a discipline. *Communications of the ACM*, 32(1):9–23, 1989.

[CGSL03] Jinxing Cheng, Michael Gruninger, Ram D. Sriram, and Kincho H. Law. Process specification language for project scheduling information exchange. *International Journal of IT in Architecture, Engineering and Construction*, 1(4):307–328, 2003.

[Cla11a] Peter Clark. KBS/ontology projects worldwide: Some ongoing KBS/Ontology projects and groups. `http://www.cs.utexas.edu/users/mfkb/related.html`, 2011.

[Cla11b] Peter Clark. The open biological and biomedical ontologies. `http://www.obofoundry.org/`, 2011.

[cMI11] Dublin core Metadata Initiative. Dcmi metadata terms. `http://dublincore.org/`, 2011.

[Com11] Petri Nets Steering Committee. Petri nets world: Online services for the international petri nets community. `http://www.informatik.uni-hamburg.de/TGI/PetriNets/`, 2011.

[Con96] Dan Conolly. Overview of SGML resources. `http://www.w3.org/MarkUp/SGML/`, 1996. ISO standard: "ISO 8879:1986 Information processing — Text and office systems — Standard Generalized Markup Language (SGML)".

[CSH06] Namyoun Choi, Il-Yeol Song, , and Hyoil Han. A Survey on Ontology Mapping. *ACM SIGMOD Record*, 35(3):34–41, 2006.

[CTB91] Ken Currie, Austin Tate, and South Bridge. O-Plan: The Open Planning Architecture. *Artificial Intelligence*, 51(1):49–86, 1991.

[Dep68] Department of Defense. *MIL-STD-881: Military Standard. Work Breakdown Structures for Defence Material Items*, 1968. MIL-STD-881.

[Dep93] Department of Defense. *MIL-STD-881-B: Military Standard. Work Breakdown Structures for Defence Material Items*, March 1993. MIL-STD-881-B.

[Dep05a] Department of Defense. *MIL-HDBK-881A: Department of Defence Handbook. Work Breakdown Structures for Defence Material Items*, July 2005. MIL-STD-881A.

[Dep05b] Department of Defense. *MIL-HDBK-881A: Department of Defence Handbook. Work Breakdown Structures for Defence Material Items*, July 2005. MIL-STD-881A.

[DHC07] Hai Dong, Farookh Khadeer Hussain, and Elizabeth Chang. Application of Protégé and SPARQL in the field of project knowledge management. *International Conference on Systems and Networks Communication*, pages 74–80, 2007.

[Dre80] Joachim Dressler. Construction management in west germany. *Journal of Construction Division*, 106(4):477–487, 1980.

[DS00] Javier Dolado and Luis Fernández Sanz, editors. *Medición para la Gestión en Ingeniería del Software*. Ra-Ma, 2000.

[EA03] Suvi Elonen and Karlos A. Artto. Problems in managing internal development projects in multi-project environments. *International Journal of Project Management*, 21(6):395–402, 2003.

[EJ03] Mats Engwall and Anna Jerbrant. The resource allocation syndrome: the prime challenge of multi-project management? *International Journal of Project Management*, 21(6):403–409, 2003.

[ES04] Mark Ehrig and York Sure. Ontology mapping: An integrated approach. In *First European Semantic Web Symposium (ESWS 2004)*, pages 74–91, 2004.

[GA08] Joana Geraldi and Gerald Adlbrecht. On faith, fact, and interaction in projects. *IEEE Engineering Management Review*, 36(2):34–49, 2008.

[GBC+06] Félix García, Manuel F. Bertoa, Coral Calero, Antonio Vallecillo, Francisco Ruiz, Mario Piattini, and Marcela Genero. Towards a consistent terminology for software measurement. *Information and Software Technology*, 48(8):631–644, 2006.

[GDD09] Dragan Gasevic, Dragan Djuric, and Vladan Devedzic. *Model Driven Engineering and Ontology Development*. Springer-Verlag, 2 edition, 2009.

[Gla02] Robert L. Glass. *Facts and Fallacies of Software Engineering*. Addison-Wesley Professional, 2002.

[Gmb04] Jenz & Partner GmbH. Business Management Ontology, 2004.

[Gre65] C. E. Greene. Communication with automata. Technical Report RADC-TR-65-377, Rome Air Development Center, Griffiss Air Force Base, 1965. Translation of thesis "Kommunikation mit automaten".

[Gro09a] Standish Group. The 2009 chaos report. Technical report, Standish Group International, 2009.

[Gro09b] W3C OWL Working Group. OWL 2 Web Ontology Language Document Overview. `http://www.w3.org/TR/owl2-overview/`, 2009. W3C Recommendation 27 October 2009.

[Gro10] Object Management Group. Unified Modeling Language (UML) specification. `http://www.omg.org/spec/UML/Current`, 2010. UML 2.3.

[Gru93] Tom R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[Han95] Robert P. Hanrahan. The idef process modelling methodology. *Crosstalk: The Journal of Defense Software Engineering*, 8(6):19–23, June 1995.

[HKM01] Harald Holz, Arne Könnecker, and Frank Maurer. Task-specific knowledge manage-ment in a process-centred see. In *Lecture Notes in Computer Science: Proceedings of the Third International Workshop on Advances in Learning Software Organizations*, volume 2176, pages 163–177. Springer-Verlag, 2001.

[HP06] Jerry R. Hobbs and Feng Pan. Time Ontology in OWL. `http://www.w3.org/TR/owl-time/`, 2006. W3C Working Draft 27 September 2006.

[HPSB+04] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. SWRL: A Semantic Web Rule Language combining OWL and RuleML. `http://www.w3.org/Submission/SWRL/`, 2004.

[Hum99] Watts S. Humphrey. *Introduction to the Team Software Process*. The SEI Series in Software Engineering. Addison Wesley Professional, 1999.

[Ins06] Project Management Institute. *Practice Standard for Work Breakdown Structures*. Project Management Institute, 2nd edition, 2006.

[Ins08] Project Management Institute. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*. Project Management Institute, 4th edition, 2008.

[Ins11] National Cancer Institute. Nci thesaurus. `http://ncit.nci.nih.gov/ncitbrowser/`, 2011.

[JC99] Mark St. John and Michael B. Cohen. Using 2-d vs. 3-d displays: Gestalt and precision tasks. In *Proceedings of the 43rd Human Factors and Ergonomics Society Meeting*, volume 5, pages 1318–1322, 1999.

[Jen03] Dieter E. Jenz. BPMO tutorial: Defining a private business process in a knowledge base. Tutorial, Jenz & Partner GmbH, 2003.

[Jen04] Dieter E. Jenz. Business Management Ontology. version 1.0: Release notes. Release notes, Jenz & Partner GmbH, 2004.

[JI07] Kumar Neeraj Jha and K.C. Iyer. Commitment, coordination, competence and the iron triangle. *International Journal of Project Management*, 25(5):527–540, 2007.

[Jur99] Jaak Jurison. Software project management: The manager's view. *Communications of the Association for Information Systems*, 2(17), September 1999.

[Kal01] Yannis Kalfoglou. *Handbook of Software Engineering and Knowledge Engineering*, volume 1, chapter Exploring ontologies, pages 863–887. World Scientific Publishing Company, 2001.

[KBS93] KBSI. Information definition for function modelling (IDEFØ): Federal information processing standard 183. Technical report, Knowledge Based Systems, Inc., 1993.

[KBS94] KBSI. Information integration for concurrent engineering (IICE): IDEF5 method report. Technical report, Knowledge Based Systems, Inc., 1994.

[KC04] Graham Klyne and Jeremy J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. `http://www.w3.org/TR/rdf-concepts/`, 2004.

[Ker09] Harold Kerzner, editor. *Project Management: a systems approach to planning, scheduling, and controlling*. Wiley, 10th edition, 2009.

[KS03] Yannis Kalfoglou and Marco Schorlemmer. Ontology Mapping: The State of the Art. *Knowledge Engineering Review*, 18(1):1–31, 2003.

[Leg09] Martha Legare. Using Gantt charts: Learn what a gantt chart shows and how to make decisions using it. `http://www.horton.com/portfolio/gantt/index.htm`, 2009.

[LH89] Lung-Chun Liu and Ellis Horowitz. A formal model for software project management. *IEEE Transactions on Software Engineering*, 15(10):1280–1293, 1989.

[Lid09] Udo Lidemann. The Design Structure Matrix (DSM). `http://www.dsmweb.org/`, 2009.

[LSH+99] Sean Luke, Lee Spector, James Hendler, Jeff Heflin, and David Rager. Simple HTML Ontology Extensions – organization ontology. `http://www.cs.umd.edu/projects/plus/SHOE/onts/org1.0.html`, 1999.

[Mar02] Robert C. Martin, editor. *Agile Software Development: Principles, Patterns and Practices*. Prentice Hall, 2002.

[MBG+07] Claudio Masolo, Stefano Borgo, Aldo Gangemi, Nicola Guarino, Alessandro Oltramari, and Luc Schneider. The wonderweb library of foundational ontologies, 2007. Report.

[McD01] Norman A. McDaniel. *Scheduling Guide for Program Managers*. Defense Systems Management College Press, October 2001.

[Mcl05] Kevin Mcleish. Introduction to petri nets. `http://www.peterlongo.it/Italiano/Informatica/Petri/index.html`, 2005.

[MGL11] Guillaume Marques, Didier Gourca, and Matthieu Laurasa. Multi-criteria performance analysis for decision making in project management. *International Journal of Project Management*, 2011. Article in Press.

[MM05] D. A. Marca and Clement L. McGowan. *IDEF0 and SADT: A Modeller's Guide*. OpenProcess, Inc., 2005.

[MW97a] Karen L. Myers and David E. Wilkins. The Act-Editor User Guide. a manual for version 2.2. Manual, SRI International Artificial Intelligence Center, 1997.

[MW97b] Karen L. Myers and David E. Wilkins. The Act formalism. version 2.2. Manual, SRI International Artificial Intelligence Center, 1997.

[NG88] David Notkin and William G. Grisworld. Extension and software development. In *Proceedings of the 10th International Conference on Software Engineering*, volume 2176, pages 274–283. IEEE Computer Society Press, 1988.

[NM01] Natalya F. Noy and Deborah L. McGuinness. Ontology development 101: A guide to creating your first ontology. Technical report, Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, 2001.

[Noy09] Natalya F. Noy. Ontology mapping. In Steffen Staab and Dr. Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 573–590. Springer Berlin Heidelberg, 2009.

[OMG08] OMG. *Software & Systems Process Engineering Meta-Model Specification. Version 2.0.* Object Management Group, 2008.

[oP10] Protégé Community of Practice. Protégé community wiki: Protégé ontology library. `http://protege.cim3.net/`, 2010.

[OPSE+10] Eila Ovaska, Susanna Pantsar-Syväniemi, Antti Evesti, Katja Henttonen, Daniele Manzaroli, Paolo Bellavista, Alessandra Toninelli, Rebecca Montanari, Carlo Giannelli, Alfredo D'Elia, Guido Zamagni, Fabio Vergari, Sara Bartolini, Luca Roffia, Federico Spadini, Tullio Salmon Cinotti, Roberto Baldoni, Sirio Scipioni, Leonardo Querzoni, Paolo Pucci, and Giorgio Laura. Interoperability platform principles. Deliverable, SOFIA Project Consortium, 2010.

[OTN+07] Martin O'Connor, Samson Tu, Csongor Nyulas, Amar Das, , and Mark Musen. *Advances in Rule Interchange and Applications (RuleML2007)*, volume 4824, chapter Querying the Semantic Web with SWRL, pages 155–159. Springer-Verlag, 2007.

[Pag04] Michael Pagels. DAML ontology library. `http://www.daml.org/ontologies/`, 2004.

[PC87] Jeffrey K. Pinto and Jeffrey G. Covin. Critical factors in project implementation: a comparison of construction and r & d projects. *IEEE Transactions on Engineering Management*, 34(1):22–27, 1987.

[Pea11] Adam Pease. Suggested Upper Merged Ontology (SUMO). `http://www.ontologyportal.org/`, 2011.

[Pet62] Carl Adam Petri. *Kommunikation mit automaten.* PhD thesis, Schriften des IIM, Heft 2, Institut für Instrumentelle Mathematik, an der Universität Bonn, 1962.

[Pet77] James L. Peterson. Petri nets. *ACM Computing Surveys (CSUR)*, 9(3):223–252, September 1977.

[Phi04] Dwayne Phillips. *The software project manager's handbook: principles that work at work*. Wiley-IEEE Computer Society Press, 2nd edition, July 2004.

[Pol93] Spiro N. Pollalis. *Computer Aided Project Management: A Visual Scheduling and Control System*. Vieweg Verlag, Wiesbaden, 1993.

[PSHH04] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. OWL Web Ontology Language Semantics and Abstract Syntax. `http://www.w3.org/TR/owl-semantics/`, 2004.

[RB06] Fran J. Ruiz-Bertol. Management challenges in software projects: Improving software project management and representation in multiproject environments. In *ICEIS Doctoral Consortium*, pages 84–93, 2006.

[RBD03] Fran J. Ruiz-Bertol and Javier Dolado. Parma: Un modelo incremental para la representación y gestión de proyectos en ingeniería del software. In Ernesto Pimentel, Nieves R. Brisaboa, and Jaime Gomez, editors, *JISBD: Actas VIII Jornadas de Ingeniería del Software y Bases de Datos*, pages 359–368, 2003.

[RBD06] Fran J. Ruiz-Bertol and Javier Dolado. Human resource representation in gantt charts. In *Software Quality Management XIV: Perspectives in Software Quality (SQM 2006)*, pages 219–230, 2006.

[RBD08] Fran J. Ruiz-Bertol and Javier Dolado. Una ontología para la gestión del conocimiento de proyectos software. *Revista Española de Innovación, Calidad e Ingeniería del Software*, 4(1):6–22, 2008.

[RBDA07] Fran J. Ruiz-Bertol, Javier Dolado, and Alain Abran. Ontology development for software engineering knowledge management. In *Workshop on Fundamentals of Software Engineering*, 2007.

[RBRD11] Fran J. Ruiz-Bertol, Daniel Rodríguez, and Javier Dolado. Applying rules to an ontology for project management. In *XVI Jornadas de Ingeniería del Software y Bases de Datos*, 2011. Submitted.

[RdA06] P.E.R. Rabelo and S.R.L. do Amorim. Ontology, management of project process, and information technologies. In *Proceedings of the 6th European Conference on Product and Process Modelling - eWork and eBusiness in Architecture, Engineering and Construction, ECPPM 2006*, pages 295–302, 2006.

[RGS10] Daniel Rodríguez, Elena García, and Salvador Sánchez. Defining software process model constraints with rules using owl and swrl. *International Journal of Software Engineering and Knowledge Engineering*, 20(4):1–16, 2010.

[RT05] Roberta Russell and Bernard W. Taylor, editors. *Operations Management : Quality and Competitiveness in a Global Environment*. John Willey & Sons, 2005.

[Rus05] Taylor Russell. *Operations Management: Quality and Competitiveness in a Global Environment*. Wiley, 5th edition, 2005.

[SA09] Demetrios Sarantis and Dimitris Askounis. A project management ontology as a reference for egovernment projects. In *International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 1–8, 2009.

[SBF98] Rudi Studer, V. Richard Benjamins, and Dieter Fensel. Knowledge engineering: Principles and methods. *Data and Knowledge Engineering*, 25(1–2):161–197, 1998.

[SBF11] Christopher Schmidt, Uldis Bojars, and Sergio Fernández. SpecGen: ontology specification generator tool. `http://forge.morfeo-project.org/wiki_en/index.php/SpecGen`, 2011.

[SCA09] Demetrios Sarantis, Yannis Charalabidis, and Dimitris Askounis. An ontology for stakeholder collaboration and knowledge exploitation in

e-government project management. In *Proceedings of the 3rd international conference on Theory and practice of electronic governance*, pages 61–67, 2009.

[SGCL99] Craig Schlenoff, Michael Gruninger, Mihai Ciocoiu, and Jintae Lee. The essence of the process specification language. *Transactions of the Society for Computer Simulation*, 16(4):204–216, 1999.

[Sla56] E.L. Slagle. Prestige and progress report. *The Journal of Industrial Engineering*, 7(1), 1956.

[SMJ98] Yuval Shahar, Silvia Miksch, and Peter Johnson. The asgaard project: a task-specific framework for the application and critiquing of time-oriented clinical guidelines. *Artificial Intelligence in Medicine*, 14(1–2):29–51, 1998.

[SSP99] Harry Scarbrough, Jacky A. Swan, and John Preston. Knowledge management: a literature review: Issues in people management. Technical report, Institute of Personnel and Development, 1999.

[Tat95] Austin Tate. Characterizing plans as a set of constraints —the ⟨i-nova⟩ model— a framework for comparative analysis. *SIGART Bull.*, 6(1):26–32, 1995.

[Tat98] Austin Tate. Roots of SPAR — shared planning and activity representation. *Knowledge Engineering Review*, 13(1):121–128, 1998.

[Tha97] Richard H. Thayer, editor. *Software Engineering Project Management*. Wiley-IEEE Computer Society Press, 2nd edition, 1997.

[TYD+09] H. Ping Tserng, Samuel Y.L. Yin, R.J. Dzeng, B. Wou, M.D. Tsai, and W.Y. Chen. A study of ontology-based risk management framework of construction projects through project life cycle. *Automation in Construction*, 18(7):994–1008, 2009.

[Uni11] Princeton University. WordNet: A lexical database for English. `http://wordnet.princeton.edu/`, 2011.

[UOWGS09] Standard Upper Ontology Working Group (SUO WG). OpenCyc. `http://opencyc.org/`, 2009.

[WADD03] Cornelius Wille, Alain Abran, Jean Marc Desharnais, and Reiner R. Dumke. The quality concepts and sub concepts in swebok: An ontology challenge. In *13th International Workshop on Software Measurement (IWSM 2003)*, 2003.

[WDC+04] Xiaohang Wang, Jin Song Dong, ChungYau Chin, Sanka R. Hettiarachchi, and Daqing Zhang. Semantic space: An infrastructure for smart spaces. *IEEE Pervasive Computing*, 3(3):32–39, 2004.

[WF02] Diana White and Joyce Fortune. Current practice in project management - an empirical study. *International Journal of Project Management*, 20(1):1–11, January 2002.

[WK00] Yingxu Wang and Graham King. *Software Engineering Processes: Principles and Application*. CRC Press, USA, 2000.

[Zel78] M.V. Zelkowitz. Perspectives in software engineering. *ACM Computing Surveys*, 10(2):197–216, 1978.

Anexo A

Project Management Ontology – Core ontology

Project Management Ontology (PMO) es una ontología de dominio centrada en establecer un modelo formal de los procesos, actividades, herramientas y técnicas específicas de la gestión de proyectos. PMO proporciona una descripción completa de los términos fundamentales y características inherentes al manejo de la información asociada a la gestión, seguimiento, control y dirección de los proyectos, así como de los procesos, relaciones, restricciones y aserciones sobre los datos de proyectos.

En este anexo se presenta la documentación asociada de la ontología núcleo de PMO (*PMO-Core*), que define el conjunto de conceptos, relaciones, axiomas y atributos que forman la base para la gestión de proyectos. En esta ontología se incluyen conceptos como proyecto, fase, entregables, productos, o actividades.

PMO-Core es el núcleo de PMO. Todas las demás ontologías en PMO están directamente relacionadas con esta ontología. Contiene el vocabulario y la estructura básica para la definición de un proyecto genérico, en cualquier ámbito de aplicación. El principal concepto definido en la ontología es el *Proyecto*, que tiene asociado una serie de atributos inherentes, como su nombre, descripción, procesos a utilizar, criterios de calidad o hitos. Así mismo, un proyecto puede ser parte de un *Programa* o un *Portafolio*.

PMO-Core

PROJECT MANAGEMENT ONTOLOGY CORE

Thesis Version — 09 May 2011

This version:
http://in2gesoft.ehu.es/ont/pmo-core.owl
Last Update:
Date: 09 May 2011
Authors:
Fran Ruiz, Tecnalia
Contributors:
Daniel Rodrígez, Universidad de Alcalá de Henares
Javier Dolado, Universidad del País Vasco

Abstract

This specification defines the Project Management Core Ontology, providing the main concepts related with project management activities and processes.

Status of this Document

This is a work developed for the PhD Thesis of Francisco Javier Ruiz Bertol. Inherently, all ontologies are in constant evolution. This specification meets the adequate requirements needed to be presented for this PhD Thesis, in the sense that the mapping with Project Management Body of Knowledge processes are captured in the ontology derived from this specification.
If there is some suggestion or improvement that can be made for the ontology, please notify to the author (Fran.Ruiz@tecnalia.com), or contact with the supervisors of the thesis, Daniel Rodríguez (daniel.rodriguezg@uah.es) or Javier Dolado (javier.dolado@ehu.es). Thank you.

A.1 Introduction

Project Management Ontology (PMO), a set of ontologies that describe the generic knowledge about the project management. This ontology provides the first formal knowledge storage in project management domain. It is constructed in a modular and structured manner, so domain-specific ontologies for a concrete domain can be joined to PMO to create a specific ontology for project management in that domain. This can be done using ontology mapping or ontology merging, where common concepts can be compared between the ontologies. PMO includes:

- A basic taxonomy about projects common to all subject areas describing the skeleton of key terms needed to manage a project. This taxonomy includes relations of the type `isA` (subclasses) or `has` (composition) to define the parts in which is divided a project or a project component.
- A full vocabulary directly related to project management, but applicable or extensible to most of the subject areas. This vocabulary has been directly obtained from the knowledge appeared in the PMBOK.
- A knowledge base system (KBS) containing the expertise knowledge about how to manage adequately a project. This includes the structure, semantic content, and also individuals, all of them expressing guidance for correctly manage the project.
- A set of slot or properties applicable to every concept defined in the taxonomy and the vocabulary.
- The relationships among concepts defined in the taxonomy. These relationships are based on the knowledge used by the project managers.

A.1.1 Terminology and Notation

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

Namespace URIs of the general form "http://www.example.com/" represent some application-dependent or context-dependent URI as defined in RFC 2396.

The XML Namespace URIs that MUST be used by implementations of this specification are:

- http://ehu.es/pmo/pmo-core# - PMO-Core Ontology Namespace
- http://ehu.es/pmo/pmo-org# - PMO-Organization Ontology Namespace
- http://ehu.es/pmo/pmo-plan# - PMO-Planning Ontology Namespace
- http://ehu.es/pmo/pmo-process# - PMO-Process Ontology Namespace
- http://ehu.es/pmo/pmo-cost# - PMO-Cost Ontology Namespace

A.2. PMO Core ontology overview

The PMO Core definitions presented here are written using a computer language (RDF/OWL) that makes it easy for software to process some basic facts about the terms in the PMO Core Ontology, and consequently about the things described in this PhD Thesis.

A.2.1. Example

Here is a very basic document describing the project basics:

```
<pmo-core:Project rdf:ID="SOFIA">
  <pmo-core:identifier rdf:datatype="&xsd;string">SOFIA Project</pmo-core:identifier>
  <pmo-core:description rdf:datatype="&xsd;string">Smart Objects For Intelligent Applications</pmo-core:description>
  <pmo-core:hasComponents>
    <pmo-core:Work_Packate rdf:ID="WP1">
      <pmo-core:hasActivities>
        <pmo-core:ScheduleActivity rdf:ID="T1.1">
          <pmo-core:percent_complete rdf:datatype="&xsd;string">100%</pmo-core:percent_complete>
          <pmo-core:critical rdf:datatype="&xsd;boolean">true</pmo-core:critical>
        </pmo-core:ScheduleActivity>
      </pmo-core:hasActivities>
    </pmo-core:Work_Packate rdf:ID="WP1">
    <pmo-core:Work_Packate rdf:ID="WP2"/>
  </pmo-core:hasComponents>
  <pmo-core:hasPerformingOrganization>
    <pmo-org:Organization rdf:ID="Tecnalia"/>
  </pmo-core:hasPerformingOrganization>
</pmo-core:Project>
```

A.3. Cross-reference for PMO Core classes and properties

Class: pmo-core:Accept

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Accept`

- The act of formally receiving or acknowledging something and regarding it as being true, sound, suitable, or complete.

Class: pmo-core:Acceptance

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Acceptance`

- The act of formally receiving or acknowledging something and regarding it as being true, sound, suitable, or complete.

Class: pmo-core:Acceptance_Criteria

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Acceptance_Criteria`

- Those criteria, including performance requirements and essential conditions, which must be met before project deliverables are accepted.

sub-class-of:
    pmo-core:Criteria
in-range-of:
    pmo-core:acceptance_criteria

Class: pmo-core:Activity

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Activity`

- A component of work performed during the course of a project.

in-domain-of:
    pmo-core:outline
    pmo-core:identifier
in-range-of:
    pmo-core:hasActivities

Class: pmo-core:Application_Area

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Application_Area`

- A category of projects that have a common components significant in such projects, but are not needed or present in all projects. Application areas are usually defined in terms of either the product (i.e., by similar technologies or production methods) or the type of customer (i.e., internal versus external, government versus commercial) or industry sector (i.e., utilities, automotive, aerospace, information technologies). Application areas can overlap.

in-domain-of:
    pmo-core:definition_term
    pmo-core:description
    pmo-core:name

in-range-of:
pmo-core:hasApplicationAreas

Class: pmo-core:Apportioned_Effort

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Apportioned_Effort`

- Effort applied to project work that is not readily divisible into discrete efforts for that work, but which is related in direct proportion to measurable discrete work efforts.

sub-class-of:
pmo-core:Effort

Class: pmo-core:Approval

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Approval`

- The act of formally confirming, sanctioning, ratifying, or agreeing to something.

Class: pmo-core:Approved_Change_Request

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Approved_Change_Request`

- A change request that has been preocesses throught the integrated change control process and approved.

sub-class-of:
pmo-core:Change_Request
in-domain-of:
pmo-core:acceptance_criteria
pmo-core:approval_result
pmo-core:approval_date

Class: pmo-core:Baseline

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Baseline`

- The approved time phased plan, plus or minus approved project scope, cost, schedule, and technical changes.

in-domain-of:
pmo-core:baseline_start_date
pmo-core:baseline_finish_date
in-range-of:
pmo-core:hasBaseline
pmo-core:associatedBaselines

Class: pmo-core:Change_Request

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Change_Request`

- Requests to expend or reduce the project scope, modify policies, processes, plans, or procedures, modify costs or budgets, or revise schedules.

in-domain-of:
pmo-core:change_request_initiator
pmo-core:request_date
pmo-core:description
pmo-core:identifier
pmo-core:scope

Class: pmo-core:Component

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Component`

- A constituent part, element, or piece of a complex whole.

in-domain-of:
pmo-core:planned_value
pmo-core:hasEffort
pmo-core:actual_cost
pmo-core:budget
pmo-core:description
pmo-core:earned_value
pmo-core:hasBaseline
pmo-core:hasComponents
pmo-core:name
pmo-core:percent_complete
pmo-core:scope
in-range-of:

pmo-core:hasComponents

Class: pmo-core:Cost

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Cost`

- The monetary value or price of a project activity or component that includes monetary worth of the resources required to perform and complete the activity or component, or to produce the component. A specific cost can be composed of a combination of cost component including direct labor hours, other direct costs, indirect labor hours, other indirect costs, and purchased price.

in-domain-of:
pmo-core:neededMeasure
pmo-core:cost_type
pmo-core:description
in-range-of:
pmo-core:reserve
pmo-core:budget_cost
pmo-core:associatedCost
pmo-core:actual_cost
pmo-core:estimate

Class: pmo-core:Cost_Baseline

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Cost_Baseline`

- An approved time phased plan for costs. If the baseline is defined at schedule activity level, the cost is calculated in base to the cost needed to perform that schedule activity.

sub-class-of:
pmo-core:Baseline
in-domain-of:
pmo-core:associatedCost
pmo-core:hasCostBaselines
in-range-of:
pmo-core:hasCostBaselines

Class: pmo-core:Cost_Estimate

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Cost_Estimate`

- An estimation based in cost.

sub-class-of:
pmo-core:Estimate
in-domain-of:
pmo-core:reserve
in-range-of:
pmo-core:planned_value
pmo-core:budget_at_completion
pmo-core:budget
pmo-core:earned_value

Class: pmo-core:Criteria

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Criteria`

- Standards, rules, or test on which a judment or decision can be based, or by which a product, service, result or process can be evaluated.

in-range-of:
pmo-core:hasAcceptanceCriteria

Class: pmo-core:Critical_Activity

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Critical_Activity`

- Any schedule activity on a critical path in a project schedule. Most commonly determined by using the critical path method. Although some activities are "critical" in the dictionary sense, without being on the critical path, this meaning is seldom used in the project context.

sub-class-of:
pmo-core:Schedule_Activity

Class: pmo-core:Critical_Path_Method

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Critical_Path_Method`

- A schedule network analysis technique used to determine the amount of scheduling flexibility (the amount of float) on various logical network paths in the project schedule network, and to determine the minimum total project duration. Early start and finish dates are calculated by means of

a forward pass, using a specified start date. Late start and finish dates are calculated by means of a backward pass, starting from a specified completion date, which sometimes is the project early finish date determined during the forward pass calculation.

in-domain-of:
pmo-core:early_start_date
pmo-core:early_finish_date
pmo-core:late_finish_date
pmo-core:late_start_date
in-range-of:
pmo-core:hasCPM

Class: pmo-core:Currency

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Currency`

- An official currency issued by a government or national bank to measure the value of something.

in-domain-of:
pmo-core:symbol
pmo-core:name
in-range-of:
pmo-core:currency

Class: pmo-core:Defect

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Defect`

- An imperfection or deficiency in a project component where that component does not meet its requirements or specifications and needs to be either repaired or replaced.

in-range-of:
pmo-core:hasDefects

Class: pmo-core:Deliverable

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Deliverable`

- Any unique and verifiable product, result, or capability to perform a service that must be produced to complete a process, phase, or project.

sub-class-of:
pmo-core:Component
in-domain-of:
pmo-core:hasAcceptanceCriteria
pmo-core:hasDefects

Class: pmo-core:Discrete_Effort

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Discrete_Effort`

- Work Effort that is directly identifiable to the completion of specific work breakdown structure components and deliverables, and that can be directly planned and measured.

sub-class-of:
pmo-core:Effort

Class: pmo-core:Document

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Document`

- A medium and the information recorded thereon, that generally has permanence and can be read by a person or a machine.

sub-class-of:
pmo-core:Result

Class: pmo-core:Dummy_Activity

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Dummy_Activity`

- A schedule activity of zero duration used to show a logical relationship in the arrow diagramming method. Dummy activities are used when logical relationships cannot be completely or correctly described with schedule activity arrows.

sub-class-of:
pmo-core:Activity

Class: pmo-core:Duration

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Duration`

- The total number of work periods (not including holidays or other nonworking periods) required to complete a schedule activity or work breakdown structure component. Usually expressed as workdays or workweeks. Sometimes incorrectly equated with elapsed time.

sub-class-of:
    pmo-core:Measure
in-domain-of:
    pmo-core:duration_unit
in-range-of:
    pmo-core:lead
    pmo-core:lag
    pmo-core:duration
    pmo-core:remaining_duration
    pmo-core:actual_duration
    pmo-core:original_duration
    pmo-core:estimate

Class: pmo-core:Effort

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Effort`

- The number of labor units required to complete a schedule activity or work breakdown structure component

sub-class-of:
    pmo-core:Measure
in-domain-of:
    pmo-core:effort_unit
in-range-of:
    pmo-core:hasEffort
    pmo-core:estimate
    pmo-core:neededMeasure

Class: pmo-core:Estimate

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Estimate`

- A quantitative assessment of the likely amount or outcome. Usually applied to project costs, resources, effort, and durations and is usually preceded by a modifier. (i.e., preliminary, conceptual, feasibility, order-of-magnitue, definitive). It should always include some indicator of accuracy.

in-domain-of:
    pmo-core:estimate
    pmo-core:estimation_type
    pmo-core:accuracy

Class: pmo-core:Knowledge_Area

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Knowledge_Area`

- The concepts and facts of the subject of study, as well as relations among them and mechanisms for how to combine them to solve problems in that area.

in-domain-of:
    pmo-core:description
    pmo-core:name

Class: pmo-core:Logical_Relationship

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Logical_Relationship`

- A dependency between two project schedule activities, or between a project schedule activity and a schedule milestone.

in-domain-of:
    pmo-core:successorActivity
    pmo-core:predecessorActivity
    pmo-core:hasRelationshipType
in-range-of:
    pmo-core:hasLogicalRelationships

Class: pmo-core:Logical_Relationship_Type

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Logical_Relationship_Type`

- The set of possible types of logical or precedence relationships.

in-domain-of:
    pmo-core:description
    pmo-core:name

in-range-of:
pmo-core:hasRelationshipType

Class: pmo-core:Management_Contingency_Reserve

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Management_Contingency_Reserve

- The set of budgets reserved for unplanned, but potentially required, changes to project scope and cost. These are the "unknown unknowns", and the project manager must obtain approval before obligating or spending this reserve.

sub-class-of:
pmo-core:Reserve

Class: pmo-core:Management_Process

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Management_Process

- The set of processes that are related to project management directly related activities.

sub-class-of:
pmo-core:Process
in-domain-of:
pmo-core:successorManagementProcesses
pmo-core:predecessorManagementProcesses
in-range-of:
pmo-core:successorManagementProcesses
pmo-core:hasManagementProcesses
pmo-core:predecessorManagementProcesses

Class: pmo-core:Measure

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Measure

- A quantified way to describe a thing.

in-domain-of:
pmo-core:value

Class: pmo-core:Monetary_Worth

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Monetary_Worth

- The monetary value and currency associated to that value.

sub-class-of:
pmo-core:Measure
in-domain-of:
pmo-core:currency
in-range-of:
pmo-core:neededMeasure

Class: pmo-core:Near-Critical_Activity

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Near-Critical_Activity

- A schedule activity that has low total float. The concept of near-critical is equally applicable to a schedule activity or schedule network path. The limit below which total float is considered near critical is subject to expert judgment and varies from project to project.

sub-class-of:
pmo-core:Schedule_Activity

Class: pmo-core:Non-Critical_Activity

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Non-Critical_Activity

- A schedule activity that has high total float or that has not detected as critical activity in the critical path method.

sub-class-of:
pmo-core:Schedule_Activity

Class: pmo-core:Objective

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Objective

- Something toward which work is to be directed, a strategic position to be attained, or a purpose to be achieved, a result to be obtained, a product to be produced, or a service to be performed.

in-domain-of:
pmo-core:description
pmo-core:name
in-range-of:
pmo-core:hasObjectives

Class: pmo-core:Opportunity

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Opportunity`

- A condition or situation favourable to the project, a positive set of circumstances, a positive set of events, a risk that will have a positive impact on project objectives, or a possibility for positive changes.

in-range-of:
pmo-core:hasOpportunities

Class: pmo-core:Performance_Measurement_Baseline

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Performance_Measurement_Baseline`

- An approved plan for the project work against which project execution is compared and deviations are measured for management control. The performance measurement baseline typically integrates scope, scope, schedule, and cost parameters of a project, but may also include technical and quality parameters.

sub-class-of:
pmo-core:Baseline
in-domain-of:
pmo-core:associatedBaselines

Class: pmo-core:Phase

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Phase`

- A collection of logically related project activities, usually culminating in the completion of a major deliverable.

sub-class-of:
pmo-core:Component
in-domain-of:
pmo-core:hasSubphases
pmo-core:WBS_level
in-range-of:
pmo-core:hasPhases
pmo-core:hasSubphases

Class: pmo-core:Portfolio

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Portfolio`

- A collection of projects or programs and other work that are grouped together to facilitate effective management of that work to meet strategic business objectives. The projects or programs of the portfolio may not necessarily be interdependent or directly related.

in-domain-of:
pmo-core:hasPrograms
pmo-core:description
pmo-core:hasObjectives
pmo-core:hasProjects
pmo-core:identifier

Class: pmo-core:Process

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Process`

- A set of interrelated actions and activities performed to achieve a specified set of prodcuts, results, or services.

in-domain-of:
pmo-core:description
pmo-core:name

Class: pmo-core:Process_Group

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Process_Group`

- A collection of logically related processes.

in-domain-of:
pmo-core:description
pmo-core:name

Class: pmo-core:Product

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Product

- An artifact that is produced, is quantifiable, and can be either and end item in itself or a component item.

sub-class-of:
    pmo-core:Deliverable
in-domain-of:
    pmo-core:hasItems
in-range-of:
    pmo-core:hasItems

Class: pmo-core:Program

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Program

- A group of related projects managed in a coordinated way to obtain benefits and control not available from managing them individually. Programs may include elements of related work outside of the scope of the discrete projects in the program.

in-domain-of:
    pmo-core:description
    pmo-core:hasObjectives
    pmo-core:hasProjects
    pmo-core:identifier
in-range-of:
    pmo-core:hasPrograms

Class: pmo-core:Project

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Project

- A temporary endeavor undertaken to create a unique product, service, or result.

in-domain-of:
    pmo-core:hasSubprojects
    pmo-core:hasPerformingOrganization
    pmo-core:hasThreats
    pmo-core:budget_at_completion
    pmo-core:hasApplicationAreas
    pmo-core:hasOpportunities
    pmo-core:cost_performance_index
    pmo-core:data_date
    pmo-core:cost_variance
    pmo-core:budget
    pmo-core:description
    pmo-core:hasBaseline
    pmo-core:hasComponents
    pmo-core:hasObjectives
    pmo-core:identifier
    pmo-core:scope
in-range-of:
    pmo-core:hasSubprojects
    pmo-core:hasProjects

Class: pmo-core:Project_Life_Cycle

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Project_Life_Cycle

- A collection of generally sequential project phases whose name and number are determined by the control needs of the organization or organizations involved in the project.

in-domain-of:
    pmo-core:hasPhases
    pmo-core:description
    pmo-core:name

Class: pmo-core:Project_Management_Knowledge_Area

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Project_Management_Knowledge_Area

- An identified area of project management defined by its knowledge requirements and described in terms of its component processes, practices, inputs, outputs, tools, and techniques.

sub-class-of:
    pmo-core:Knowledge_Area
in-domain-of:

pmo-core:hasManagementProcesses

Class: pmo-core:Project_Management_Process_Group

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Project_Management_Process_Group

- The five process groups required for any project that have clear dependencies and that are required to be performed in the same sequence on each project, independent of the application area or the specifics of the applied project life cycle.

sub-class-of:
pmo-core:Process_Group
in-domain-of:
pmo-core:hasManagementProcesses

Class: pmo-core:Requested_Change

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Requested_Change

- A formally documented change request that is submitted for approval to the integrated change control process.

sub-class-of:
pmo-core:Change_Request
in-domain-of:
pmo-core:submmit_date

Class: pmo-core:Requirement

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Requirement

- A condition or capability that must be met or possessed by a system, product, service, result, or component to satisfy a contract, standard, specification, or other formally imposed documents. Requirements include the quantified and documented need, wants, and expectations of the sponsor, customer, and other stakeholders.

Class: pmo-core:Reserve

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Reserve

- A provision in the project management plan to mitigate cost and/or schedule risk. Often used with a modifier to provide further detail on what types of risk are meant to be mitigated. The specific meaning of the modified term varies by application area.

Class: pmo-core:Resource_Requirement

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Resource_Requirement

- The need of skilled human resource to accomplish a schedule activity.

sub-class-of:
pmo-core:Requirement
in-domain-of:
pmo-core:hasCharacteristics
in-range-of:
pmo-core:hasResourceRequirements

Class: pmo-core:Result

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Result

- An output from performing project management processes and activities. Results include outcomes (e.g. integrated systems, revised process, restructured organization, tests, trained personnel, etc.) and documents (e.g. policies, plans, studies, procedures, specifications, reports, etc.).

sub-class-of:
pmo-core:Deliverable

Class: pmo-core:Schedule

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Schedule

- A detailed plan that includes start and finish dates.

in-domain-of:
pmo-core:remaining_duration
pmo-core:actual_duration
pmo-core:original_duration
pmo-core:schedule_finish_date
pmo-core:actual_start_date
pmo-core:schedule_start_date
in-range-of:

pmo-core:associatedSchedule

Class: pmo-core:Schedule_Activity

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Schedule_Activity

- A discrete scheduled component of work performed during the course of a project. A schedule activity normally has an estimated duration, an estimated cost, and estimated resource requirements. Schedule activities are connected to ther schedule activities or schedule milestones with logical relationships, and are decomposed from work packages.

sub-class-of:
pmo-core:Activity
in-domain-of:
pmo-core:lead
pmo-core:lag
pmo-core:activity_duration
pmo-core:hasCPM
pmo-core:hasResourceRequirements
pmo-core:duration
pmo-core:hasLogicalRelationships
pmo-core:free_float
pmo-core:current_start_date
pmo-core:current_finish_date
pmo-core:critical
pmo-core:target_start_date
pmo-core:actual_finish_date
pmo-core:activity_code
pmo-core:actual_cost
pmo-core:budget
pmo-core:earned_value
pmo-core:hasBaseline
pmo-core:percent_complete
pmo-core:start_imposed_date
in-range-of:
pmo-core:inverse_of_successor_activities
pmo-core:successorActivity
pmo-core:predecessorActivity
pmo-core:actual_cost_of_work_performed

Class: pmo-core:Schedule_Baseline

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Schedule_Baseline

- An approved time phased plan for dates. If the baseline is defined at schedule activity level, the schedule is calculated in base to the scheduled dates.

sub-class-of:
pmo-core:Baseline
in-domain-of:
pmo-core:hasScheduleBaselines
pmo-core:associatedSchedule
in-range-of:
pmo-core:hasScheduleBaselines

Class: pmo-core:Schedule_Milestone

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Schedule_Milestone

- A significant event in the project schedule, such as an event restraining future work or marking the completion of a major deliverable.

sub-class-of:
pmo-core:Activity
in-domain-of:
pmo-core:start_imposed_date

Class: pmo-core:Service

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Service

- Useful work performed that does not produce a tangible product or result, such as performing any of the business functions supporting production or distribution.

sub-class-of:
pmo-core:Deliverable

Class: pmo-core:Subproject

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Subproject

- A smaller portion of the overall project created when a project is subdivided into more manageable components or pieces. Subprojects are usually represented in the work breakdown structure.

sub-class-of:
    pmo-core:Component

Class: pmo-core:Summary_Activity

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Summary_Activity

- A group of related schedule activities aggregated at some summary level, and displayed/reported as a single activity at that summary level.

sub-class-of:
    pmo-core:Activity
in-domain-of:
    pmo-core:hasActivities

Class: pmo-core:Technical_Change_Baseline

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Technical_Change_Baseline

- An time-phased plan for technical changes that must be done in a project.

sub-class-of:
    pmo-core:Baseline

Class: pmo-core:Threat

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Threat

- A condition or situation unfavourable to the project, a negative set of circumstances, a negative set of events, a risk that will have a negative impact on project objective if it occurs, or a possibility for negative changes.

in-range-of:
    pmo-core:hasThreats

Class: pmo-core:Work_Package

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#Work_Package

- A deliverable or project work component at the lowest level of each branch of the work breakdown structure. The work package includes the schedule activities and schedule milestones required to complete the work package deliverable or project work component.

sub-class-of:
    pmo-core:Component
in-domain-of:
    pmo-core:hasActivities

Property: pmo-core:WBS_level

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#WBS_level

- The level in which the phase is located. e.g. 1 at root level, 1.1 at second level, 1.1.1 at third level, and so on.

OWL Type:
    DatatypeProperty
Domain:
    pmo-core:Phase
Range:
    xsd:string

Property: pmo-core:acceptance_criteria

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#acceptance_criteria

- Those criteria, including performance requirements and essential conditions, which must be met before project deliverables are accepted.

OWL Type:
    ObjectProperty
Domain:
    pmo-core:Approved_Change_Request
Range:
    pmo-core:Acceptance_Criteria

Property: pmo-core:accuracy

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#accuracy

- An expression that determines the level under and above the estimate that are allowed.

OWL Type:
DatatypeProperty
Domain:
pmo-core:Estimate
Range:
xsd:string

Property: pmo-core:activity_code

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#activity_code

- One or more numerical or text values that identify characteristics of the work or in some way categorize the schedule activity that allows filtering and ordering of activities within reports.

OWL Type:
DatatypeProperty
Domain:
pmo-core:Schedule_Activity
Range:
xsd:string

Property: pmo-core:activity_duration

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#activity_duration

- The time in calendar units between the start and finish of a schedule activity.

OWL Type:
ObjectProperty
Domain:
pmo-core:Schedule_Activity
Range:
http://www.isi.edu/~pan/damltime/time-entry.owl#DurationDescription

Property: pmo-core:actual_cost

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#actual_cost

- Total costs actually incurred and recorded in accomplishing work performed during a given time period for a schedule activity

OWL Type:
ObjectProperty
Domain:
pmo-core:Schedule_Activity
pmo-core:Component
Range:
pmo-core:Cost

Property: pmo-core:actual_cost_of_work_performed

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#actual_cost_of_work_performed

- Total costs actually incurred and recorded in accomplishing work performed during a given time period for a schedule activity.

OWL Type:
ObjectProperty
Range:
pmo-core:Schedule_Activity

Property: pmo-core:actual_duration

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#actual_duration

- The time in calendar units between the actual start date of the schedule activity and either the data date of the project schedule if the schedule activity is in progress or the actual finish date if the schedule activity is complete.

OWL Type:
ObjectProperty
Domain:
pmo-core:Schedule
Range:
pmo-core:Duration

Property: pmo-core:actual_finish_date

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#actual_finish_date`

- The point in time that work actually ended on a schedule activity.

OWL Type:
    DatatypeProperty
Domain:
    pmo-core:Schedule_Activity
Range:
    xsd:date

Property: pmo-core:actual_start_date

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#actual_start_date`

- The point in time that work actually ended on a schedule activity.

OWL Type:
    DatatypeProperty
Domain:
    pmo-core:Schedule
Range:
    xsd:date

Property: pmo-core:approval_date

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#approval_date`

- Date of approval for the requested change.

OWL Type:
    DatatypeProperty
Domain:
    pmo-core:Approved_Change_Request
Range:
    xsd:date

Property: pmo-core:approval_result

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#approval_result`

- The result for the change request.

OWL Type:
    DatatypeProperty
Domain:
    pmo-core:Approved_Change_Request
Range:
    xsd:string

Property: pmo-core:associatedBaselines

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#associatedBaselines`

- The set of baselines (schedule, cost, scope, technical, quality) that are used for definition of the performance measurement baseline.

OWL Type:
    ObjectProperty
Domain:
    pmo-core:Performance_Measurement_Baseline
Range:
    pmo-core:Baseline

Property: pmo-core:associatedCost

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#associatedCost`

- The cost associated to this cost baseline. It can be either a calculated cost based on its child cost baselines or be itself a cost.

OWL Type:
    ObjectProperty
Domain:
    pmo-core:Cost_Baseline
Range:
    pmo-core:Cost

Property: pmo-core:associatedSchedule

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#associatedSchedule`

- The schedule associated to this schedule baseline. It can be either a calculated schedule based on its child scheudle baselines or be itself a schedule.

OWL Type:
    ObjectProperty
Domain:
    pmo-core:Schedule_Baseline
Range:
    pmo-core:Schedule

Property: pmo-core:baseline_finish_date

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#baseline_finish_date`

- The finish date of a schedule activity in the approved schedule baseline.

OWL Type:
    DatatypeProperty
Domain:
    pmo-core:Baseline
Range:
    xsd:date

Property: pmo-core:baseline_start_date

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#baseline_start_date`

- The start date of a schedule activity in the approved schedule baseline.

OWL Type:
    DatatypeProperty
Domain:
    pmo-core:Baseline
Range:
    xsd:date

Property: pmo-core:budget

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#budget`

- The approved estimate for the project, work breakdown structure component or scheudle activity.

OWL Type:
    ObjectProperty
Domain:
    pmo-core:Project
    pmo-core:Component
    pmo-core:Schedule_Activity
Range:
    pmo-core:Cost_Estimate

Property: pmo-core:budget_at_completion

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#budget_at_completion`

- The sum of all the budget values established for the work to be performed on a project or a work breakdown structure or a schedule activity. The total planned value for the project.

OWL Type:
    ObjectProperty
Domain:
    pmo-core:Project
Range:
    pmo-core:Cost_Estimate

Property: pmo-core:budget_cost

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#budget_cost`

- The monetary value or price of a project activity.

OWL Type:
    ObjectProperty

Range:
pmo-core:Cost

Property: pmo-core:change_request_initiator

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#change_request_initiator

- The person that has submitted the change request.

OWL Type:
ObjectProperty
Domain:
pmo-core:Change_Request
Range:
http://in2gesoft.ehu.es/ont/pmo-organization.owl#Project_Team_Member

Property: pmo-core:cost_performance_index

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#cost_performance_index

- A measure of cost efficiency on a project. It is the ratio of the earned value (EV) to actual costs (AC). CPI = EV / AC. If CPI > 1, then indicates a favourable condition, and else if CPI < 1, then indicates an unfavourable condition.

OWL Type:
DatatypeProperty
Domain:
pmo-core:Project
Range:
xsd:float

Property: pmo-core:cost_type

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#cost_type

- The type of cost using the performing organization vocabulary.

OWL Type:
DatatypeProperty
Domain:
pmo-core:Cost
Range:
xsd:string

Property: pmo-core:cost_variance

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#cost_variance

- A measure of cost performance on a project. It is the algebraic difference betwwen earned value (EV) and actual cost (AC). CV = EV - AC. If CV > 0, then indicates a favourable condition, and else if CPI < 0, then indicates an unfavourable condition.

OWL Type:
DatatypeProperty
Domain:
pmo-core:Project
Range:
xsd:float

Property: pmo-core:critical

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#critical

- Any schedule activity on a critical path in a project schedule. Most commonly determined by using the critical path method. Although some activities are "critical", in the dictionary sense, without being on the critical path, this meaning is seldom used in the project context.

OWL Type:
DatatypeProperty
Domain:
pmo-core:Schedule_Activity
Range:
xsd:boolean

Property: pmo-core:currency

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#currency

- The currency used to measure the estimate.

OWL Type:
ObjectProperty
Domain:
pmo-core:Monetary_Worth
Range:
pmo-core:Currency

Property: pmo-core:current_finish_date

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#current_finish_date

- The current estimate of the point in time when a schedule activity will be completed, where the estimate reflects any reported work progress.

OWL Type:
DatatypeProperty
Domain:
pmo-core:Schedule_Activity
Range:
xsd:date

Property: pmo-core:current_start_date

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#current_start_date

- The current estimate of the point in time when a schedule activity will begin, where the estimate reflects any report progress.

OWL Type:
DatatypeProperty
Domain:
pmo-core:Schedule_Activity
Range:
xsd:date

Property: pmo-core:data_date

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#data_date

- The date up to or throug which the project's reporting system has provided actual status and accomplishments. In some reporting systems, the status information for the data date is included in the plast and in some systems the status information is in the future.

OWL Type:
DatatypeProperty
Domain:
pmo-core:Project
Range:
xsd:date

Property: pmo-core:definition_term

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#definition_term

- The different types that can be defined in that application area. e.g. Product, Type of Customer, Industry Sector.

OWL Type:
DatatypeProperty
Domain:
pmo-core:Application_Area
Range:
xsd:string

Property: pmo-core:description

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#description

- A short phrase or label for each object of this type, usually used with an identifier to differentiate that object from other objects of the same type. The description normaly describes a scope of work.

OWL Type:
DatatypeProperty
Domain:
pmo-core:Portfolio
pmo-core:Project
pmo-core:Program
pmo-core:Objective
pmo-core:Knowledge_Area
pmo-core:Process
pmo-core:Process_Group

pmo-core:Project_Life_Cycle
pmo-core:Component
pmo-core:Application_Area
pmo-core:Change_Request
pmo-core:Logical_Relationship_Type
pmo-core:Cost

Range:
xsd:string

Property: pmo-core:duration

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#duration`

- The total number of work periods (not including holidays or other nonworking periods) required to complete a schedule activity or work breakdown structure component. Usually expressed as workdays or workweeks. Sometimes incorrectly equated with elapsed time.

OWL Type:
ObjectProperty
Domain:
pmo-core:Schedule_Activity
Range:
pmo-core:Duration

Property: pmo-core:duration_unit

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#duration_unit`

- The unit in which is expressed the duration.

OWL Type:
ObjectProperty
Domain:
pmo-core:Duration
Range:
http://www.isi.edu/~pan/damltime/time-entry.owl#TemporalUnit

Property: pmo-core:early_finish_date

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#early_finish_date`

- In the critical path method, the earliest possible point in time on which the uncompleted portions of a schedule activity (or the project) can finish, based on the schedule network logic, the data date, and any schedule constraints. Early finish dates can change as the project progresses and as changes are made to the project management plan.

OWL Type:
DatatypeProperty
Domain:
pmo-core:Critical_Path_Method
Range:
xsd:date

Property: pmo-core:early_start_date

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#early_start_date`

- In the critical path method, the earliest possible point in time on which the uncompleted portions of a schedule activity (or the project) can start, based on the schedule network logic, the data date, and any schedule constraints. Early start dates can change as the project progresses and as changes are made to the project management plan.

OWL Type:
DatatypeProperty
Domain:
pmo-core:Critical_Path_Method
Range:
xsd:date

Property: pmo-core:earned_value

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#earned_value`

- The value of completed work expressed in terms of the approved budget assigned to that work for a schedule activity or work breakdown structure component. Also referred to as the budgeted cost of work performed (BCWP).

OWL Type:
ObjectProperty
Domain:
pmo-core:Schedule_Activity

pmo-core:Component
Range:
pmo-core:Cost_Estimate

Property: pmo-core:effort_unit

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#effort_unit

- The unit in which is measured the effort. e.g. Staff hours, Staff days or Staff weeks.

OWL Type:
ObjectProperty
Domain:
pmo-core:Effort
Range:
http://www.isi.edu/~pan/damltime/time-entry.owl#TemporalUnit

Property: pmo-core:estimate

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#estimate

- Definicion de estimacion

OWL Type:
ObjectProperty
Domain:
pmo-core:Estimate
Range:
pmo-core:Cost
pmo-core:Effort
pmo-core:Duration

Property: pmo-core:estimation_type

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#estimation_type

- The type of estimate that is being carried out. E.g. Preliminary, Conceptual, Feasibility, Order-of-Magnitude, Definitive.

OWL Type:
DatatypeProperty
Domain:
pmo-core:Estimate
Range:
xsd:string

Property: pmo-core:free_float

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#free_float

- The amount of time that a schedule activity can be delayed without delaying the early start of any inmediately following schedule activities.

OWL Type:
ObjectProperty
Domain:
pmo-core:Schedule_Activity
Range:
http://www.isi.edu/~pan/damltime/time-entry.owl#DurationDescription

Property: pmo-core:hasAcceptanceCriteria

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#hasAcceptanceCriteria

- Those criteria, including performance requirements and essential conditions, which must be met before project deliverables are accepted.

OWL Type:
ObjectProperty
Domain:
pmo-core:Deliverable
Range:
pmo-core:Criteria

Property: pmo-core:hasActivities

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#hasActivities

- The set of activities in which is divided a Work Package.

OWL Type:
    ObjectProperty
Domain:
    pmo-core:Work_Package
    pmo-core:Summary_Activity
Range:
    pmo-core:Activity

Property: pmo-core:hasApplicationAreas

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#hasApplicationAreas`

- The set of applications areas that are covered by the project.

OWL Type:
    ObjectProperty
Domain:
    pmo-core:Project
Range:
    pmo-core:Application_Area

Property: pmo-core:hasBaseline

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#hasBaseline`

- The set of baselines that are defined at project, work component or schedule activity level.

OWL Type:
    ObjectProperty
Domain:
    pmo-core:Schedule_Activity
    pmo-core:Project
    pmo-core:Component
Range:
    pmo-core:Baseline

Property: pmo-core:hasCPM

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#hasCPM`

- The critical path method values used for calculating the critical path in the network logic.

OWL Type:
    ObjectProperty
Domain:
    pmo-core:Schedule_Activity
Range:
    pmo-core:Critical_Path_Method

Property: pmo-core:hasCharacteristics

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#hasCharacteristics`

- The minimum characteristics that the resource must met.

OWL Type:
    ObjectProperty
Domain:
    pmo-core:Resource_Requirement

Property: pmo-core:hasComponents

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#hasComponents`

- The set of constituent phases that define the project.

OWL Type:
    ObjectProperty
Domain:
    pmo-core:Project
    pmo-core:Component
Range:
    pmo-core:Component

Property: pmo-core:hasCostBaselines

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#hasCostBaselines`

- The component cost baseline that are part of this Cost Baseline.

OWL Type:
    ObjectProperty
Domain:
    pmo-core:Cost_Baseline
Range:
    pmo-core:Cost_Baseline

Property: pmo-core:hasDefects

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#hasDefects`

- The set of defects that are present in verification of deliverable.

OWL Type:
    ObjectProperty
Domain:
    pmo-core:Deliverable
Range:
    pmo-core:Defect

Property: pmo-core:hasEffort

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#hasEffort`

- The effort used for completion of this component.

OWL Type:
    ObjectProperty
Domain:
    pmo-core:Component
Range:
    pmo-core:Effort

Property: pmo-core:hasItems

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#hasItems`

- The set of component items in which is composed the product.

OWL Type:
    ObjectProperty
Domain:
    pmo-core:Product
Range:
    pmo-core:Product

Property: pmo-core:hasLogicalRelationships

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#hasLogicalRelationships`

- The set of logical relationships that are constrained in the schedule activity.

OWL Type:
    ObjectProperty
Domain:
    pmo-core:Schedule_Activity
Range:
    pmo-core:Logical_Relationship

Property: pmo-core:hasManagementProcesses

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#hasManagementProcesses`

- The set of processes that are accomplished inside this knowledge area.

OWL Type:
    ObjectProperty
Domain:
    pmo-core:Project_Management_Knowledge_Area
    pmo-core:Project_Management_Process_Group
Range:
    pmo-core:Management_Process

Property: pmo-core:hasObjectives

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#hasObjectives

- The collection of objectives that want to be attained.

OWL Type:
    ObjectProperty
Domain:
    pmo-core:Portfolio
    pmo-core:Project
    pmo-core:Program
Range:
    pmo-core:Objective

Property: pmo-core:hasOpportunities

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#hasOpportunities

- The set of opportunities that are taked into account in the project.

OWL Type:
    ObjectProperty
Domain:
    pmo-core:Project
Range:
    pmo-core:Opportunity

Property: pmo-core:hasPerformingOrganization

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#hasPerformingOrganization

- The organization whose personnel are most directly involved in doing the work of project.

OWL Type:
    ObjectProperty
Domain:
    pmo-core:Project
Range:
    http://in2gesoft.ehu.es/ont/pmo-organization.owl#Organization

Property: pmo-core:hasPhases

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#hasPhases

- The set of phases needed to accomplish the life cycle.

OWL Type:
    ObjectProperty
Domain:
    pmo-core:Project_Life_Cycle
Range:
    pmo-core:Phase

Property: pmo-core:hasPrograms

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#hasPrograms

- The collection of programs that are part of a Portfolio

OWL Type:
    ObjectProperty
Domain:
    pmo-core:Portfolio
Range:
    pmo-core:Program

Property: pmo-core:hasProjects

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#hasProjects

- The collection of projects that are joined in the program and/or in the portfolio.

OWL Type:
    ObjectProperty
Domain:
    pmo-core:Portfolio
    pmo-core:Program
Range:

pmo-core:Project

Property: pmo-core:hasRelationshipType

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#hasRelationshipType`

- The type of relation between the predecessor and successor activities.

OWL Type:
ObjectProperty
Domain:
pmo-core:Logical_Relationship
Range:
pmo-core:Logical_Relationship_Type

Property: pmo-core:hasResourceRequirements

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#hasResourceRequirements`

- The resource requirements that are needed to accomplish the schedule activity. The resource can be material, or human.

OWL Type:
ObjectProperty
Domain:
pmo-core:Schedule_Activity
Range:
pmo-core:Resource_Requirement

Property: pmo-core:hasScheduleBaselines

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#hasScheduleBaselines`

- The component cost baseline that are part of this Cost Baseline.

OWL Type:
ObjectProperty
Domain:
pmo-core:Schedule_Baseline
Range:
pmo-core:Schedule_Baseline

Property: pmo-core:hasSubphases

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#hasSubphases`

- The set of subphases in which is divided the current phase.

OWL Type:
ObjectProperty
Domain:
pmo-core:Phase
Range:
pmo-core:Phase

Property: pmo-core:hasSubprojects

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#hasSubprojects`

- A collection of subprojects, that is subdivided from the project to provide more manegeable components or pieces. A subproject can be referred to as a project, managed as a project, and acquired from a seller.

OWL Type:
ObjectProperty
Domain:
pmo-core:Project
Range:
pmo-core:Project

Property: pmo-core:hasThreats

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#hasThreats`

- The set of threats taked into account in the project.

OWL Type:
ObjectProperty
Domain:

pmo-core:Project
Range:
pmo-core:Threat

Property: pmo-core:identifier

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#identifier`

- A short unique numeric or text identification assigned to this object to differentiate from other objects of the same object.

OWL Type:
DatatypeProperty
Domain:
pmo-core:Portfolio
pmo-core:Program
pmo-core:Project
pmo-core:Activity
pmo-core:Change_Request
Range:
xsd:string

Property: pmo-core:inverse_of_successor_activities

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#inverse_of_successor_activities`

-

OWL Type:
ObjectProperty
Range:
pmo-core:Schedule_Activity

Property: pmo-core:lag

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#lag`

- A modification of a logical relationship that directs a delay in the successor activity.

OWL Type:
ObjectProperty
Domain:
pmo-core:Schedule_Activity
Range:
pmo-core:Duration

Property: pmo-core:late_finish_date

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#late_finish_date`

- In the critical path method, the latest possible point in time on that a schedule activity (or the project) may be completed based on the schedule network logic, the project completion date, and any constraints assigned to the schedule activities without violating a schedule constraint or delaying the project completion date. The late finish dates are determined during the backward pass calculation of the project schedule network.

OWL Type:
DatatypeProperty
Domain:
pmo-core:Critical_Path_Method
Range:
xsd:date

Property: pmo-core:late_start_date

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#late_start_date`

- In the critical path method, the latest possible point in time on that a schedule activity (or the project) may begin based on the schedule network logic, the project completion date, and any constraints assigned to the schedule activities without violating a schedule constraint or delaying the project completion date. The late start dates are determined during the backward pass calculation of the project schedule network.

OWL Type:
DatatypeProperty
Domain:
pmo-core:Critical_Path_Method
Range:
xsd:date

Property: pmo-core:lead

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#lead

- A modification of a logical relationship that allows an acceleration of the successor activity.

OWL Type:
    ObjectProperty
Domain:
    pmo-core:Schedule_Activity
Range:
    pmo-core:Duration

Property: pmo-core:name

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#name

- A descriptive phrase that fully defines the object.

OWL Type:
    DatatypeProperty
Domain:
    pmo-core:Knowledge_Area
    pmo-core:Objective
    pmo-core:Process
    pmo-core:Process_Group
    pmo-core:Project_Life_Cycle
    pmo-core:Component
    pmo-core:Application_Area
    pmo-core:Logical_Relationship_Type
    pmo-core:Currency
Range:
    xsd:string

Property: pmo-core:neededMeasure

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#neededMeasure

- The needed measure to describe the cost.

OWL Type:
    ObjectProperty
Domain:
    pmo-core:Cost
Range:
    pmo-core:Effort
    pmo-core:Monetary_Worth

Property: pmo-core:original_duration

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#original_duration

- The activity duration originally assigned to a schedule activity and not updated as progress is reported on the activity.

OWL Type:
    ObjectProperty
Domain:
    pmo-core:Schedule
Range:
    pmo-core:Duration

Property: pmo-core:outline

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#outline

- A brief description of what the activity must perform.

OWL Type:
    DatatypeProperty
Domain:
    pmo-core:Activity
Range:
    xsd:string

Property: pmo-core:percent_complete

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#percent_complete

- An estimate, expressed as a percent, of the amount of work that has been completed on an activity or a work breakdown structure component.

OWL Type:
DatatypeProperty
Domain:
pmo-core:Schedule_Activity
pmo-core:Component
Range:
xsd:string

Property: pmo-core:planned_value

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#planned_value

- The authorized budget assigned to the scheduled work to be accomplished for a schedule activity or work breakdown structure component.

OWL Type:
ObjectProperty
Domain:
pmo-core:Component
Range:
pmo-core:Cost_Estimate

Property: pmo-core:predecessorActivity

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#predecessorActivity

- The schedule activity that determines when the logical successor activity can begin.

OWL Type:
ObjectProperty
Domain:
pmo-core:Logical_Relationship
Range:
pmo-core:Schedule_Activity

Property: pmo-core:predecessorManagementProcesses

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#predecessorManagementProcesses

- The set of management processes that are logically related backwards to this management process.

Inverse:
pmo-core:successorManagementProcesses
OWL Type:
ObjectProperty
Domain:
pmo-core:Management_Process
Range:
pmo-core:Management_Process

Property: pmo-core:remaining_duration

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#remaining_duration

- The time in calendar units, between the data date of the project schedule and the finish date of a schedule activity that has an actual start date. This represents the time needed to complete a schedule activity where the work is in progress.

OWL Type:
ObjectProperty
Domain:
pmo-core:Schedule
Range:
pmo-core:Duration

Property: pmo-core:request_date

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#request_date

- Date of approval for the request change.

OWL Type:
DatatypeProperty
Domain:
pmo-core:Change_Request
Range:
xsd:date

Property: pmo-core:reserve

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#reserve

- Provision in the project management plan to mitigate cost and/or schedule risk. Often used with a modifier to provide further detail on what types of risk are meant to be mitigated. The specific meaning of the modified term varies by application area.

OWL Type:
ObjectProperty
Domain:
pmo-core:Cost_Estimate
Range:
pmo-core:Cost

Property: pmo-core:schedule_finish_date

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#schedule_finish_date

- The point in time that work was scheduled to finish on a schedule activity. The scheduled finish date is normally within the range of dates delimited by the early finish date and teh late finish date. It may reflect resource leveling of scarce resources. Sometimes called planned finish date.

OWL Type:
DatatypeProperty
Domain:
pmo-core:Schedule
Range:
xsd:date

Property: pmo-core:schedule_start_date

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#schedule_start_date

- The point in time that work was scheduled to start on a schedule activity. The scheduled start date is normally within the range of dates delimited by the early start date and teh late start date. It may reflect resource leveling of scarce resources. Sometimes called planned start date.

OWL Type:
DatatypeProperty
Domain:
pmo-core:Schedule
Range:
xsd:date

Property: pmo-core:scope

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#scope

- The work that must be performed to deliver a product, service, or resutl with the specified features and functions.

OWL Type:
DatatypeProperty
Domain:
pmo-core:Project
pmo-core:Component
pmo-core:Change_Request
Range:
xsd:string

Property: pmo-core:start_imposed_date

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#start_imposed_date

- A fixed start date imposed on a schedule activity or schedule milestone in the form of a "start not earlier than".

OWL Type:
DatatypeProperty
Domain:
pmo-core:Schedule_Activity
pmo-core:Schedule_Milestone
Range:
xsd:date

Property: pmo-core:submmit_date

URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#submmit_date

- Date of submission for the requested change.

OWL Type:
    DatatypeProperty
Domain:
    pmo-core:Requested_Change
Range:
    xsd:date

Property: pmo-core:successorActivity

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#successorActivity`

- The schedule activity that follows a predecessor activity, as determined by their logical relationship.

OWL Type:
    ObjectProperty
Domain:
    pmo-core:Logical_Relationship
Range:
    pmo-core:Schedule_Activity

Property: pmo-core:successorManagementProcesses

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#successorManagementProcesses`

- The set of management processes logically related to the current process.

Inverse:
    pmo-core:predecessorManagementProcesses
OWL Type:
    ObjectProperty
Domain:
    pmo-core:Management_Process
Range:
    pmo-core:Management_Process

Property: pmo-core:symbol

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#symbol`

- The symbol or set of characters used commonly to express the currency.

OWL Type:
    DatatypeProperty
Domain:
    pmo-core:Currency
Range:
    xsd:string

Property: pmo-core:target_start_date

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#target_start_date`

- The date that work is planned (targeted) to start on a schedule activity.

OWL Type:
    DatatypeProperty
Domain:
    pmo-core:Schedule_Activity
Range:
    xsd:date

Property: pmo-core:value

`URI: http://in2gesoft.ehu.es/ont/pmo-core.owl#value`

- A numeric value that determined jointly with the other parameter of this object

OWL Type:
    DatatypeProperty
Domain:
    pmo-core:Measure
Range:
    xsd:float

Anexo B

Project Management Ontology – Organization ontology

Project Management Ontology (PMO) es una ontología de dominio que define un modelo formal de los procesos, actividades, herramientas y técnicas específicas de la gestión de proyectos. PMO proporciona una descripción completa de los términos fundamentales y características inherentes al manejo de la información asociada a la gestión, seguimiento, control y dirección de los proyectos, así como de los procesos, relaciones, restricciones y aserciones sobre los datos de proyectos.

PMO-Organization proporciona la estructura organizativa de proyecto, definiendo los conceptos de equipo, persona, atribuciones, habilidades, asignaciones, etc. El enfoque tomado para el desarrollo de esta ontología ha sido la división desde el punto de vista de la gestión para la organización que desarrolla el proyecto, el desarrollo de los equipos, y los actores que forman parte de dicho proyecto. En la ontología se han definido los conceptos básicos de una organización, como estructura organizativa, persona, puesto o habilidades, y se ha extendido añadiendo conceptos más relacionados a la gestión del proyectos, como equipo de proyecto, miembro de equipo, gestor o equipo virtual. Finalmente se han redefinido o agregado nuevos términos conducentes a su adaptación a las actividades de gestión, como por ejemplo, habilidades de gestión, responsabilidades, competencias o comunicaciones.

PMO-Organization

PROJECT MANAGEMENT ONTOLOGY ORGANIZATION

Thesis Version — 09 May 2011

This version:
http://in2gesoft.ehu.es/ont/pmo-organization.owl
Last Update:
Date: 09 May 2011
Authors:
Fran Ruiz, Tecnalia
Contributors:
Daniel Rodrígez, Universidad de Alcalá de Henares
Javier Dolado, Universidad del País Vasco

Abstract

This specification defines the Project Management Organization Ontology, providing the main concepts related with project management organization and roles.

Status of this Document

This is a work developed for the PhD Thesis of Francisco Javier Ruiz Bertol. Inherently, all ontologies are in constant evolution. This specification meets the adequate requirements needed to be presented for this PhD Thesis, in the sense that the mapping with Project Management Body of Knowledge processes are captured in the ontology derived from this specification. If there is some suggestion or improvement that can be made for the ontology, please notify to the author (Fran.Ruiz@tecnalia.com), or contact with the supervisors of the thesis, Daniel Rodríguez (daniel.rodriguezg@uah.es) or Javier Dolado (javier.dolado@ehu.es). Thank you.

B.1 Introduction

Project Management Ontology (PMO), a set of ontologies that describe the generic knowledge about the project management. This ontology provides the first formal knowledge storage in project management domain. It is constructed in a modular and structured manner, so domain-specific ontologies for a concrete domain can be joined to PMO to create a specific ontology for project management in that domain. This can be done using ontology mapping or ontology merging, where common concepts can be compared between the ontologies. PMO includes:

- A basic taxonomy about projects common to all subject areas describing the skeleton of key terms needed to manage a project. This taxonomy includes relations of the type `isA` (subclasses) or `has` (composition) to define the parts in which is divided a project or a project component.
- A full vocabulary directly related to project management, but applicable or extensible to most of the subject areas. This vocabulary has been directly obtained from the knowledge appeared in the PMBOK.
- A knowledge base system (KBS) containing the expertise knowledge about how to manage adequately a project. This includes the structure, semantic content, and also individuals, all of them expressing guidance for correctly manage the project.
- A set of slot or properties applicable to every concept defined in the taxonomy and the vocabulary.
- The relationships among concepts defined in the taxonomy. These relationships are based on the knowledge used by the project managers.

B.1.1 Terminology and Notation

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

Namespace URIs of the general form "http://www.example.com/" represent some application-dependent or context-dependent

URI as defined in RFC 2396.

The XML Namespace URIs that MUST be used by implementations of this specification are:

- http://ehu.es/pmo/pmo-core# - PMO-Core Ontology Namespace
- http://ehu.es/pmo/pmo-org# - PMO-Organization Ontology Namespace
- http://ehu.es/pmo/pmo-plan# - PMO-Planning Ontology Namespace
- http://ehu.es/pmo/pmo-process# - PMO-Process Ontology Namespace
- http://ehu.es/pmo/pmo-cost# - PMO-Cost Ontology Namespace

B.2. PMO Organization ontology overview

The PMO Organization definitions presented here are written using a computer language (RDF/OWL) that makes it easy for software to process some basic facts about the terms in the PMO Organization Ontology, and consequently about the things described in this PhD Thesis.

B.2.1. Example

Here is a very basic document describing people and roles responsibility:

```
<organization:Project_Team_Member rdf:ID="JohnDoe">
  <organization:person_birthdate rdf:datatype="&xsd;date">1970-05-04</organization:person_birthdate>
  <organization:has_Role>
    <organization:Individual_Role rdf:ID="JavaDeveloper"/>
  </organization:has_Role>
  <organization:has_Responsibility>
    <organization:Management_Responsibility rdf:ID="Development_Responsibility"/>
  </organization:has_Responsibility>
</organization:Project_Team_Member>
<organization:Project_Team_Member rdf:ID="JaneDoe"/>
<organization:Group_Role rdf:ID="Development"/>
<organization:Individual_Role rdf:ID="PHPDeveloper"/>
<rdf:Description rdf:ID="Group_Role_5">
  <rdfs:comment rdf:datatype="&xsd;string"></rdfs:comment>
</rdf:Description>
<organization:Organization_Role rdf:ID="MoneyMakingMachine"/>
<organization:Organization_Role rdf:ID="NGO"/>
<organization:Management_Responsibility rdf:ID="Management_Responsibility"/>
```

B.3. Cross-reference for PMO Organization classes and properties

Class: pmo-org:Business

`URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#Business`

- A commercial or industrial enterprise and the people who constitute it.

sub-class-of:
pmo-org:Enterprise

Class: pmo-org:Company

`URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#Company`

- An institution created to conduct business.

sub-class-of:
pmo-org:Enterprise

Class: pmo-org:Corporation

`URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#Corporation`

- A business firm whose articles of incorporation have been approved in some state

sub-class-of:
pmo-org:Enterprise

Class: pmo-org:Enterprise

`URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#Enterprise`

- An organization created for business ventures.

sub-class-of:
pmo-org:Organization

Class: pmo-org:Firm

`URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#Firm`

- The members of a business organization that owns or operates one or more establishments.

sub-class-of:
pmo-org:Enterprise

Class: pmo-org:Govermental_Agency

`URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#Govermental_Agency`

- An administrative unit of government.

sub-class-of:
pmo-org:Organization

Class: pmo-org:Group_Role

`URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#Group_Role`

- A defined function to be performed by a group, such as managing, controlling, or development.

sub-class-of:
pmo-org:Role

Class: pmo-org:Individual_Role

`URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#Individual_Role`

- A defined function to be performed by a project team member, such as testing, filing, inspecting, coding.

sub-class-of:
pmo-org:Role
in-domain-of:
pmo-org:associated_Activities

Class: pmo-org:Join

`URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#Join`

- The act of incorporation someone to a project team.

in-domain-of:
pmo-org:staff_who_joins
pmo-org:playing_Role
pmo-org:starting_date
pmo-org:joins_to_project_team
pmo-org:will_adquire_Responsibility
in-range-of:
pmo-org:incorporations

Class: pmo-org:Management_Responsibility

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#Management_Responsibility

- The social force related to management issues that binds to the courses of action demanded by that force.

sub-class-of:
    pmo-org:Responsibility
in-domain-of:
    pmo-org:has_associated_Management_Activities

Class: pmo-org:Organization

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#Organization

- A group of people organized for some purpose or to perform some type of work within an organization.

in-domain-of:
    pmo-org:manages_Project_Team
    pmo-org:performsProject
    pmo-org:has
    pmo-org:name
in-range-of:
    pmo-org:belongs_To
    pmo-org:is_hired_by_Organization
    pmo-org:is_Managed_by_Organization

Class: pmo-org:Organization_Role

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#Organization_Role

- A defined function to be performed by an organization.

sub-class-of:
    pmo-org:Role

Class: pmo-org:Organizational_Unit

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#Organizational_Unit

- The logical unit in which is divided the organization.

in-domain-of:
    pmo-org:has_Assigned_Staff
    pmo-org:plays_Role
    pmo-org:belongs_To
    pmo-org:has_Manager
    pmo-org:participates
    pmo-org:is_divided_on
    pmo-org:name
in-range-of:
    pmo-org:is_Assigned_to_Organizational_Unit
    pmo-org:has
    pmo-org:is_divided_on

Class: pmo-org:Partnership

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#Partnership

- The members of a business venture created by contract.

sub-class-of:
    pmo-org:Enterprise

Class: pmo-org:Person

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#Person

- Personnel who assist their superior in carrying out an assigned task.

in-domain-of:
- pmo-org:person_birthdate
- pmo-org:person_surname
- pmo-org:person_name

Class: pmo-org:Project_Management_Team

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#Project_Management_Team

- The members of the project team who are directly involved inproject management activities.

sub-class-of:
- pmo-org:Project_Team

Class: pmo-org:Project_Sponsor

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#Project_Sponsor

- The person or group that provides the financial resources, in cash or in kind, for the project.

sub-class-of:
- pmo-org:Project_Team

Class: pmo-org:Project_Team

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#Project_Team

- The coordinated group that is performing the work of the project.

in-domain-of:
- pmo-org:has_Chief_Executive
- pmo-org:incorporations
- pmo-org:is_Comprised_of_Project_Team_Members
- pmo-org:releases
- pmo-org:is_Assigned_To_Project
- pmo-org:is_Managed_by_Organization
- pmo-org:name

in-range-of:
- pmo-org:is_Associated_to_Project_Team
- pmo-org:manages_Project_Team

Class: pmo-org:Project_Team_Member

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#Project_Team_Member

- The person who reports either directly or indirectly to the project manager, and who is responsible for performing project work as a regular part of their assigned duties.

sub-class-of:
- pmo-org:Staff_Member

in-domain-of:
- pmo-org:is_Associated_to_Project_Team
- pmo-org:has_Responsibility
- pmo-org:manages_Organizational_Unit
- pmo-org:is_Enganged_In_Project
- pmo-org:has_Role

in-range-of:
- pmo-org:is_Comprised_of_Project_Team_Members

Class: pmo-org:Release

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#Release

- The act of freeing someone of a project team.

in-domain-of:
    pmo-org:staff_who_releases
    pmo-org:release_Date
    pmo-org:releases_from_Project_Team
in-range-of:
    pmo-org:releases

Class: pmo-org:Responsibility

`URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#Responsibility`

- The social force that binds to the courses of action demanded by that force.

in-domain-of:
    pmo-org:responsibility_name
    pmo-org:responsibility_description

Class: pmo-org:Role

`URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#Role`

- A defined function to be performed by a project team member, group or organization.

in-domain-of:
    pmo-org:role_description
    pmo-org:role_name

Class: pmo-org:Staff_Member

`URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#Staff_Member`

- A person who is hired by a corporation to perform a job.

sub-class-of:
    pmo-org:Person
in-domain-of:
    pmo-org:is_Assigned_to_Organizational_Unit
    pmo-org:is_hired_by_Organization
    pmo-org:has_Calendar
    pmo-org:occupies_Position
    pmo-org:idCardNumber

Class: pmo-org:Virtual_Team

`URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#Virtual_Team`

- A group of persons with a shared objective who fulfill their roles with little or no time spent meeting face to face. Various forms of technology are often used to facilitate communication among team members. Virtual teams can be comprised of persons separated by great distances.

sub-class-of:
    pmo-org:Project_Team

Property: pmo-org:DatatypeProperty_11

`URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#DatatypeProperty_11`

-

OWL Type:
    DatatypeProperty

Property: pmo-org:DatatypeProperty_3

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#DatatypeProperty_3

-

OWL Type:
DatatypeProperty

Property: pmo-org:ObjectProperty_44

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#ObjectProperty_44

-

OWL Type:
ObjectProperty

Property: pmo-org:associated_Activities

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#associated_Activities

- The activities directly associated to the corresponding role.

OWL Type:
ObjectProperty
Domain:
pmo-org:Individual_Role

Property: pmo-org:belongs_To

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#belongs_To

- The belonging to Organizational Units.

Inverse:
pmo-org:has
OWL Type:
ObjectProperty
Domain:
pmo-org:Organizational_Unit
Range:
pmo-org:Organization

Property: pmo-org:has

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#has

- The set of Organizational Units in which is divided the Organization.

Inverse:
pmo-org:belongs_To
OWL Type:
ObjectProperty
Domain:
pmo-org:Organization
Range:
pmo-org:Organizational_Unit

Property: pmo-org:has_Assigned_Staff

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#has_Assigned_Staff

- The assigned staff to this entity.

OWL Type:
ObjectProperty
Domain:
pmo-org:Organizational_Unit

Property: pmo-org:has_Calendar

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#has_Calendar

- The calendar that will use the person during the stance in the current Organization.

OWL Type:
    ObjectProperty
Domain:
    pmo-org:Staff_Member

Property: pmo-org:has_Chief_Executive

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#has_Chief_Executive

- The Chief Executive of the Project Team

OWL Type:
    ObjectProperty
Domain:
    pmo-org:Project_Team

Property: pmo-org:has_Manager

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#has_Manager

- The manager that manages the Organizational Unit.

OWL Type:
    ObjectProperty
Domain:
    pmo-org:Organizational_Unit

Property: pmo-org:has_Responsibility

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#has_Responsibility

-

OWL Type:
    ObjectProperty
Domain:
    pmo-org:Project_Team_Member

Property: pmo-org:has_Role

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#has_Role

-

OWL Type:
    ObjectProperty
Domain:
    pmo-org:Project_Team_Member

Property: pmo-org:has_associated_Management_Activities

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#has_associated_Management_Activities

- The set of associated activities directly related with the responsibility.

OWL Type:
    ObjectProperty
Domain:
    pmo-org:Management_Responsibility

Property: pmo-org:idCardNumber

`URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#idCardNumber`

- The internal identification for the person in the corporation.

OWL Type:
    DatatypeProperty
Domain:
    pmo-org:Staff_Member
Range:
    xsd:string

Property: pmo-org:incorporations

`URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#incorporations`

-

OWL Type:
    ObjectProperty
Domain:
    pmo-org:Project_Team
Range:
    pmo-org:Join

Property: pmo-org:is_Assigned_To_Project

`URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#is_Assigned_To_Project`

- The set of Projects that are managed by the project team.

OWL Type:
    ObjectProperty
Domain:
    pmo-org:Project_Team

Property: pmo-org:is_Assigned_to_Organizational_Unit

`URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#is_Assigned_to_Organizational_Unit`

-

OWL Type:
    ObjectProperty
Domain:
    pmo-org:Staff_Member
Range:
    pmo-org:Organizational_Unit

Property: pmo-org:is_Associated_to_Project_Team

`URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#is_Associated_to_Project_Team`

- The project team which belongs this person.

Inverse:
    pmo-org:is_Comprised_of_Project_Team_Members
OWL Type:
    ObjectProperty
Domain:
    pmo-org:Project_Team_Member
Range:
    pmo-org:Project_Team

Property: pmo-org:is_Comprised_of_Project_Team_Members

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#is_Comprised_of_Project_Team_Members

- The set of project team members.

Inverse:
pmo-org:is_Associated_to_Project_Team
OWL Type:
ObjectProperty
Domain:
pmo-org:Project_Team
Range:
pmo-org:Project_Team_Member

Property: pmo-org:is_Enganged_In_Project

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#is_Enganged_In_Project

- The set of Projects where this member is engaged in.

OWL Type:
ObjectProperty
Domain:
pmo-org:Project_Team_Member

Property: pmo-org:is_Managed_by_Organization

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#is_Managed_by_Organization

- The organization that manages the Project Team.

Inverse:
pmo-org:manages_Project_Team
OWL Type:
ObjectProperty
Domain:
pmo-org:Project_Team
Range:
pmo-org:Organization

Property: pmo-org:is_divided_on

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#is_divided_on

- The hierarchical units in what is divided an Organizational Unit.

OWL Type:
ObjectProperty
Domain:
pmo-org:Organizational_Unit
Range:
pmo-org:Organizational_Unit

Property: pmo-org:is_hired_by_Organization

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#is_hired_by_Organization

- The organization that hires the person.

OWL Type:
ObjectProperty
Domain:
pmo-org:Staff_Member
Range:
pmo-org:Organization

Property: pmo-org:joins_to_project_team

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#joins_to_project_team

-

OWL Type:
ObjectProperty
Domain:
pmo-org:Join

Property: pmo-org:manages_Organizational_Unit

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#manages_Organizational_Unit

-

OWL Type:
ObjectProperty
Domain:
pmo-org:Project_Team_Member

Property: pmo-org:manages_Project_Team

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#manages_Project_Team

- The set of Project Teams that are directly managed by the Organization.

Inverse:
pmo-org:is_Managed_by_Organization
OWL Type:
ObjectProperty
Domain:
pmo-org:Organization
Range:
pmo-org:Project_Team

Property: pmo-org:name

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#name

- The name of the Organization

OWL Type:
DatatypeProperty
Domain:
pmo-org:Organization
pmo-org:Organizational_Unit
pmo-org:Project_Team
Range:
xsd:string

Property: pmo-org:occupies_Position

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#occupies_Position

-

OWL Type:
ObjectProperty
Domain:
pmo-org:Staff_Member

Property: pmo-org:participates

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#participates

- The set of people that participates in this Organizational Unit.

OWL Type:
    ObjectProperty
Domain:
    pmo-org:Organizational_Unit

Property: pmo-org:performsProject

`URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#performsProject`

- The projects that are peformed by the organization.

OWL Type:
    ObjectProperty
Domain:
    pmo-org:Organization

Property: pmo-org:person_birthdate

`URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#person_birthdate`

- The date of birth for the person.

OWL Type:
    DatatypeProperty
Domain:
    pmo-org:Person
Range:
    xsd:date

Property: pmo-org:person_name

`URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#person_name`

- The family name of the person.

OWL Type:
    DatatypeProperty
Domain:
    pmo-org:Person
Range:
    xsd:string

Property: pmo-org:person_surname

`URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#person_surname`

- The surname of the person.

OWL Type:
    DatatypeProperty
Domain:
    pmo-org:Person
Range:
    xsd:string

Property: pmo-org:playing_Role

`URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#playing_Role`

-

OWL Type:
    ObjectProperty
Domain:
    pmo-org:Join

Property: pmo-org:plays_Role

`URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#plays_Role`

- The Role that is playing the current Organizational Unit.

OWL Type:
    ObjectProperty
Domain:
    pmo-org:Organizational_Unit

Property: pmo-org:release_Date

`URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#release_Date`

-

OWL Type:
    ObjectProperty
Domain:
    pmo-org:Release

Property: pmo-org:releases

`URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#releases`

-

OWL Type:
    ObjectProperty
Domain:
    pmo-org:Project_Team
Range:
    pmo-org:Release

Property: pmo-org:releases_from_Project_Team

`URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#releases_from_Project_Team`

-

OWL Type:
    ObjectProperty
Domain:
    pmo-org:Release

Property: pmo-org:responsibility_description

`URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#responsibility_description`

- A full description about the responsibility defined here.

OWL Type:
    DatatypeProperty
Domain:
    pmo-org:Responsibility
Range:
    xsd:string

Property: pmo-org:responsibility_name

`URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#responsibility_name`

- A descriptive name for the responsibility.

OWL Type:
    DatatypeProperty

Domain:
pmo-org:Responsibility

Property: pmo-org:role_description

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#role_description

- A brief description about the role.

OWL Type:
DatatypeProperty
Domain:
pmo-org:Role
Range:
xsd:string

Property: pmo-org:role_name

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#role_name

- The name of the role.

OWL Type:
DatatypeProperty
Domain:
pmo-org:Role
Range:
xsd:string

Property: pmo-org:staff_who_joins

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#staff_who_joins

-

OWL Type:
ObjectProperty
Domain:
pmo-org:Join

Property: pmo-org:staff_who_releases

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#staff_who_releases

-

OWL Type:
ObjectProperty
Domain:
pmo-org:Release

Property: pmo-org:starting_date

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#starting_date

-

OWL Type:
ObjectProperty
Domain:
pmo-org:Join

Property: pmo-org:will_adquire_Responsibility

URI: http://in2gesoft.ehu.es/ont/pmo-organization.owl#will_adquire_Responsibility

-

OWL Type:
    ObjectProperty
Domain:
    pmo-org:Join

Anexo C

Reglas SWRL para PMO

En este anexo extiende la ontología núcleo de PMO (*PMO-Core*), aplicando una serie de reglas en SWRL. La base de estas reglas están definidas *Project Management Ontology*, determinando las entidades dependientes de la estructura de la organización que se presentan en el capítulo 6.

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
  <!ENTITY pmo "http://in2gesoft.ehu.es/ont/activity061122.owl#" >
  <!ENTITY org "http://in2gesoft.ehu.es/ont/organization061123.owl#"
   >
  <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
  <!ENTITY time "http://www.isi.edu/~pan/damltime/time-entry.owl#" >
  <!ENTITY sqwrl "http://sqwrl.stanford.edu/ontologies/built-ins/3.4/
    sqwrl.owl#">
  <!ENTITY prj1 "http://in2gesoft.ehu.es/ont/ProjectInstance.owl#">
]>

<rdf:RDF xmlns="http://in2gesoft.ehu.es/ont/rules.owl#"
 xml:base="http://in2gesoft.ehu.es/ont/rules.owl"
 xmlns:pmo="http://in2gesoft.ehu.es/ont/activity061122.owl#"
 xmlns:org="http://in2gesoft.ehu.es/ont/organization061123.owl#"
 xmlns:prj1="http://in2gesoft.ehu.es/ont/ProjectInstance.owl#"
```

```
xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:sqwrl="http://sqwrl.stanford.edu/ontologies/built-ins/3.4/
  sqwrl.owl#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
xmlns:swrl="http://www.w3.org/2003/11/swrl#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:swrla="http://swrl.stanford.edu/ontologies/3.3/swrla.owl#"
xmlns:dc="http://purl.org/dc/elements/1.1/#"
xmlns:dcterms="http://purl.org/dc/terms/#">

<owl:Ontology rdf:about="">
 <rdfs:label>Rules associated to PMO</rdfs:label>
 <owl:versionInfo>v1.0 2011/05/10 12:00:00</owl:versionInfo>
 <rdfs:comment xml:lang="en">Rules associated to the PMO</
  rdfs:comment>
 <dc:title>Rules for Project Management Ontology</dc:title>
 <dc:subject>Rules for Project Management Ontology</dc:subject>
 <dc:description>Rules for Project Management Ontology</
  dc:description>
 <dc:creator>Fran Ruiz, Tecnalia</dc:creator>
 <dc:creator>Daniel Rodriguez, UAH</dc:creator>
 <dc:creator>Javier Dolado, UPV-EHU</dc:creator>
 <dc:date>2011-05-10 12:00:00</dc:date>
 <owl:imports rdf:resource="http://swrl.stanford.edu/ontologies/3.3/
  swrla.owl"/>
 <owl:imports rdf:resource="http://sqwrl.stanford.edu/ontologies/
  built-ins/3.4/sqwrl.owl"/>
 <owl:imports rdf:resource="http://in2gesoft.ehu.es/ont/
  organization061123.owl"/>
 <owl:imports rdf:resource="http://in2gesoft.ehu.es/ont/
  activity061122.owl"/>
 <owl:imports rdf:resource="http://in2gesoft.ehu.es/ont/ProjectA.owl
  "/>
</owl:Ontology>

<!-- RULE 1 -->
<!-- org:Project_Team_Member(?p) => sqwrl:select(?p) -->
<swrl:Imp rdf:ID="Rule-1">
 <swrl:body>
```

```
  <swrl:AtomList>
   <rdf:first>
    <swrl:ClassAtom>
     <swrl:classPredicate rdf:resource="&org;Project_Team_Member"/>
     <swrl:argument1>
      <swrl:Variable rdf:ID="p"/>
     </swrl:argument1>
    </swrl:ClassAtom>
   </rdf:first>
   <rdf:rest rdf:resource="&rdfs;nil"/>
  </swrl:AtomList>
 </swrl:body>
 <swrl:head>
  <swrl:AtomList>
   <rdf:rest rdf:resource="&rdfs;nil"/>
   <rdf:first>
    <swrl:BuiltinAtom>
     <swrl:arguments>
      <rdf:List>
       <rdf:rest rdf:resource="&rdfs;nil"/>
       <rdf:first rdf:resource="#p"/>
      </rdf:List>
     </swrl:arguments>
     <swrl:builtin rdf:resource="&sqwrl;select"/>
    </swrl:BuiltinAtom>
   </rdf:first>
  </swrl:AtomList>
 </swrl:head>
 <swrla:isRuleEnabled rdf:datatype="&xsd;boolean">
  false
 </swrla:isRuleEnabled>
</swrl:Imp>

<!-- RULE 2 -->
<!-- pmo:ManagementIssue(pmo:problemaSerio) AND
  -->
<!-- pmo:MaintenanceActivity(?a) =>
  -->
<!--   pmo:hasIssue(pmo:problemaSerio, pmo:revisionGeneralProyecto)
  -->
```

```
<!--    AND pmo:hasIssueManaged(pmo:problemaSerio, pmo:reactivo) AND
   -->
<!--    pmo:managesIssue(?a, pmo:problemaSerio)
   -->
<swrl:Imp rdf:ID="Rule-2">
 <swrl:head>
  <swrl:AtomList>
   <rdf:first>
    <swrl:IndividualPropertyAtom>
     <swrl:argument1>
      <rdf:Description rdf:about="&pmo;problemaSerio">
       <pmo:hasIssue rdf:resource="&pmo;revisionGeneralProyecto"/>
       <pmo:hasIssueManaged rdf:resource="&pmo;reactivo"/>
      </rdf:Description>
     </swrl:argument1>
     <swrl:argument2 rdf:resource="&pmo;revisionGeneralProyecto"/>
     <swrl:propertyPredicate rdf:resource="&pmo;hasIssue"/>
    </swrl:IndividualPropertyAtom>
   </rdf:first>
   <rdf:rest>
    <swrl:AtomList>
     <rdf:first>
      <swrl:IndividualPropertyAtom>
       <swrl:propertyPredicate rdf:resource="&pmo;hasIssueManaged"/>
       <swrl:argument1 rdf:resource="&pmo;problemaSerio"/>
       <swrl:argument2 rdf:resource="&pmo;reactivo"/>
      </swrl:IndividualPropertyAtom>
     </rdf:first>
     <rdf:rest>
      <swrl:AtomList>
       <rdf:first>
        <swrl:IndividualPropertyAtom>
         <swrl:argument1>
          <swrl:Variable rdf:ID="a"/>
         </swrl:argument1>
         <swrl:argument2 rdf:resource="&pmo;problemaSerio"/>
         <swrl:propertyPredicate rdf:resource="&pmo;managesIssue"/>
        </swrl:IndividualPropertyAtom>
       </rdf:first>
       <rdf:rest rdf:resource="&rdfs;nil"/>
      </swrl:AtomList>
```

```
      </rdf:rest>
     </swrl:AtomList>
    </rdf:rest>
   </swrl:AtomList>
  </swrl:head>
  <swrl:body>
   <swrl:AtomList>
    <rdf:first>
     <swrl:ClassAtom>
      <swrl:classPredicate rdf:resource="&pmo;ManagementIssue"/>
      <swrl:argument1 rdf:resource="&pmo;problemaSerio"/>
     </swrl:ClassAtom>
    </rdf:first>
    <rdf:rest>
     <swrl:AtomList>
      <rdf:first>
       <swrl:ClassAtom>
        <swrl:argument1 rdf:resource="#a"/>
        <swrl:classPredicate rdf:resource="&pmo;MaintenanceActivity"/
   >
       </swrl:ClassAtom>
      </rdf:first>
      <rdf:rest rdf:resource="&rdfs;nil"/>
     </swrl:AtomList>
    </rdf:rest>
   </swrl:AtomList>
  </swrl:body>
  <swrla:isRuleEnabled rdf:datatype="&xsd;boolean">
   false
  </swrla:isRuleEnabled>
</swrl:Imp>

<!-- RULE 3 -->
<!-- pmo:ManagementIssue(pmo:problemaNimio) AND -->
<!-- pmo:hasIssueManaged(?i, pmo:reactivo) AND -->
<!-- pmo:MaintenanceActivity(?a) =>             -->
<!--    pmo:managesIssue(?a, ?i)                -->
<swrl:Imp rdf:ID="Rule-3">
 <swrl:head>
  <swrl:AtomList>
   <rdf:first>
```

```
    <swrl:IndividualPropertyAtom>
     <swrl:argument1 rdf:resource="#a"/>
     <swrl:argument2>
      <swrl:Variable rdf:ID="i"/>
     </swrl:argument2>
     <swrl:propertyPredicate rdf:resource="&pmo;managesIssue"/>
    </swrl:IndividualPropertyAtom>
   </rdf:first>
   <rdf:rest rdf:resource="&rdfs;nil"/>
  </swrl:AtomList>
 </swrl:head>
 <swrl:body>
  <swrl:AtomList>
   <rdf:first>
    <swrl:ClassAtom>
     <swrl:classPredicate rdf:resource="&pmo;ManagementIssue"/>
     <swrl:argument1 rdf:resource="&pmo;problemaNimio"/>
    </swrl:ClassAtom>
   </rdf:first>
   <rdf:rest>
    <swrl:AtomList>
     <rdf:first>
      <swrl:IndividualPropertyAtom>
       <swrl:argument1 rdf:resource="#i"/>
       <swrl:argument2 rdf:resource="&pmo;reactivo"/>
       <swrl:propertyPredicate rdf:resource="&pmo;hasIssueManaged"/>
      </swrl:IndividualPropertyAtom>
     </rdf:first>
     <rdf:rest>
      <swrl:AtomList>
       <rdf:first>
        <swrl:ClassAtom>
         <swrl:argument1 rdf:resource="#a"/>
         <swrl:classPredicate rdf:resource="&pmo;MaintenanceActivity
  "/>
        </swrl:ClassAtom>
       </rdf:first>
       <rdf:rest rdf:resource="&rdfs;nil"/>
      </swrl:AtomList>
     </rdf:rest>
    </swrl:AtomList>
```

```
    </rdf:rest>
   </swrl:AtomList>
  </swrl:body>
  <swrla:isRuleEnabled rdf:datatype="&xsd;boolean">
   false
  </swrla:isRuleEnabled>
</swrl:Imp>

<!-- RULE 4 -->
<!-- pmo:ManagementIssue(pmo:personnelProblem) AND  -->
<!-- pmo:hasIssueManaged(?i, pmo:internal) AND       -->
<!-- pmo:ManagementActivity(?a) =>                   -->
<!-- pmo:managesIssue(?a, ?i)                        -->
<swrl:Imp rdf:ID="Rule-4">
 <swrl:head>
  <swrl:AtomList>
   <rdf:first>
    <swrl:IndividualPropertyAtom>
     <swrl:argument1 rdf:resource="#a"/>
     <swrl:argument2 rdf:resource="#i"/>
     <swrl:propertyPredicate rdf:resource="&pmo;managesIssue"/>
    </swrl:IndividualPropertyAtom>
   </rdf:first>
   <rdf:rest rdf:resource="&rdfs;nil"/>
  </swrl:AtomList>
 </swrl:head>
 <swrl:body>
  <swrl:AtomList>
   <rdf:first>
    <swrl:ClassAtom>
     <swrl:argument1>
      <rdf:Description rdf:about="&pmo;personnelProblem">
       <pmo:hasIssue rdf:resource="&pmo;personnelIssue"/>
       <pmo:hasIssueManaged rdf:resource="&pmo;internal"/>
      </rdf:Description>
     </swrl:argument1>
     <swrl:classPredicate rdf:resource="&pmo;ManagementIssue"/>
    </swrl:ClassAtom>
   </rdf:first>
   <rdf:rest>
    <swrl:AtomList>
```

```
      <rdf:first>
       <swrl:IndividualPropertyAtom>
        <swrl:argument1 rdf:resource="#i"/>
        <swrl:argument2 rdf:resource="&pmo;internal"/>
        <swrl:propertyPredicate rdf:resource="&pmo;hasIssueManaged"/>
       </swrl:IndividualPropertyAtom>
      </rdf:first>
      <rdf:rest>
       <swrl:AtomList>
        <rdf:first>
         <swrl:ClassAtom>
          <swrl:argument1 rdf:resource="#a"/>
          <swrl:classPredicate rdf:resource="&pmo;ManagementActivity"
   />
         </swrl:ClassAtom>
        </rdf:first>
        <rdf:rest rdf:resource="&rdfs;nil"/>
       </swrl:AtomList>
      </rdf:rest>
     </swrl:AtomList>
    </rdf:rest>
   </swrl:AtomList>
  </swrl:body>
  <swrla:isRuleEnabled rdf:datatype="&xsd;boolean">
   false
  </swrla:isRuleEnabled>
</swrl:Imp>

<!-- Some additional declarations -->
<rdf:Description rdf:about="&pmo;PerfectiveActivity">
 <pmo:managesIssue rdf:resource="&pmo;problemaSerio"/>
</rdf:Description>
<org:Individual_Role rdf:about="&prj1;PHPDeveloper"/>
<rdf:Description rdf:about="&pmo;CorrectiveMaintenance">
 <pmo:managesIssue rdf:resource="&pmo;problemaSerio"/>
</rdf:Description>
<org:Organization_Role rdf:about="&prj1;NGO"/>
<rdf:Description rdf:about="&pmo;schedulingActivity">
 <pmo:managesIssue rdf:resource="&pmo;personnelProblem"/>
</rdf:Description>
<rdf:Description rdf:about="&pmo;personnelActivity">
```

```
  <pmo:managesIssue rdf:resource="&pmo;personnelProblem"/>
 </rdf:Description>
 <org:Management_Responsibility rdf:about="&prj1;
    Management_Responsibility"/>
 <org:Project_Team_Member rdf:about="&prj1;JohnDoe"/>
 <org:Project_Team_Member rdf:about="&prj1;JaneDoe"/>
 <org:Group_Role rdf:about="&prj1;Development"/>
 <org:Organization_Role rdf:about="&prj1;MoneyMakingMachine"/>

</rdf:RDF>
```