Modelo jerárquico de roles para organizaciones de pequeño tamaño

Juan M. Pikatza, Iker U. Larburu, Francisco J. Sobrado, Juan J. García

[1] Departamento de Lenguajes y Sistemas Informáticos, UPV/EHU, Apdo. 649
20080 Donostia – San Sebastián
{jippiacj, jiblaeni, jibsootf,
jjga}@si.ehu.es

Resumen. La dificultad de implantar la mejora de procesos software en organizaciones de tamaño reducido nos lleva a definir jerarquías de roles aplicables durante el proceso de crecimiento de pequeñas organizaciones. Para facilitar la aplicación, se construirá un *framework* software accesible vía Web para la generación de Sistemas de Ayuda a la Decisión que permitan almacenar datos del proceso de desarrollo del software (PDS) y posibilitar así la optimización del proceso y el alcance de elevadas cotas de calidad. Mediante la consulta de la literatura sobre el tema y modelos y métodos más ampliamente aceptados, proponemos una definición de modelo jerárquico de roles compatible para diferentes tamaños de la organización. Como complemento al modelo se propone una arquitectura de herramienta de soporte para el almacenamiento de información y ayuda a la decisión en el PDS. La relación entre tareas, artefactos y roles define un proceso. Con procesos de diferentes tipos puede implementarse el desarrollo basado en el dominio mediante la reutilización parcial o total de activos generados en proyectos anteriores o componentes desarrollados mediante procesos destinados al desarrollo de componentes. El uso de estos activos ya probados incrementa la calidad del producto final, facilita la planificación y disminuye el tiempo de desarrollo. Crecer con un modelo hace que la implantación de procesos de mejora sea más sencilla cuando la organización alcanza un nivel medio o grande.

1 Introducción

Que la definición de un proceso de desarrollo del software apropiado puede mejorar la efectividad y eficiencia de la ingeniería del software es algo ampliamente aceptado y, como consecuencia de ello, exigido por muchos clientes. Sin embargo, a las pequeñas compañías les resulta difícil asignar recursos para la mejora del proceso de desarrollo del software (SPI) y adaptar los modelos de procesos existentes a sus necesidades. Muchos de los conocidos modelos de proceso de desarrollo del software y modelos de referencia, tales como el *Capability Maturity Model* (CMM) [3] que esta siendo ampliamente aceptado en EEUU y ahora también en Europa, ofrecen una buena base para el SPI pero también presentan un excesivo costo si se desea aplicarlos estrictamente. Más concretamente, no contemplan los aspectos de negocio y el

hecho de que, en diferentes situaciones, sean necesarios diferentes modelos de proceso. Gestionar con éxito el desarrollo de productos software, con evidencia visible del progreso, exige una visión más global del proceso. No es suficiente con tener un único modelo de proceso idóneo.

1.1 El problema

Para evitar el caos es necesario enfocar en el proceso de desarrollo del software. Sin embargo, los modelos existentes proponen un elevado número de roles de organización, en el caso de CMM son 25, que muchas veces no es posible cubrir con el personal disponible y necesitan ser escalados para adecuarlos a las necesidades de las pequeñas compañías. Por otra parte, dichos modelos deben ser aplicables de forma continuada y adaptarse al crecimiento de la compañía para ser útiles.

1.2 Objetivos y alcance

El objetivo de nuestra investigación es el desarrollo de modelos de mejora del PDS para pequeñas organizaciones (menos de 50 trabajadores) y adaptables al cambio. En su desarrollo se están utilizando modelos bien conocidos, bien descritos, y disponibles públicamente como CMM y PSP/TSP (*Personal Software Process/Team Software Process*)[6] junto con otros procesos de ingeniería del software e ingeniería del conocimiento como *Racional Unified Process*® (RUP) [7] y CommonKADS [13]. Alcanzar el nivel 2 de CMM cuando todavía la organización es pequeña facilita el alcance de niveles superiores cuando el tamaño pase a ser medio (> 50 empleados). La evolución puede llegar a ser extremadamente rápida, deberían seguirse diferentes modelos de mejora de procesos en función del ritmo de evolución.

1.3 Antecedentes

La implantación de programas de mejora de la calidad en el desarrollo del software en organizacións pequeñas requiere unos recursos de los éstas no disponen. Existen modelos como CMM, BOOTSTRAP[2], SPICE [5], PSP/TSP y P-CMM (*People Capability Maturity Model*) [4] aplicables a organizacións medianas y grandes. A pesar de la importancia adquirida por el modelo CMM, existen trabajos tendentes a adaptar procesos como P-CMM para pequeñas compañías [10].

P-CMM pretende mejorar el rendimiento de los activos humanos de las organizaciones de desarrollo de software funcionando como una guía para corregir continuamente su desarrollo, motivación, y atracción de estrategias de talento. Es un marco de madurez que no define métodos específicos para alcanzar los objetivos sino que define los objetivos y los pasos necesarios para alcanzarlos.

En cuanto a los intentos de adaptar CMM para pequeñas organizaciones, existen varios trabajos recientes que enfocan el problema de diferentes maneras:

1. *Dynamic CMM* [9] que es capaz de dar soporte a grupos desde 2 personas en adelante dando soporte al crecimiento de la organización.
2. La aproximación de matriz para la definición de procesos de [14] que define cuatro procesos software que cumplen el estándar ISO y tres criterios de adaptación de los mismos que permiten adecuar un proceso a un proyecto, de ésta forma se consigue un proceso que satisface los requerimientos de la NASA con respecto a los estándares ISO 9001 y CMM.
3. El modelo EPRAM [1] que adapta el modelo CMM para pequeñas organizacións (< 12 personas) de comercio electrónico, esta diseñado para cumplir los requisitos del nivel 2 de CMM.
4. El framework, inacabado, para la gestión del desarrollo de productos software propuesto en [12] combina gestión del negocio con gestión de procesos mediante cuatro ciclos de control.

Dado que el PDS esta sujeto a evolución y puede alcanzar una gran complejidad dependiendo del tamaño de la organización, distribución geográfica de los componentes del equipo de desarrollo, multiplicidad de proyectos en curso, multiplicidad de dominios, presión para acortar plazos con respecto a la competencia, la calidad de producto requerida por el mercado y la exigencia por parte de los clientes de haber sido evaluado en un determinado nivel (≥ 2) de CMM; es necesario gestionar el abundante conocimiento a utilizar en la definición y actualización de los PDS. Una de las características principales de los procesos necesarios es la reutilización de activos (assets) correspondientes tanto a la ingeniería del software como a la ingeniería del conocimiento, lo que hace necesaria la integración de la gestión del conocimiento en la gestión del PDS. Existen varias aproximaciones para la integración de la Gestión de Procesos de Negocio y la Gestión del Conocimiento, en [11] podemos encontrar una que enfoca en el modelado de procesos de negocio de conocimiento intensivo y débilmente estructurados en un framework que considera tareas relacionadas con el conocimiento y objetos de conocimiento con una herramienta de workflow que implementa el meta-modelo teórico.

La incorporación del conocimiento, implica la búsqueda de un proceso para su incorporación a un proceso de negocio. En éste sentido, se pueden mencionar los resultados de los proyectos desarrollados por el *Software Process Improvement Research Action Laboratory* de Finlandia [8]. La fuerte distribución geográfica y de organización en una "Corporación de Software Virtual" parece que es la solución de muchos aspectos del problema pero lleva, también, a nuevos problemas. Los procesos en el desarrollo de software, han sido ampliamente estudiados y existen medios apropiados para sus estructuración y gestión, p.e. modelos de proceso detallados, algo que esta lejos de suceder en la gestión del conocimiento. En [8] podemos encontrar el Modelo de Proceso de Gestión del Conocimiento como medio de integración de la Gestión del Conocimiento con procesos de negocio.

1.4 Metodología

En el desarrollo de un modelo de PDS para pequeñas organizaciones, propondremos un modelo que aplicaremos a casos de estudio para obtener mediciones, analizarlos y validar el modelo.

En éste proceso, es necesario disponer de un soporte informático que facilite la utilización real del modelo y permita gestionar la información generada en la definición, actualización y ejecución del modelo de PDS.

En el desarrollo del modelo, basado en el sentido común y en la experiencia en el desarrollo del software, se han tomado como base los modelos CMM, TSP, y PSP junto con el RUP incorporando conceptos de CommonKADS para los aspectos relativos al conocimiento. La idea básica es que los usuarios, además de adquirir una visión del modelo, de la magnitud del esfuerzo requerido y de los roles, responsabilidades, tareas y documentos; puedan implantar su uso con la ayuda de un sistema informático de soporte. Con estas intenciones, hemos interpretado, reducido y reorganizado el CMM original y tenido en cuenta los conceptos definidos en CommonKADS y RUP.

1.5 Contenidos del trabajo

En la siguiente sección, se destina a relacionar los métodos y recursos utilizados en le presente trabajo. En la sección tercera, El modelo de jerarquía de roles. Y en la sección cuarta, las conclusiones más importantes.

2 Materiales y métodos

A partir de trabajos previos, principalmente [9] [12], nos hemos encaminado hacia el establecimiento de un modelo de jerarquía de roles que admita múltiples procesos y tipos de proyectos cuando el tamaño de la organización lo permita.

Dado la escasa aplicación de los modelos existentes, nos proponemos también construir una herramienta para la gestión de procesos y proyectos. Para ello, como procesos para el ciclo de vida, se tendrán en cuenta RUP [7] y CommonKADS [13] en lo relativo a la gestión del conocimiento incluyendo las plantillas de artefactos que incorporan.

El Sistema de Ayuda a la Toma de Decisiones (DSS) utilizará el API de Protégé-2000[1] para la definición y actualización de ontologías y la adquisición del conocimiento y el API de Jess[2] para implementar las funciones de motor de inferencia.

[1] http://protege.stanford.edu/index.html
[2] http://herzberg.ca.sandia.gov/jess/

3 Resultados

Como solución a los problemas antes expresados se realizan dos propuestas: por una parte un modelo de organización escalable según el número de empleados [9] con la posibilidad de definición de tipos de proyecto [12], por otra parte, la arquitectura de una herramienta de gestión de procesos y proyectos denominada SPK (*Software Prozesuen Kudeatzailea*).

La primera propuesta incluye un modelo de organización escalable en el número de empleados. Este modelo se muestra las tres escalas en las que se ha desarrollado: S (*Small*) cuando se disponen de 40-80 empleados (Fig. 1), XS (*eXtra Small*) cuando se disponen de 10-30 empleados (Fig. 2) y XXS (*eXtra eXtra Small*) cuando se disponen de 2-10 empleados (Fig. 3).

La Tabla 1 además de recoger los roles que se mencionan en CMM para el nivel 2 también indica (anotándolos en cursiva) los roles necesarios para que el modelo pueda cubrir el uso de tipos de proyecto. El rol anotado en negrita (Customer SQA) es un rol optativo para el modelo, solo aparecerá en caso que el cliente exija su inclusión en el proyecto que haya contratado.

Tabla 1. Roles por categorías según número de empleados.

Roles en CMM	***Abrev.***	***S***	***XS***	***XXS***
Senior Manager	SM	SM	SM	
Project Type Manager	PTM	PTM		
Project Manager	PM	PM	PM	PM
SW Quality Assurance Group	SQA	SQA	SQA	SQA
Project Type SQA	PTSQA	PTSQA		
Customer SQA	**CSQA**	**CSQA**	**CSQA**	**CSQA**
Project Type SCM	PTSCM	PTSCM		
Project SCM	PSCM	PSCM	PSCM	
SW Configuration Management	SCM	SCM	SCM	SCM
SW Configuration Control Board	SCCB	SCM		
SW Engineering Group	SE	SE	SE	SE
System Group	SG	SG	SG	SG
System Test Group	STG	STG	STG	STG

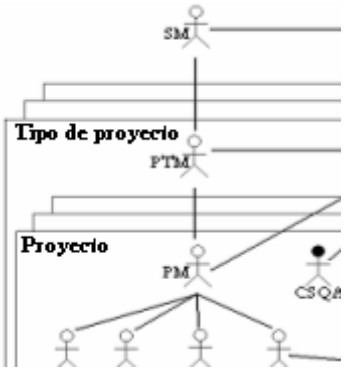
El modelo de organización de las tres escalas mencionadas es el que se muestra en las figuras 1, 2 y 3. Las siglas de los actores corresponden a los roles que desempeñan, y las líneas de unión las comunicaciones esenciales que mantienen. El ámbito de cada rol queda indicado mediante rectángulos.

La Tabla 2 recoge las actividades y el rol (ver abreviaturas en Tabla 1) que los realiza según el tamaño de la organización.

Tabla 2. Actividades y roles según número de empleados.

Actividades fijadas en CMM	*S*	*XS*	*XXS*
Revisiones continuas	SQA, PTSQA	SQA	PM
Revisiones periódicas	SM, PTM, PM	SM	PM
Revisiones dirigidas por evento	PTM, PM	PM	PM
Revisiones de requisitos, planes, producto, actividades y acuerdos	SE	SE	SE
Revisiones de cambios en la línea base	SCM	SCM	SCM
Revisar acuerdos a entes externos a la organización	SM	SM	PM
Informar de resultados de revisiones de calidad	SQA	SQA	SQA
Planificar proyecto	PM	PM	PM
Vigilar y seguir el proyecto	PM	PM	PM
Aprobar cambios en las líneas base	PSCM	PSCM	SCM

El modelo define tres niveles: el nivel global, el nivel de tipo de proyecto y el nivel de proyecto. Los proyectos pertenecen a algún tipo y existen roles que actúan sobre todos los tipos de proyectos.



Un rol tiene visibilidad sobre todo aquello que se incluye dentro del nivel en el que esta definido. A mayor nivel mayor visibilidad, y por lo tanto mayor necesidad de usar abstracciones y delegar en roles de niveles inferiores.

Cuando la organización reciba la petición de realizar un nuevo proyecto el SM lo dirigirá a un tipo de proyecto predefinido que realice la organización, abrirá un nuevo tipo de proyecto (si lo ve necesario) o rechazará el proyecto. Una vez el proyecto es clasificado en un tipo de proyecto el PTM se asegurará de que se dispone de recursos adecuados para realizar el proyecto y creará un grupo de proyecto. En este momento el PM del grupo de proyecto es el responsable de la gestión del proyecto y solo se recurrirá a niveles superiores para asuntos que no puedan ser solucionados a nivel de proyecto.

El PTSQA del tipo de proyecto realizará inspecciones de los proyectos de su tipo. Independientemente, si el cliente lo ha solicitado, el CSQA realizará inspecciones del proyecto al que este asociado. En caso de irregularidades se informará a los afectados por la revi-

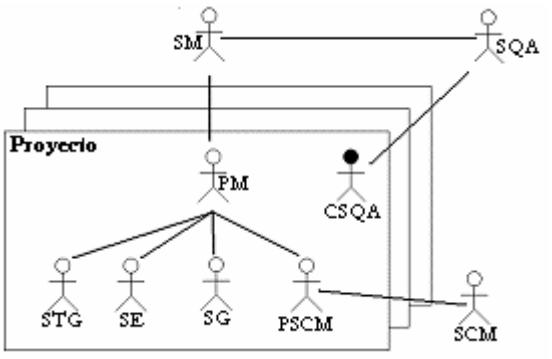


Fig. 2: Modelo de organización XS

sión y al PM del proyecto en cuestión. Se informarán en este orden a los siguientes roles mientras la irregularidad persista: PTM (vía PTSQA) y SM (a través del SQA).

El modelo de organización XS se diferencia del modelo S en la desaparición del nivel de tipo de proyecto y los roles que este incluía.

El modelo de organización XXS es tan pequeño que solo afronta la realización de un proyecto a la vez. Incluso llegando a tamaños extremadamente pequeños podría considerarse agrupar los roles en dos: los desarrolladores (SG y STG) y los gestores (PM, SQA, SE y SCM).

Independientemente del tamaño de la organización, para un desarrollo de calidad es necesario mencionar dos conceptos: la reutilización (de activos ya creados, pero también de paradigmas de desarrollo) y el desarrollo basado en el dominio (conocimiento del área en la que se mueve la organización).

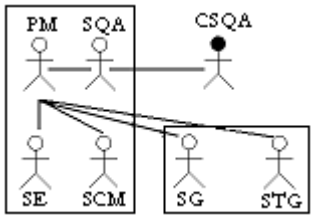


Fig. 3: Modelo de organización XXS

Como desarrollo de un proyecto viene marcado por un proceso concreto, un proyecto será una instancia de un proceso. Un proceso es un ente abstracto del que solo se pueden obtener estimaciones, mientras que un proyecto es un ente real que, durante y después de su desarrollo, produce una serie de métricas. La Fig. 4 muestra la visión que tienen las organizaciones del modelo propuesto frente a los procesos.

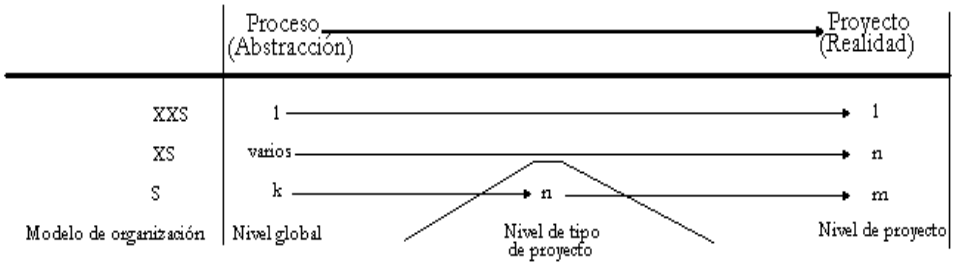


Figura 4: Visión de los procesos.

Una organización del tipo XXS solo realizará un proyecto a la vez, tendrá definido un proceso de desarrollo que utilizará para realizar todos sus proyectos. A medida que la organización crezca (XS) comenzará a desarrollar viarios proyectos a la vez, lo que motivará que aparezcan nuevos procesos de desarrollo según el tipo de proyectos que realice. Cuando la organización alcance un tamaño mayor (S), ya no solo tendrá una batería de procesos generales a nivel global, sino que habrá especializado algunos de estos procesos a nivel de tipo de proyecto, obteniendo procesos especializados e incluso parcialmente desarrollados.

La segunda propuesta de la solución es una arquitectura de herramienta para la gestión de procesos y proyectos (SPK). Esta herramienta se implementará mediante una arquitectura cliente/servidor, con cliente fino y comunicación vía Web. La arquitectura de la herramienta en el lado del servidor se muestra en la Fig. 5.

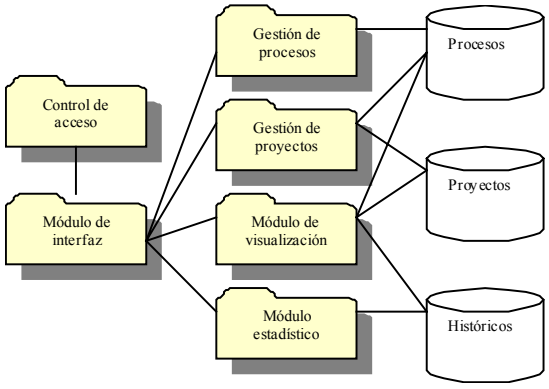


Fig. 5: Arquitectura de SPK

SPK ofrecerá facilidades de gestión a nivel de proceso y proyecto. Como se observa en la Fig. 6, hay dos niveles bien definidos: (por encima de la línea negra) el nivel de proceso y (por debajo de la línea negra) el nivel de proyecto. El nivel de proyecto es prácticamente igual al nivel de proceso porque un proyecto es la instancia de un proceso.

Un proceso se define especificando tuplas {qué se hace, quién lo hace, cuándo se hace}. En el modelo esto se refleja en otra tupla, {artefacto, rol, tarea}, el artefacto representa lo qué se ha de hacer, el rol quién ha de hacerlo y la tarea el cuándo. El proyecto, como instancia de un proceso, se representa también como instancias de la tupla anterior.

Como cualquier tarea se puede descomponer en tareas mas pequeñas, tanto el proceso, como el proyecto, se definen como una tarea de alto nivel estando relacionada con el sistema o producto que produce como artefacto y todos los roles que participan.

Una tarea puede ser:

a) *Una tarea simple,* una tarea que no puede ser descompuesta.

b) *De tipo secuencial*, una tarea que contiene subtareas. Estas subtareas están relacionadas mediante dependencias temporales y tienen que completarse todas las subtareas para dar la tarea secuencial por terminada.

c) *De tipo elección*, una tarea que abre un abanico de posibilidades. Para dar una supertarea de este tipo por finalizada basta con finalizar una de las subtareas que contiene.

d) *De tipo ciclo*, una tarea que va a repetirse una y otra vez hasta que se da una condición de salida.

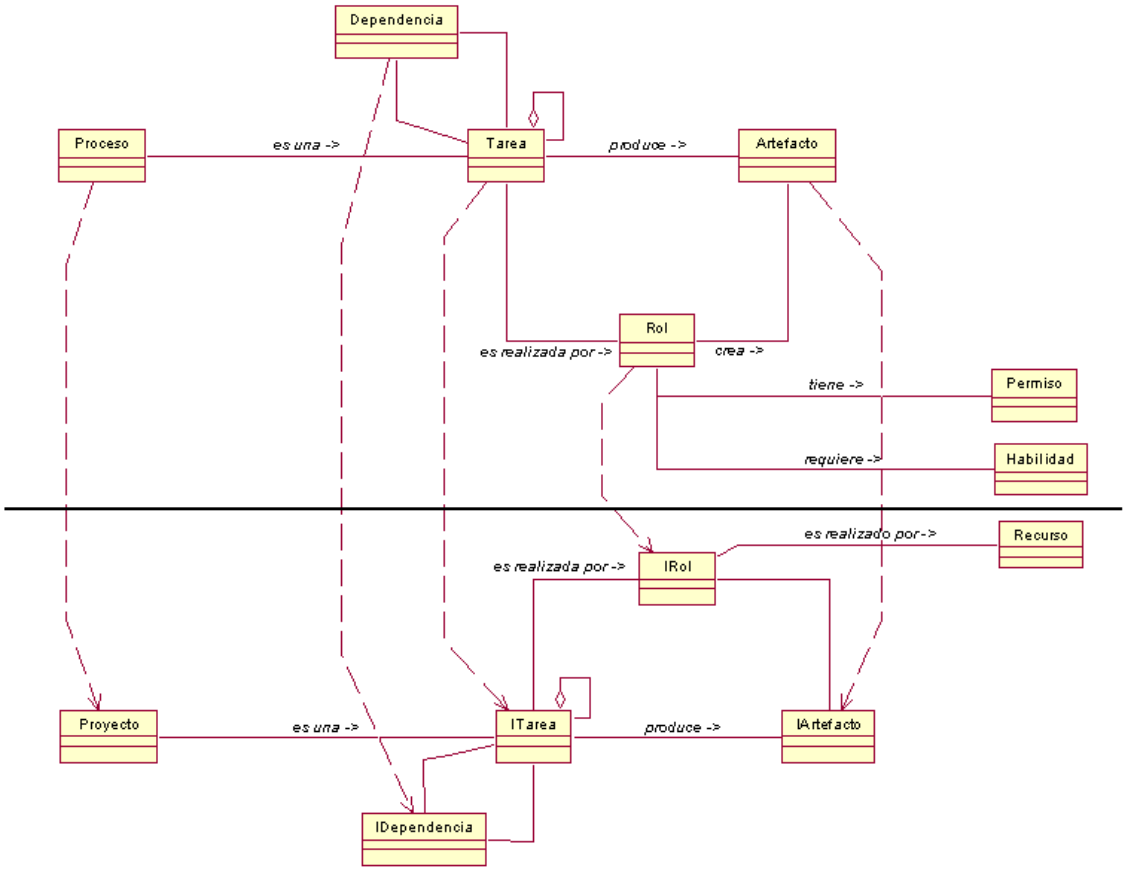


Fig. 6: Diseño de clases para la definición de procesos.

Las tareas de tipo secuencial, de tipo elección y de tipo ciclo se consideran supertareas porque pueden contener subtareas en su interior. Esta distribución crea una jerarquía de tareas, es decir, un proceso es una tarea de nivel 1, cualquier subtarea directa una tarea de nivel i es una tarea de nivel i+1.

Pueden existir dependencias temporales entre dos tareas, sin importar el nivel que estas pudieran tener. Las dependencias existentes pueden ser cuatro: Comienzo-Comienzo (CC), Comienzo-Fin (CF), Fin-Comienzo (FC) y Fin-Fin (FF). Teniendo todo esto en cuenta un proceso se modela de la siguiente manera:

1. Un proceso es una tarea simple o una supertarea, dependiendo del grado de especificación necesario. Normalmente será una supertarea. Dependiendo de la complejidad del proceso puede no necesitar de la descomposición en sub-

tareas. En el caso general un proceso será definido desde las tareas de mas alto nivel hacia lo especifico (*top down*), la posibilidad de descomponer cualquier tarea en subtareas permite llegar hasta un nivel de detalle adecuado para las necesidades de cualquiera (ver Fig. 7.a).

2. Para posibilitar las relaciones entre dos tareas sin importar el nivel de éstas, cualquier supertarea define implícitamente dos hitos: el de comienzo y el de fin de la supertarea. Cualquier subtarea directa tiene una dependencia FC entre el hito de comienzo y ella, así como una dependencia FC entre ella y el hito de fin. De esta manera las dependencias con la supertarea se modelan como dependencias con los hitos de comienzo y fin de la supertarea.
3. Las tareas directas de las supertareas de tipo elección no pueden tener más dependencias temporales que las mencionadas en el punto dos con los hitos de comienzo y fin de la supertarea (ver Fig. 7.b).
4. Para modelar las supertareas de elección y ciclo existen tres tipos de hitos especiales:
    a. OR, que mediante una condición permitirán elegir entre varias subtareas a realizar.
    b. XOR, igual que los hitos OR pero permitiendo solo un único camino
    c. C(n), que se considerarán cumplidos cuando n de los caminos que da a él este satisfecho
    d. Condicionales, que mientras no se cumpla la condición expresada en él obligarán a repetir la supertarea
5. El hito de comienzo de una supertarea de elección es un hito OR o un hito XOR, mientras que el hito de fin es un hito C
6. El hito de comienzo de una supertarea de ciclo es un hito normal mientras que el de fin es un hito condicional

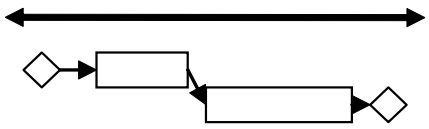


Fig. 7.a: Supertarea secuencial

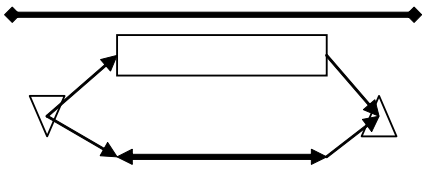


Fig. 7.b: Supertarea de elección

SPK pretende manejar un repositorio de activos. Este repositorio es esencial para el desarrollo basado en el dominio. En este repositorio se mantendrán artefactos, tanto como plantillas como artefactos totalmente desarrollados de proyectos concretos, procesos y los proyectos realizados como históricos.

Este repositorio posibilitará la obtención de estimaciones cada vez más realistas para aplicar en futuros proyecto. Los componentes o artefactos esenciales de cada dominio serán identificados y estarán accesibles, proporcionando calidad (utilización de componentes ya probados) y eficiencia (menor tiempo de desarrollo al usar elementos desarrollados previamente).

Como último punto a comentar se pretende integrar dos herramientas como Protege-2000 y Jess en SPK. Protégé-2000 se utilizará como editor de ontologías, y al ofrecer la posibilidad de generar plantillas para adquirir instancias, también como herramienta de adquisición de conocimiento. Por otra parte, se usará Jess como motor de inferencia para razonar sobre el conocimiento adquirido.

4 Conclusiones

El modelo propuesto permite, conocido el tamaño de la organización, identificar un modelo adecuado par comenzar con la implementación de un proceso de mejora del PDS cuando el tamaño de la organización es aún controlable. El crecimiento sin ajustarse a ningún modelo puede llevar a un punto en el que los cambios no se pueden dar, aún reconocida su viabilidad, por la oposición de los componentes de la organización.

Cuando el número de empleados se incremente, y sea viable desarrollar varios proyectos (incluso de varios tipos de proyecto) en paralelo, el modelo ofrece una dirección de crecimiento a la organización identificando niveles superiores al de un simple proyecto con la aparición de nuevos roles.

Los tipos de proyecto pueden servir para el desarrollo de aplicaciones dentro de las líneas de producto de la organización o el desarrollo de componentes reutilizables en cualquier tipo de proyecto. Las diferentes líneas de producto definen el dominio de la organización.

La reutilización sistemática de activos, componentes software o artefactos, facilita la planificación, reduce el tiempo y costos de desarrollo, y aumenta la calidad final del producto al usar componentes probados anteriormente. Al trabajar en un dominio concreto con una variabilidad conocida, utilizando los datos históricos, permite hacer estimaciones más precisas sobre futuros desarrollos.

References

1. Antón A.I, Carter R.A., Srikanth H., Sureka A., Williams L.A., Yang K. and Yang L. Tailored CMM for a Small e-Commerce Company Level 2: Repeatable. North Carolina State University Department of Computer Science Technical Report TR-2001-09, (2001)
2. Bicego, A., Khurana, M., and Kuvaja, P., BOOTSTRAP 3.0 – A SPICE conformant software process assessment methodology, in Hawkins, C., Ross, M. and Staples, G., (Eds),

Software Quality Management VI – Quality Improvement Issues, Springer- Verlag, London, pp. 26 – 37, (1998)
3. Carnegie Mellon University / Software Engineering Institute, The Capability Maturity Model: Guidelines for Improving the Software Process, II. Series, Addison-Wesley,(1995).
4. Curtis B., Hefley W.E. and Millar S. Overview of the People Capability Maturity Model, CMU/SEI-95-MM-01, Carnegie Mellon University, (1995)
5. El Emam K., Drouin J.N., Melo W.(Eds.). Spice: The Theory and Practice of Software Process Improvement and Capability Determination. IEEE Computer Society; ISBN: 0818677988 (1998)
6. Humphrey W. The Team Software ProcessSM (TSPSM), Technical Report CMU/SEI-2000-TR-023 (2000)
7. Kruchten P. The Rational Unified Process. An Introduction, 2nd ed. Booch, Jacobson y Rumbaugh (Ed.) The Addison-Wesley Object Technology Series, Mass. (2000)
8. Kucza, T. and Komi-Sirviö, S. Utilising Knowledge Management in Software Process Improvement - The Creation of a Knowledge Management Process Model. In the 7th International Conference on Concurrent Enterprising (ICE) (2001).
9. Laryd A. and Terttu Orci T. Dynamic CMM for Small Organizations, Proceedings ASSE 2000, the First Argentine Symposium on Software Engineering, Tandil, Argentina, pp. 133--149, (2000)
10. Olofsson S. Tailoring P-CMMSM for Small Companies. Department of Computer Science, Umea University. UMINF-00.1, ISSN-0348-0542, (2000)
11. Papavassiliou, G., Ntioudis S., Mentzas G. and Abecker A Business Process Knowledge Modelling: Method and Tool. Presented at the DEXA conference, TAKMA-2002, Third International Workshop on Theory and Applications of Knowledge Management, Aix-en-Provence, 2-6 (2002)
12. Rautiainen K., Lassenius C., VähÄniitty J., Pyhäjärvi M. and Vanhanen J. A tentative Framework for Managing Software Product Development in Small Companies. Proceedings of the Hawai'i International Conference on System Science, January 7-10, 2002, Big Island, Hawaii (2002)
13. Schreiber G., Akkermans H., Anjewierden A., de Hoog R., Shadbolt N., Van de Velde W., and Wielinga B. Knowledge Engineering and Management: the CommonKADS Methodology. A Bradford Book, The MIT Press (1999)
14. Schultz D., Bachman J., Landis L., Stark M., Godfrey S. and Morisio M. A Matrix Approach to Software Process Definition. Software Engineering Workshop Greenbelt, MD, (2001). Available from: http://mabwww.gsfc.nasa.gov/papers/DOCS/AMApproach.pdf