Evaluación de tecnología Java para la implantación de un proceso de desarrollo de software en PYMEs

Larburu[1] I.U., Pikatza[1] J.M., Sobrado[1] F.J., Garcia[2] J.J. y López de Ipiña[3] D.

[1]Departamento de Lenguajes y Sistemas Informáticos (UPV/EHU)‡
{jiblaeni, jippiacj, jibsootf} @ sc.ehu.es
[2]Wireless Systems Research Department, Avaya Labs Research, Basking Ridge, NJ (USA)
jjga@avaya.com
[3]Consulting Department, 3G LAB, Cambridge (England)
dipina@3glab.com

Abstract. Debido a los beneficios que ofrece la producción de software siguiendo un proceso definido y a las exigencias de calidad de los clientes, la definición e implantación de Procesos de Desarrollo de Software (PDS) se ha convertido en prioritaria, lo cual es una tarea dificultosa para las pequeñas organizaciones por el elevado número de roles a definir. En este artículo mostramos un esquema organizativo para organizaciones de pequeño tamaño y el PDS correspondiente para la elaboración de proyectos. El PDS, construido en base a procesos existentes, cumple los estándares SW-CMM y CMMI hasta el nivel 2 y puede evolucionar con la organización. De cara a la implantación del PDS definido, presentamos una arquitectura de Sistema de Ayuda a la Toma de Decisiones que ofrece gestión de datos y ayuda a la decisión multicriterio (MCDA) en base a dicho PDS. El prototipo desarrollado, utilizando tecnología Java, Protégé-2000 y Jess ofrece un comportamiento eficiente.

Palabras clave. Gestión de procesos, calidad del software, *Capability Maturity Model, Rational Unified Process*.

1 Introducción

Para dar una solución efectiva a los retos socioeconómicos actuales es imprescindible contar con las Tecnologías de la Información y Comunicación, y desarrollar bajo la disciplina que exige un proceso definido e implantado [4] por las ventajas que ofrece: mayor efectividad, eficacia, rendimiento, capacidad de negocio y predictibilidad.

Podemos definir un proceso como un conjunto de pasos a realizar (tareas) llevados a cabo por responsables (roles) para producir elementos (artefactos) de uso interno o entregables usando para ello herramientas y recursos disponibles [15]. El proceso podrá ser optimizado en base a mediciones realizadas sobre proyectos anteriores.

‡ Fac. Informática (UPV/EHU), Dep. L.S.I, Apdo. 649, 20080 Donostia - San Sebastián.

Las organizaciones apuestan por un desarrollo de calidad tanto como medida de reducción de costos como por exigencias por parte de los clientes de cumplimiento de determinados niveles de calidad definidos en estándares establecidos. Sin embargo, son minoritarias las organizaciones que trabajan bajo la disciplina de un proceso.

El problema reside en que cuando las organizaciones son pequeñas es más fácil definir e implantar un proceso, pero no pueden afrontar el esfuerzo organizativo y de gestión que supone. Por el contrario, cuando la organización tiene una tamaño mediano o grande la definición del proceso es compleja pero aún más la implantación por los hábitos de trabajo adquiridos y la resistencia a los cambios que impone el proceso.

En el presente trabajo, nos hemos propuesto: 1) Definir un modelo evolutivo de organización y desarrollo del software (MEODS), aplicable cuando la organización es pequeña y que permita evolucionar hacia modelos de calidad certificados y 2) Definir la arquitectura de un Sistema de Ayuda a la Toma de Decisiones (DSS), estrategia de implantación de procesos recomendada por diversos autores [21], capaz de facilitar la gestión de la información sobre un proyecto e implementar procesos que incluyen ayuda a la decisión multicriterio (MCDA) como medio efectivo de implantación.

El articulo esta organizado de la siguiente manera: seguimos con los trabajos previos relacionados, continuando con los métodos y recursos empleados, para proseguir con los resultados conseguidos, y acabar con las conclusiones obtenidas.

2 Trabajos previos

Existe abundante literatura sobre diferentes aspectos de la definición e implantación de procesos. En cuanto a modelos de calidad existentes, utilizados para evaluar procesos y emitir certificaciones, definen las áreas de interés clave pero no sus detalles de implementación. Estos modelos se pueden clasificar en dos bloques:

1. Los modelos por etapas que definen, por cada nivel, las áreas de interés que se deben cumplir. Los más extendidos son *Software Capability Maturity Model* (SW-CMM) [16] o la versión por etapas de *CMM-Integration* (CMMI) [6].
2. Los modelos continuos que definen el nivel de calidad por cada área de interés. Entre ellos SPICE (o ISO 15504) [8], Bootstrap [3] o CMMI versión continua.

El modelo SW-CMM ha gozado de gran aceptación pero define un excesivo número de roles para las pequeñas organizaciones. Se ha investigado en dos líneas: 1) El desarrollo de modelos complementarios al SW-CMM para la mejora del rendimiento individual, *Personal Software Process* [10], y para la mejora el rendimiento de los grupos de desarrollo, *Team Software Process* [11]. 2) Adaptaciones de SW-CMM para pequeñas organizaciones, por ejemplo, *Dynamic-CMM* (D-CMM) [14] que ofrece un modelo de organización para grupos de dos personas en adelante, EPRAM [1] dirigido específicamente al ámbito del comercio electrónico, *Matrix Approach* [24] que define modelos que cumplen con ISO y sus criterios de adaptación, o el *Framework* inacabado de Rautiainen [18] para la gestión de procesos.

En cuanto a procesos de desarrollo detallados para el ciclo de vida del software, los más extendidos son *Rational Unified Process* (RUP) [13] para la ingeniería del

software y CommonKADS [22] para la ingeniería del conocimiento. Además, se pueden encontrar plantillas de documentos en otras fuentes, por ejemplo IMB-BC[?] .

Para la implantación del proceso mediante DSSs accesibles vía Web, existe tecnología como la recogida en la especificación J2EE[?] con la que construir el sistema de almacenamiento de datos (*Data Warehousing*). A este sistema hay que añadir el sistema de gestión del conocimiento mediante la representación de procesos siguiendo diferentes formalismos (Tabla 1) y diferentes métodos de ayuda a la decisión multicriterio (MCDA) [2].

Tabla 1. Comparativa de formalismos de representación de procesos

		RF	RG	Ab	E	CD	Ad	MV
Gramáticas formales		✍✍		✍✍	✍✍		✍✍	
Diagramas de estado	[9]		✍✍	✍✍	✍✍	✍✍	✍✍	
Redes de Petri	[19]	✍✍	✍✍		✍✍			
Diagramas IDEF	[12]		✍✍	✍✍	✍✍		✍✍	
Diagramas PERT		✍✍	✍✍	✍✍	✍✍		✍✍	✍✍
Diagramas Gantt			✍✍	✍✍	✍✍		✍✍	
BPML	[5]		✍✍	✍✍	✍✍	✍✍	✍✍	
GLIF	[17]		✍✍	✍✍	✍✍	✍✍	✍✍	

>Representación formal (RF)
>Representación gráfica (RG)
>Abstracción (Ab) posibilidad de definir submodelos
>Ejecución (E)
>Capacidad Decisión (CD)
>Adaptabilidad (Ad)
>Múltiples Vistas (MV)

3 Materiales y métodos

Para la definición del MEODS para la fase de elaboración de un proyecto informático hemos seleccionado SW-CMM y CMMI como modelos de calidad por que fijan las áreas clave necesarias para cada nivel de calidad, RUP como proceso de desarrollo detallado y algunas plantillas existentes en el IMB-BC.

De SW-CMM y CMMI nos fijamos en las *Key Process Area* (KPA) necesarias para alcanzar el nivel 2. Una KPA es un conjunto de objetivos a cumplir. Estas KPAs están complementadas con las disciplinas del RUP, que son equivalentes a las KPAs de CMM en cuanto que recogen un conjunto de tareas con un objetivo conocido.

El SW-CMM define también una jerarquía de roles de gestión y un conjunto de grupos auxiliares que hemos incorporado en el MEODS interpretándolos todos ellos como roles. En cuanto a la organización de los roles seleccionados, nos hemos basado en las relaciones entre roles que propone el D-CMM [14] y en la estructura organizativa que define el *Framework* de Rautiainen [18].

Para la implantación del MEODS mediante un DSS hemos considerado tres aspectos: el modelado de procesos como guías, la incorporación de la ayuda a la decisión multicriterio, y los métodos y recursos para la construcción del DSS.

Para el modelado de procesos hemos optado por GLIF que es un modelo para la representación de guías médicas desarrollado por *InterMed* [17]. La necesidad de flujos de trabajo flexibles para la representación de guías médicas utilizadas en nuestra línea de investigación de telemedicina, es interesante también para el PDS, por lo que nos hemos propuesto experimentar con GLIF también en esta línea.

[?] IMB, Government of British Columbia. http://srmwww.gov.bc.ca/imb/3star/index.html
[?] Java 2 Platform, Enterprise Edition. http://java.sun.com/j2ee

La versión actual de GLIF, la 3.5, permite la definición de guías a tres niveles: nivel conceptual o de representación mediante diagramas de flujo, nivel computable o de comprobación de consistencia, y nivel de implementación o integración de la guía en los sistemas de información pertinentes.

Para la ayuda a la toma de decisiones en los nodos de decisión de un flujo de trabajo, hemos decidido utilizar métodos de Ayuda a la Decisión Multicriterio (MCDA). Debido a que no disponemos de datos para la evaluación de clasificadores para la gestión de procesos hemos elegido, inicialmente, PROAFTN por su poder clasificatorio en tareas de diagnóstico médico. En la comparativa presentada por Belacel [2], el PROAFTN se muestra como el mejor clasificador entre técnicas como árboles de decisión, reglas de producción, K-NN, regresión logística y perceptrones multicapa.

El método de MCDA, PROAFTN, utiliza el sistema relacional descrito en Roy [20] y Vincke [26]. Este algoritmo evita calcular repetidamente las distancias y los problemas que pueden ofrecer los datos representados en diferentes unidades. Se basa en principios de concordancia y no discordancia, por lo que se engloba dentro de los algoritmos no totalmente compensados, utiliza aprendizaje tanto inductivo como deductivo y permite utilizar criterios cuantitativos y cualitativos.

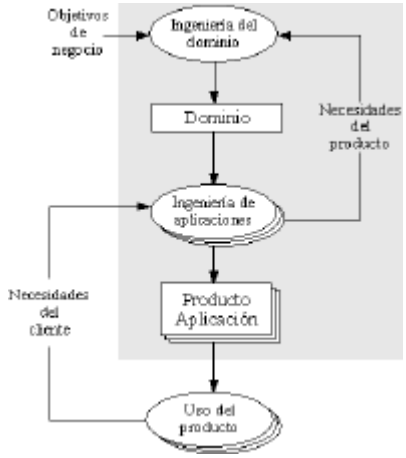


Fig. 1. Desarrollo basado en el dominio.

En la construcción del DSS utilizamos como método de reutilización sistemática el denominado Desarrollo Basado en el Dominio (Fig. 1) construyendo módulos Web para las diversas funcionalidades a ofrecer. Hemos utilizado la especificación J2EE de la tecnología Java, Protégé-2000 v1.7[?] como entorno de desarrollo de ontologías por el buen nivel de funcionamiento demostrado en múltiples aplicaciones [23], JESS v6.1[?] como motor de inferencia por su amplio uso, incluyendo la extensión para lógica difusa FuzzyJESS[?], y JessTab[?] como puente de comunicación entre Protégé-2000 y JESS. En cuanto a metodologías de desarrollo, hemos usado RUP [13] para el desarrollo de software, CommonKADS [22] para el desarrollo de sistemas basados en el conocimiento y METHONTOLOGY [7] para la definición de ontologías.

[?] Stanford Medical Informatics. http://protege.stanford.edu.

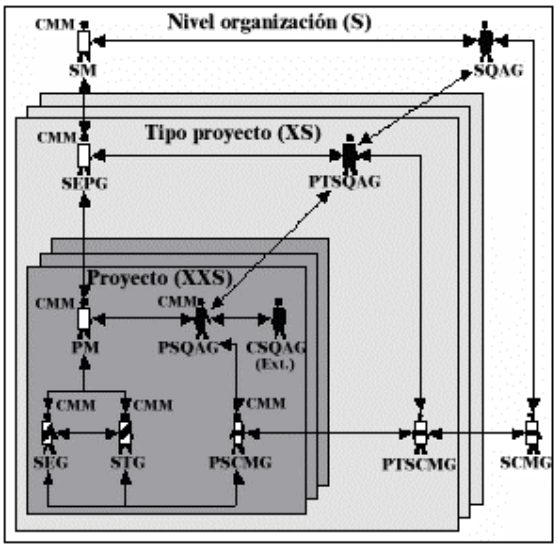
[?] Sandia National Laboratories. http://herzberg.ca.sandia.gov/jess/

[?] NRC-CNRC, IIT-ITI. http://www.iit.nrc.ca/IR_public/fuzzy/fuzzyJToolkit2.html

4 Resultados

El MEODS se compone de dos partes: el modelo organizativo (relaciones entre roles) y el de desarrollo (PDS para la elaboración de proyectos informáticos).

El modelo organizativo (Fig. 2 y 3) tiene tres niveles: nivel de proyecto (organizaciones XXS), nivel de tipo de proyecto (organizaciones XS) y nivel de organización (organización S). Los roles marcados con ^{CMM} son aquellos que corresponden directamente a su homónimo en SW-CMM. Los roles SQAG y SCMG tienen representantes en los tres niveles organizativos, estos se diferencian añadiendo los prefijos P (nivel de proyecto) y PT (nivel tipo de proyecto) a los nombres de los roles.



Gestión	
SM	Senior Manager
SEPG	Software Engineering Process Group
PM	Project Manager
Calidad	
SQAG	Software Quality Assurance Group
PTSQAG	Project Type SQAG
PSQAG	Project SQAG
CSQAG	Customer SQAG
Configuración	
SCMG	Software Configuration Management Group
PTSCMG	Project Type SCMG
PSCMG	Project SCMG
Desarrollo	
SEG	Software Engineering Group
STG	System Test Group

Fig. 2. Jerarquía de roles y niveles de organización

El nivel organizativo aumenta con el tamaño de la organización, pero el cambio de nivel es más cualitativo que cuantitativo. Una organización muy pequeña que sólo puede llevar un proyecto simultáneamente se considera XXS. Cuando esta organización adquiera la capacidad de llevar varios proyectos simultáneos siguiendo un proceso definido, el nivel es XS. Y cuando sea capaz de distinguir los proyectos que lleva en tipos de proyecto y gestionar estos tipos como líneas de desarrollo de la organización (diferentes procesos o ajustando el mismo proceso según las necesidades de cada proyecto) alcanza el nivel S.

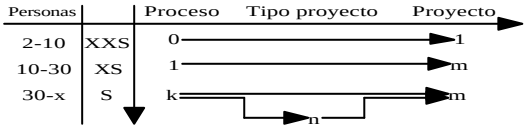


Fig. 3. Capacidad de la organización por nivel

La Fig. 4 muestra las relaciones entre las disciplinas del PDS que proponemos. El nombre y procedencia de estas disciplinas están recogidos en la Tabla 2.

El proceso comienza con la petición de propuesta (RFP) y finaliza con la oferta que evaluará el cliente. Esta propuesta tiene un nivel de detalle suficiente como para que el sistema informático que define puede ser construido, incluso, por terceros.

A medida que la organización evoluciona a través de los diferentes niveles organizativos, el PDS sufre variaciones en el conjunto de métricas, la comunicación entre roles de niveles organizativos diferentes, y la aparición de sub-artefactos específicos producto de la mayor complejidad de los proyectos y el mayor grado de detalle necesario en el conjunto inicial de artefactos (Tabla 3).

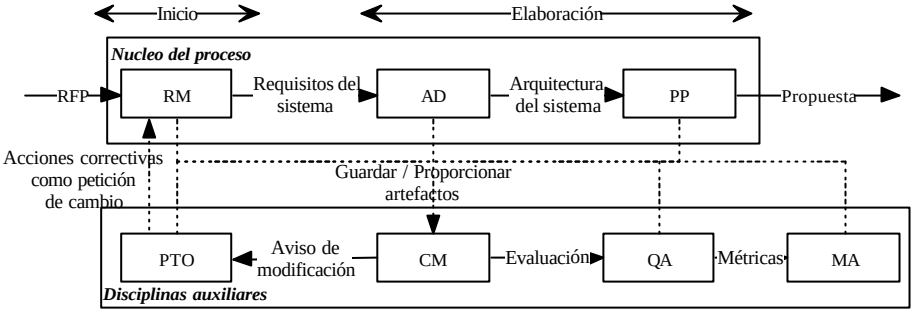


Fig. 4. Relación entre disciplinas del proceso

Table 2. Procedencia de las disciplinas

		SW-CMM	*CMMI*	*RUP*
Gestión de requisitos	RM	?	?	?
Análisis y diseño	AD			?
Planificación de proyectos	PP	?	?	?
Seguimiento y supervisión de proyectos	PTO	?	?	
Gestión de configuración	CM	?	?	?
Aseguramiento de calidad	QA	?	?	
Mediciones y análisis	MA		?	

El núcleo del proceso recoge las disciplinas que están directamente relacionadas con la obtención de una propuesta a partir de la RFP. Estas disciplinas se ejecutan en dos fases (Fig. 4): la de inicio que se encarga de capturar todos los requisitos del sistema y la de elaboración que se encarga de desarrollar la propuesta a partir de los requisitos capturados.

Las disciplinas auxiliares son las que ofrecen integridad de configuración y aseguramiento de la calidad a los artefactos, y la gestión al proyecto.

La Tabla 4 recoge los subprocesos por disciplina junto con los roles responsables de las mismas y los artefactos de entrada y salida.

Table 3. Cambios en las disciplinas por nivel organizativo

			XXS	*XS*	*S*
NÚCLEO			No cambia		
DISCIPLINAS AUXILIARES	PTO		No cambia		
	CM	Crear entorno	No cambia		
		Gestión de artefactos y evaluación de entorno	Solo se tiene en cuenta el ámbito del proyecto	Hay que tener en cuenta los artefactos públicos de los demás proyectos	Hay que tener en cuenta los artefactos públicos de los demás tipos de proyecto
	QA	Evaluación	No cambia		

		Rechazo de evaluación	El PM puede rechazar la evaluación del PSQAG	PSQAG puede pedir una segunda evaluación al PTSQAG. El SEPG puede rechazarla.	El SQAG puede realizar una última evaluación. El SM puede rechazarla
	MA		Cuanto mayor es el nivel organizativo mayor es el número de métricas y mayor la complejidad de análisis		

Tabla 4. Subprocesos por disciplina

				Rol	*Artefacto de entrada*	*Artefacto de salida*
NÚCLEO	Inicio	Gestión de requisitos	Desarrollar visión	PM	- RFP	- Visión
			Gestionar peticiones de cambio	PM	-Petición de cambio	-
	Elaboración	Análisis y diseño	Desarrollar arquitectura del sistema	SEG	-Visión	- Modelo de datos - Modelo de análisis - Modelo de diseño
		Planificación de proyectos	Obtener Plan de Desarrollo de Software	PM SEG	-Visión -Modelo de datos -Modelo de análisis -Modelo de diseño	- Propuesta
DISCIPLINAS AUXILIARES		Gestión de configuración	Establecer entorno	PSCMG	-	-
			Gestionar peticiones de artefactos	PSCMG	- Pet. artefacto	- Artefacto - Pet. Artefacto (mod)
			Gestionar almacenamiento de artefactos	PSCMG	- Pet. Archivar. - Pet. artefacto - Artefacto	-
			Evaluar estado de entorno	PSCMG	- Catálogo de artefactos	- Catálogo de artefactos
		Aseguramiento de calidad	Evaluar calidad	PSQAG	- Artefacto	- Artefacto (aceptado)
		Mediciones y análisis	Establecer métricas	PM	-	- Métricas
			Obtener mediciones	PM	- Artefacto	- Mediciones
			Analizar mediciones	PM	- Mediciones	- Evaluación
		Seguimiento y supervisión de proyectos	Comparar contra el plan el estado actual	PM	- Plan - Estado del proyecto	- Acciones correctoras

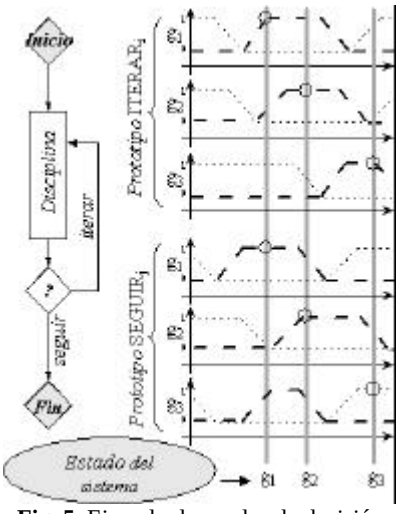


Fig. 5. Ejemplo de ayuda a la decisión

GLIF define guías (*Guidelines*) que se ejecutan siguiendo algoritmos (*Algorithm*), que se componen de pasos (*Guideline steps*) que se pueden descomponer en subguías. El núcleo de SPOnt (Fig.6) esta definido sobre 4 conceptos: proceso, tarea, rol y artefacto.

Hemos definido *proceso* como una guía y *tarea* como un *paso.* Los roles y los artefactos son conceptos propios de SPOnt y no tienen reflejo en el ámbito de GLIF. Las *tareas,* realizadas por un *rol* concreto, utilizan y producen *artefactos* dentro del proceso.

Las ontologías de Protégé-2000 son ejecutadas con JESS mediante su *plug-in JessTab*.

Los nodos decisión, al tener que resolver un problema (*Problem*) de clasificación, llevan implícita la ejecución de un proceso que implementa un método MCDA; en éste caso, PROAFTN. Como puede verse en la Fig. 5, las clases PROAFTN son las opciones de pasar a otra disciplina (*seguir)* o repetirla (*iterar)*. Para cada una, definimos prototipos sobre varios atributos del estado del sistema. El *clasificable* es el estado del sistema en el momento de decidir. En este caso el prototipo *seguir* es vetado por g_3, por lo tanto el estado actual quedaría asignado a la clase *iterar.*

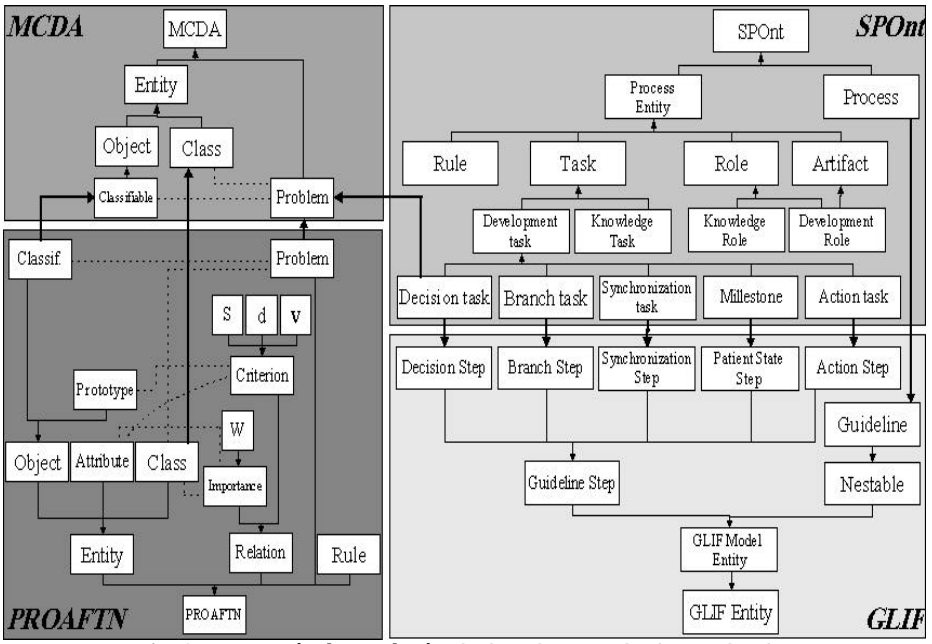


Fig. 6. Integración de ontologías SPOnt, GLIF, MCDA y PROMAF

El prototipo de DSS (Fig. 6) tiene la misma arquitectura que el DSS Arnasa [25], su funcionalidad esta repartida entre diversos módulos Web especializados que, además de la ayuda a la decisión, cubren aspectos como la visualización interactiva 2D y 3D, análisis de datos, control de acceso basado en roles, gestión de planes, comunicación entre roles para el trabajo en grupo, o la internacionalización de la interfaz.

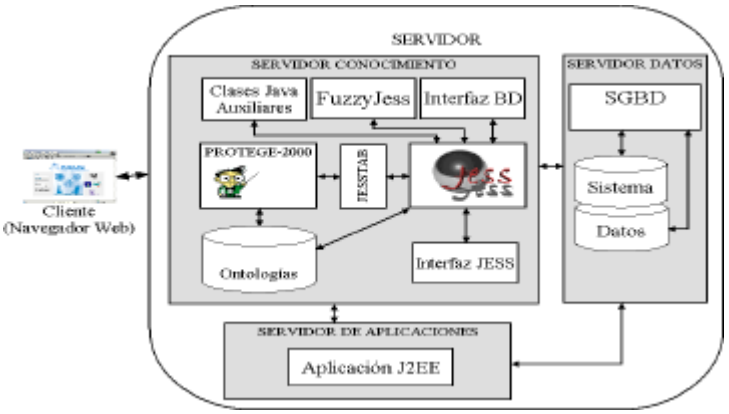


Fig. 7. Arquitectura del DSS

4 Conclusiones

El presenta trabajo a producido un modelo evolutivo de organización y desarrollo de software para organizaciones de pequeño tamaño, adaptándose a su crecimiento. Además, siendo los DSS el método más recomendado para la implantación de procesos, hemos definido una arquitectura de DSS accesible vía Web que integra el modelo propuesto con un sistema de gestión de datos de proceso y métodos de ayuda a la decisión multicriterio. Este método es aplicable en otros dominios, prueba de ello es el DSS Arnasa que implanta guías médicas en el dominio del tratamiento del asma.

El PDS desarrollado sólo cubre la elaboración del proyecto software, por lo tanto, es necesario realizar un trabajo similar para la definición de un PDS aplicable a la construcción del proyecto elaborado que, partiendo de una propuesta, llegue a construir y desplegar el producto objeto del proyecto.

En relación con trabajos previos, debemos señalar las siguientes concordancias: 1) Los niveles del modelo organizativo, son similares a los del modelo definido por Rautiainen. 2) Las relaciones entre los roles existentes son similares a las del modelo D-CMM que incluye algunos de los roles correspondientes al nivel 2 de SW-CMM. 3) Las disciplinas del PDS, se corresponden con las KPAs del nivel 2 de CMMI y a las disciplinas de RUP que consideramos pertenecen a la elaboración de proyectos informáticos y que no se correspondan con KPAs existentes en CMMI. En cuanto a la construcción del DSS, hemos utilizado la arquitectura del DSS Arnasa.

Por la experiencia con Arnasa, podemos decir que la gestión de los datos de evolución del proceso es una ventaja suficiente como para que un usuario acepte un DSS en su entorno de trabajo por la ayuda a la toma de decisiones que su monitorización supone. Además, los usuarios médicos confían en las recomendaciones del sistema si están basadas en guías de amplio consenso. Algo parecido se puede esperar en el caso de nuestro PDS una vez alcance un amplio consenso.

La eficiencia del PDS definido no ha sido probada en su globalidad debido a la falta de un PDS de construcción para complementarlo e iniciar la implantación del mismo. Sin embargo, la utilización de un proceso definido, como RUP, en nueve proyectos de fin de carrera ha demostrado ventajas evidentes en cuanto al control de los proyectos y la calidad de sus resultados. De esta experiencia cabe esperar que tras la implantación del PDS mediante el DSS, ofrecerá ventajas adicionales en cuanto a la monitorización y la ayuda a la decisión para el control por la disposición de todos los datos del proyecto, actual carencia principal en la investigación en ingeniería del software. Los módulos Web de la arquitectura de DSS propuesta ofrecen una efectividad suficiente, habiendo sido probados en las sucesivas versiones de Arnasa.

Agradecimientos. Este trabajo ha sido desarrollado con financiación recibida del Departamento de Economía y Turismo de la Diputación Foral de Guipúzcoa [OF-151/2002? bajo el patrocinio de la Unión Europea y, parcialmente, por la recibida del Ministerio de Ciencia y Tecnología [MCYT – TIC2001-1143-C03-01].

Referencias

1. Anton A.I et al: Tailored CMM for a Small e-Commerce Company Level 2: Repeatable. North Carolina State University Department of Computer Science TR-2001-09 (2001)
2. Belacel N. et al: Multicriteria Fuzzy Assignment Method: a Useful Tool to Assist Medical Diagnosis. Artificial Intelligence in Medicine 21 (2001).
3. Bicego A. et al: 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 (1998)
4. Breton E. et al: Weaving Definition and Execution Aspects of Process Meta-Models. In Proceedings of 35th Hawaii International Conference on System Sciences (2002)
5. Business Process Management Initiative (BPMI.org): Business Process Modelling Notation Working Draft (0.9) (2002)
6. CMMI Product Team. Capability Maturity Model Integration: CMMI-SE/SW/IPPD/SS V1.1 Staged Representation. Software Engineering Institute, CMU/SEI-2002-TR012 (2002)
7. Fernandez M. et al: METHONTOLOGY: From Ontological Art toward Ontological Engineering. In Proc. of AAAI-97 Spring Symposium on Ontological Engineering (1997) 33-40
8. Garcia S: Evolving Improvement Paradigms: Capability Maturity Models® ISO/IEC 15504 (PDTR). In Wiley / Gaulthier-Villars (eds): Software Process Improvement and Practice, volume 3, issue 1 (1998).
9. Harel D: STATEMATE: Working Environment for the Development of Complex Reactive Systems. In IEEE Transactions on Software Engineering (1990)
10. Humphrey W.S: The Personal Software Process (PSP). Software Engineering Institute, CMU/SEI-2000-TR-022 ESC-TR-2000-022 (2000)
11. Humphrey W.S: The Team Software Process (TSP) Software Engineering Institute, CMU/SEI-2000-TR-023 ESC-TR-2000-023 (2000)

12. IDEF: Integration Definition for Function Modelling (IDEF0). National Institute of Standards and Technology (1993)
13. Kruchten P: The Rational Unified Process. An Introduction, 2nd ed. In Booch, the Addison-Wesley Object Technology Series, Mass. (2000)
14. Laryd A. et al: Dynamic CMM for Small Organizations. In proceedings of ASSE 2000, the First Ar-gentine Symposium on Software Engineering, Tandil, Argentina (2000)
15. Lobsitz R.M: A Method for Assembling a Project-Specific Software Process Definition. In Proceedings of 29th Hawaii International Conference on System Sciences (1996)
16. Paulk M.C. et al: Key Practices of Capability Maturity Model, Version 1.1. Software Engineering Institute, CMU/SEI-93-TR-25, DTIC Number ADA263432 (1993)
17. Peleg M. et al: Comparing Computer-Interpretable Guideline Models: A Case Study Approach. In proceedings of JAMIA, vol. 10, No. 1, Jan-Feb (2003), 52-68
18. Rautiainen K. et al: A tentative Framework for Managing Software Product Development in Small Companies. In Proc. of 35th Hawaii Int. Conference on System Sciences (2002)
19. van der Aalst WMP: The Application of Petri Nets to Workflow Management. In The Journal of Circuits, Systems and Computers, vol 8, no. 1 (1998) 21-66 http://citeseer.nj.nec.com/vanderaalst98application.html
20. Roy B: Multicriteria methodology for decision aiding. In Kluwer Academic (1996)
21. Rus I. et al: Knowledge management in software engineering. In IEEE Software, vol. 19, no. 3 (2002) 26-38
22. Schreiber G. et al: Knowledge Engineering and Management: the CommonKADS Methodology. Bradford Book, MIT press, (1999)
23. Schreiber G. et al: A Case Study in Using Protégé-2000 as a tool for CommonKADS. In Proc. of 12th Int. Conf. on Knowledge Engineering and Knowledge Management (2001).
24. Schultz D. et al: A Matrix Approach to Software Process Definition. Software Engineering Workshop Greenbelt, MD (2001).
25. Sobrado F.J. et al: Arnasa: una forma de desarrollo basado en el dominio en la construcción de un DSS para la gestión del proceso de tratamiento del asma vía Web. In Proc. of JISBD (2002).
27. Vincke Ph: Multicriteria decision aid. In J. Wiley, New York (1992).