Organiza

UNIVERSIDAD EAFIT
Abierta al mundo

Apoya

CLEI

XII Conferencia Iberoamericana de Ingeniería de Requisitos y Ambientes de Software

IDEAS

2009

IDEAS 2009

Memorias

XII Conferencia Iberoamericana de Ingeniería de Requisitos y Ambientes de Software

Editores
Antonio Brogi, João Araújo, Raquel Anaya.

Medellín, Colombia
Abril 13 - 17, 2009

Memorias

XII Conferencia Iberoamericana de Ingeniería de Requisitos y Ambientes de Software

Ficha Técnica
Memorias de la XII Conferencia Iberoamericana de Ingeniería
de Requisitos y Ambientes de Software (IDEAS '09)
Editores: Antonio Brogi, João Araújo, Raquel Anaya.
Abril, 2009 - Medellín, Colombia

ISBN: 978-958-44-5028-9

PRESIDENCIA DEL COMITÉ ORGANIZADOR

Raquel Anaya
Universidad EAFIT, Colombia

PRESIDENCIA DEL COMITÉ DE PROGRAMA

João Araújo
Universidade Nova de Lisboa, Portugal

Antonio Brogi
Università di Pisa, Italy

MIEMBROS DEL COMITÉ CIENTÍFICO

Alejandra Cechich
Alessandro García
Álvaro Arenas
Amador Duran
Antonio Brogi
Antonio Vallecillo
Carla Reis
Carla Silva
Claudia Pons
César Acuña
Coral Calero
Dan Hirsch
Daniel Riesco
Daniela Godoy
Demetrio Ovalle
Elena Navarro
Ernest Teniente
Ernesto Pimentel
Fernanda Alencar
Francisco Pinheiro
Francisco Ruíz
Gaston Mousques
Genoveva Vargas
Guilherme Travassos
Gustavo Rossi
Hernan Melgratti
Isabel Díaz
Isabel Brito
Jaelson Castro
Jaime Muñoz
Jesús García Molina
João Araújo
João Falcão e Cunha
Jonás Montilva
Jorge Triñanes
José Pow-Sang
José Maldonado
Juan Carlos Trujillo
Juan Hernández
Júlio Leite
Luca Cernuzzi
Luis Guerrero
Luis Olsina
Lyrene Silva
Marcello Visconti
Márcio Delamaro
Márcio Barros
María Lencastre
Miguel Katrib
Oscar Dieste
Oscar Pastor
Rafael Calvo
Raquel Anaya
Regina Braga
Renata Guizzardi
Ricardo Falbo
Ruby Casallas
Sandra Fabbri
Silvia Gordillo
Vicente Pelechano
Victor Santander
Xavier Franch

ORGANIZACIÓN LOCAL

Alberto Restrepo
Mónica Henao
Lucas Macías Franco
Isabel Morales

PREFACIO

Bienvenidos a la décimo segunda versión de la Conferencia de Ingeniería de Requisitos y Ambientes Software (IDEAS 2009) que va ser realizado en Medellín Colombia y es organizado por el Departamento de Informática y Sistemas de la Escuela de Ingeniería de la Universidad EAFIT, del 13 al 17 de abril del 2009.

Desde su primera edición en 1.998, IDEAS fue concebido como un espacio para estimular y facilitar el intercambio de conocimiento y experiencias y para orientar las relaciones entre grupos de investigación iberoamericanos que trabajan en diversas áreas de la Ingeniería de Software. IDEAS provee un foro que permite que investigadores, educadores y profesionales presenten y discutan los desarrollos más recientes en ingeniería de software.

El primer evento de IDEAS fue realizado en 1.998 en Torres, Brasil, como un workshop. Desde entonces, el evento se ha realizado de manera exitosa en diversos paises de Latinoamérica: San José-Costa Rica (IDEAS'99), Cancún-México (IDEAS'00), Heredia-Costa Rica (IDEAS'01), La Habana-Cuba (IDEAS'02), Asunción-Paraguay (IDEAS'03), Arequipa-Perú (IDEAS'04), Valparaíso-Chile (IDEAS'05), La Plata-Argentina (IDEAS'06), Isla de Margarita-Venezuela (IDEAS'07), Recife-Brasil (IDEAS'08), y Medellín-Colombia (IDEAS'09). Vale la pena destacar que este año se aprueba oficialmente el cambio de nombre del evento de Workshop a Conferencia, teniendo en cuenta su evolución en número de trabajos presentados y participantes inscritos.

La agenda académica de IDEAS'09 cuenta con tres conferencias plenarias, dos mesas redondas, cuatro tutoriales y la presentación de los trabajos aceptados. Los tres conferencistas invitados son Jorge Villalobos (Universidad de los Andes, Colombia) quien discutirá las tendencias recientes y retos en el desarrollo de arquitecturas orientadas a servicios, Guilherme Travassos (Universidad Federal de Río de Janeiro, Brasil) quien presentará el estado de la ingeniería de software experimental, y Ernesto Pimentel (Universidad de Málaga, España) quien analizará la aplicación de los métodos formales para coordinar y adaptar servicios y componentes. Los dos paneles estarán orientados a abrir espacios de discusión alrededor de las iniciativas de la academia para responder a las demandas del mercado laboral y el papel de la industria de software latinoamericana en el mercado mundial, respectivamente. La conferencia estará precedida por dos días de tutoriales que estarán orientados a los temas de la ingeniería de requisitos orientada a aspectos, proyectos de desarrollo centrados en la arquitectura, modelado de sistemas multi-agente y calidad de evaluación de aplicaciones Web 2.0, respectivamente.

IDEAS siempre ha recibido artículos en español, portugués e inglés. Para esta edición hemos recibido un total de 82 trabajos de 18 países distintos. Cada trabajo

fue revisado por al menos tres miembros del Comité de Programa. Después de un riguroso proceso de revisión, fueron aceptados 19 artículos completos y 19 artículos cortos.

El trabajo del Comité de Programa y de los revisores adicionales que colaboraron en el proceso de evaluación de artículos es sobresaliente. Todos los autores recibieron comentarios detallados de los evaluadores. Agradecemos a todos los revisores por su excelente trabajo y agradecemos también a todos los autores que enviaron sus trabajos a la conferencia. Agradecemos a la Universidad EAFIT por el patrocinio de IDEAS'09, así mismo al Comité de Organización local que hizo posible la realización de esta conferencia.

Finalmente, extendemos una cordial bienvenida a conferencistas, autores, estudiantes y profesionales que nos acompañaran en IDEAS'09. Esperamos que puedan disfrutar del evento y además tengan la oportunidad de disfrutar de la cultura de Medellín y de la amabilidad de su gente.

Antonio Brogi
João Araújo
Raquel Anaya

Abril 2009

PREFÁCIO

Bem-vindos à 12ª Conferência Ibero-americana em Engenharia de Requisitos e Ambientes de Software (IDEAS 2009) que tem lugar em Medellín, Colômbia, organizada pelo Departamento de Informática e Sistemas, Escola de Engenharia da Universidade EAFIT, de 13 a 17 de Abril de 2009.

Desde a sua primeira edição em 1998, IDEAS foi concebida para estimular e facilitar o intercâmbio de conhecimento e de experiências, além de estreitar as relações entre grupos de pesquisa ibero-americanos trabalhando em diversas áreas da Engenharia de Software. IDEAS proporciona um fórum que tem como objetivo permitir que investigadores, educadores e profissionais apresentem e discutam os mais recentes desenvolvimentos em Engenharia de Software.

O primeiro evento de IDEAS teve lugar em 1998 em Torres, Brasil, como um workshop. Desde então, o evento foi realizado com sucesso em San Jose-Costa Rica (IDEAS'99), Cancun-México (IDEAS'00), Herédia-Costa Rica (IDEAS'01), La Habana-Cuba (IDEAS'02), Asuncion-Paraguai (IDEAS'03), Arequipa-Peru (IDEAS'04), Valparaiso-Chile (IDEAS'05), La Plata-Argentina (IDEAS'06), Isla de Margarita-Venezuela (IDEAS'07), Recife-Brasil (IDEAS'08), e Medellin, Colômbia (IDEAS'09). Vale a pena salientar que este ano o Workshop evoluiu para Conferência, uma vez que o seu tamanho, em termos de submissões e participantes, justifica esta promoção.

IDEAS'09 inclui três palestras convidadas, dois painéis de discussão, quarto tutoriais e as apresentações dos artigos. Os três palestrantes convidados são Jorge Villalobos (Universidade de Los Andes, Colômbia) que discutirá as tendências em arquiteturas orientadas a serviços, Guilherme Travassos (Universidade Federal do Rio de Janeiro, Brasil) que discutirá os desafios em engenharia de software experimental, e Ernesto Pimentel (Universidade de Málaga, Espanha) que discutirá a aplicação de métodos formais para coordenar e adaptar serviços e componentes. Os dois painéis serão voltados ao analise das iniciativas do mundo acadêmico para responder às demandas do mercado de trabalho, e discutir o papel da indústria de software latino-americana no mercado mundial, respectivamente. A conferência será precedida por dois dias de tutoriais, que enfocarão engenharia de requisitos orientada a aspectos, projetos orientados a arquiteturas, modelação de sistemas multi-agentes, e avaliação de qualidade de aplicações Web 2.0.

IDEAS sempre recebeu artigos em espanhol, português ou inglês, a fim de fomentar a interação entre pesquisadores de diferentes países ibero-americanos. Para esta edição de IDEAS, recebemos um total de 82 submissões de 18 países distintos. Cada artigo foi revisto por pelo menos três membros do Comitê de Programa. Depois de um processo de avaliação rigoroso, 19 artigos foram aceites como artigos completos

e 19 foram aceites como artigos curtos.

O trabalho do Comitê de Programa e dos outros avaliadores que colaboraram no processo de avaliação foi de altíssimo nível. Todos os autores receberam comentários detalhados dos avaliadores. Nós gostaríamos de agradecer a todos os avaliadores pelo seu excelente trabalho. Gostaríamos também de agradecer a todos os autores por submeter as suas valiosas contribuições.
Agradecemos a Universidade EAFIT pelo patrocínio de IDEAS'09, assim como o Comitê de Organização local que tornou esta conferência possível.

Finalmente, desejamos que todos os conferencistas, autores, estudantes e profissionais que nos acompanharão em IDEAS'09 sejam muito bem-vindos. Esperamos que possam desfrutar do evento e que tenham a oportunidade de conhecer a cultura de Medellín e da amabilidade de sua gente.

Antonio Brogi
João Araújo
Raquel Anaya

Abril 2009

PREFACE

Welcome to the 12th Ibero-american Conference on Requirements Engineering and Software Environments (IDEAS 2009) to be held in Medellín, Colombia, which is organised by the Department of Informatics and Systems, of Eafit University's Engineering School, from April 13 to 17, 2009.

Since its first edition in 1998, IDEAS was conceived as a space to stimulate and facilitate the exchange of knowledge and experiences, and to direct the relations among Ibero-american research groups working in diverse areas of Software Engineering. IDEAS provides a forum that allows that researchers, educators and professionals present and discuss the most recent developments in software engineering.

The first IDEAS event was held in 1998 in Torres, Brazil, as a workshop. Since then, the event has successfully taken place in San Jose - Costa Rica (IDEAS-99), Cancun-Mexico (IDEAS -00), Heredia - Costa Rica (IDEAS-01), La Habana-Cuba (IDEAS-02), Asuncion-Paraguay (IDEAS-03), Arequipa-Peru (IDEAS-04), Valparaiso-Chile (IDEAS-05), La Plata-Argentina (IDEAS-06), Isla Margarita-Venezuela (IDEAS-07), Recife-Brazil (IDEAS-08), and Medellín, Colombia (IDEAS-09). It is worth pointing out that this year a change of the name of the event is going to be approved from "Workshop" to "Conference", having into count its evolution in terms of the number of papers presented, and the number of signed participants.

IDEAS-09 features three plenary sessions, two panels, four tutorials, and the presentations of contributed papers. The three invited speakers are Jorge Villalobos (Los Andes University, Colombia) who will discuss recent trends and challenges in developing service-oriented architectures; Guilherme Travassos (Federal University of Rio de Janeiro, Brazil) who will talk about the current state of experimental software engineering; and Ernesto Pimentel (University of Malaga, Spain) who will analyze the application of formal methods to coordinate and adapt services and components. The two panels will be devoted to open discussion spaces around the initiatives of the academic world to respond to the demand from the labour market, and the role of the Latin-American software industry in the world market, respectively. The conference will be preceded by two days of tutorials, which will be oriented towards requirements engineering topics, particularly to architecture-driven projects, multi-agent systems modelling, and quality evaluation of web 2.0 applications.

IDEAS has always welcomed articles in Portuguese, Spanish, and English. For this edition of IDEAS, we received a total of 82 submissions from 18 different countries. Each paper was reviewed by at least three members of the Program Committee. After a rigorous reviewing process, 19 papers were accepted as full papers and 19 were accepted as short papers.

The work of the Program Committee and that of the extra reviewers who collaborated in the paper evaluation process was outstanding. All authors received detailed comments from the referees. We would like to thank all reviewers for their great job. We would also like to thank all authors for submitting valuable contributions. We would like to thank EAFIT University for sponsoring IDEAS-09, as well as the local Organizing Committee that made it possible to run this conference.

Finally, we cordially welcome all the conference participants, authors, students and professionals that will join IDEAS-09. We do hope that you will enjoy the event and will also have the chance to experience the culture of Medellin and its people's hospitality.

Antonio Brogi
João Araújo
Raquel Anaya

April 2009

TABLA DE CONTENIDO

SESIÓN 3. DESARROLLO DIRIGIDO POR MODELOS (SHORT PAPERS)

SESIÓN 4. MEJORA DEL PROCESO SOFTWARE

SESIÓN 5. ASPECTOS Y REQUISITOS

SESIÓN 6. MEJORA DEL PROCESO SOFTWARE (SHORT PAPERS)

CHARLAS INVITADAS

Apéndice I. Plantilla Plan Preliminar de Mejora

***Mejora de Procesos para Fomentar la Competitividad de la Pequeña y Mediana Industria del Software de Iberoamérica* - Proyecto COMPETISOFT (Financiado por CYTED)**
Plan Preliminar de Mejora. Se planifica de manera general las iteraciones a realizar para mejorar los procesos de la organización.

Ciclo de mejora	
Nombre de la empresa	
Nombre del proyecto de mejora	
Nombre del responsable de la empresa	
Nombre del responsable de COMPETISOFT	

Nivel de capacidad actual y esperado de los procesos a mejorar en este Ciclo		
Procesos a mejorar	*Nivel de Capacidad Actual*	*Nivel de Capacidad Esperado*

Iteraciones del Ciclo de mejora	
Número de iteraciones del ciclo de mejora	

Para llevar a cabo la mejora en los procesos descritos para el primer ciclo de mejora se han definido X iteraciones, de la siguiente forma:

Iteración	Proceso	Duración	Desde
1			
...			

Planeación general de las iteraciones del Ciclo de mejora.
Plan de manejo de riesgos del Ciclo de mejora
Plan de capacitación del Ciclo de mejora
Plan de mediciones del Ciclo de mejora

Estimación del esfuerzo realizado en esta actividad

Fecha	*Actividad*	*Nombre ó Rol de las personas involucradas*	*Horario*	*Tiempo Asesor (minutos)*	*Tiempo Empresa (minutos)*
			Total separado		
			Total consolidado		

Otra información relevante (Registre aquí otra información que considere relevante)

Aprobación del Plan Preliminar de Mejora	
Firma del Responsable de Mejora de la Empresa Fdo:	Firma del asesor de COMPETISOFT Fdo:

Sugerencias de mejora a esta plantilla y la actividad asociada

SMML: Lenguaje para la Representación de Modelos de Medición del Software

Beatriz Mora, Félix García, Francisco Ruiz, Mario Piattini

Dep. de Tecnología y Sistemas de Información Escuela Superior de Informática de Ciudad Real.
Universidad de Castilla-La Mancha
{Beatriz.Mora | Felix.Garcia | Francisco.RuizG | Mario.Piattini}@uclm.es

Resumen. Los Lenguajes de Dominio Específico (DSLs) y la medición del Software son actualmente importantes campos de investigación en la Ingeniería del Software. De hecho, han llegado a ser aspectos importantes en la industria del software. Los lenguajes de dominio facilitan el proceso de desarrollo del software en un dominio específico, y la medición puede ayudar a controlar determinados errores críticos en el desarrollo y mantenimiento del software, facilitando la toma de decisiones. Este trabajo presenta un lenguaje que permite definir modelos de medición del software basados en una Ontología de Medición del Software. Los modelos están sintáctica y semánticamente bien formados a partir de un metamodelo de medición específico, que está alineado con la ontología mencionada.

Palabras clave: SMML, Medición del Software, DSLs.

1 Introducción

La medición del software se ha convertido en un aspecto fundamental de la Ingeniería del Software [9]. Está demostrando ser muy eficaz en la construcción de sistemas de predicción de alta calidad para grandes proyectos de bases de datos [22], en la comprensión y mejora de los proyectos de desarrollo y mantenimiento del software [1], en la evaluación y garantía de calidad de sistemas, evidenciando las áreas problemáticas [4], en la determinación de mejores prácticas de trabajo con el fin de ayudar a los usuarios e investigadores en su trabajo [4]. Además, las medidas software son herramientas importantes que ayudan en la evaluación y en la institucionalización de la Mejora del Proceso Software (Software Process Improvement) en las organizaciones que lo desarrollan. De hecho, la medición del software es la pieza clave de iniciativas como SW-CMM (Capability Maturity Model for Software), ISO/IEC 15504 (SPICE, Software Process Improvement and Capability dEtermination) y CMMI (Capability Maturity Model Integration). El estándar ISO/IEC 90003:2004 [17] también destaca la importancia de la medición en la gestión y garantía de la calidad.

Para llevar a cabo la medición de una forma precisa y sistemática existen distintos métodos y estándares, los más representativos son:

- Goal Question Metric (GQM): El principio básico de GQM es que la medición debe ser realizada, siempre, orientada a un objetivo. GQM define un objetivo, refina este objetivo en preguntas y define métricas que intentan dar información para responder a estas preguntas.
- Practical Software Measurement (PSM): La metodología PSM [23] se basa en la experiencia obtenida de organizaciones para saber cuál es la mejor manera de implementar un programa de medida de software con garantías de éxito.
- IEEE 1992 (Metodología para Métricas de Calidad del Software): según el estándar IEEE 1992, la calidad del software se puede considerar como el grado en el que el software posee una combinación claramente definida y deseable de atributos de calidad. Este estándar trata de definir la calidad del software para un sistema mediante una lista de atributos de calidad del software requeridos por el propio sistema.
- ISO/IEC 15939: Este estándar internacional [16] identifica las actividades y tareas necesarias para identificar, definir, seleccionar, aplicar y mejorar de manera exitosa la medición de software dentro de un proyecto general o de la estructura de medición de una empresa.

Una importante toma de decisión y mejora de proceso es la disponibilidad de un lenguaje que permita representar aquellos elementos que participan en el proceso de medición del software.

En este sentido es interesante considerar la utilización de los Lenguajes de Dominio Específico (Domain Specific Languages, DSLs). Los DSLs son una contribución de la metodología de modelado específico de dominio (Domain Specific Modeling, DSM), donde el nivel de abstracción es alto y la solución se especifica directamente con los conceptos del dominio.

Estos son los aspectos que constituyen el principal interés de este artículo, cuyo objetivo es presentar el Lenguaje de Modelado de Medición del Software (Software Modeling Measurement Language, SMML) que permite crear modelos de medición del software de una manera simple e intuitiva. Como Metamodelo de Definición de Software (Domain Definition Metamodel, DDMM) se ha utilizado el Metamodelo de Medición del Software (Software Measurement Metamodel, SMM)[14]) (para mayor detalle ver [26]). Este lenguaje pertenece al marco de trabajo Software Measurement Framework (SMF) presentado en [25]. SMF permite llevar a cabo mediciones genéricas mediante transformaciones utilizando dos modelos iniciales como punto de salida: el de medición del software y el de dominio. La tarea de SMML es facilitar la definición de modelos de medición del software, que es el punto de partida del proceso de medición del software.

El resto del artículo está organizado como sigue. En el apartado 2 se muestran los trabajos relacionados. En el apartado 3 se presenta el lenguaje SMML, junto con la definición de su sintaxis abstracta, sintaxis concreta y su semántica. Para ilustrar el uso del lenguaje se presenta un ejemplo en el apartado 4. Por último, las conclusiones y trabajo futuro se presentan en el apartado 5.

2 Trabajos Relacionados

Existen numerosos trabajos relacionados con el desarrollo de DSLs. Por un lado se pueden encontrar publicaciones que presentan metodologías, propuestas, herramientas y patrones para facilitar el desarrollo de DSLs [3, 6, 20, 24, 29, 30].

Por otro lado, existen gran diversidad de trabajos que presentan DSL, por ejemplo: ATL (ATLAS Transformation Language) [18], un lenguaje de transformaciones de modelos con un entorno de ejecución basada en Eclipse; KM3 (Kernel MetaMetaModel) [19] que es un DSL para la representación de metamodelos; RubyTL [31] que es un lenguaje de transformación híbrido definido como un lenguaje específico del dominio embebido en el lenguaje de programación Ruby, y diseñado como un lenguaje extensible en el que un mecanismo de plugins permite añadir nuevas características al núcleo de características básicas; etc.

Con respecto a DSLs de medición del Software, Guerra et al. [15] presentan un framework de creación de lenguajes visuales específicos de dominio (Domain Specific Visual Language, DSVL). En este trabajo se desarrolló como caso de estudio un lenguaje llamado SLAMMER, que forma parte de un conjunto de herramientas de gestión de modelos que Guerra et al. han definido utilizando gramática de grafos y transformaciones de grafos, en la que la evaluación y medición de artefactos software son un elemento esencial. El objetivo es facilitar la tarea de definición de mediciones y rediseños para cualquier DSVL.

Existe un metamodelo llamado Software Metrics Meta-Model [28] desarrollado por OMG. Este metamodelo propone un formato de intercambio que permite la interoperabilidad entre las herramientas existentes, servicios y sus respectivos modelos. Este formato de intercambio común se puede aplicar igualmente para el mantenimiento y desarrollo de herramientas, servicios y modelos. A pesar de la existencia de este metamodelo, se ha optado por utilizar el ya existente Metamodelo de Medición del Software [14] ya que está basado en la Ontología de Medición del Software [11] que establece y clarifica los elementos (conceptos y relaciones) involucradas en el dominio de medición del software. Hemos comprobado que utilizar una ontología tiene importantes ventajas, especialmente, dada la importancia de una base conceptual sólida que presente el dominio del problema (ontología) de cara a poder abordar el dominio de la solución (metamodelo). Las ontologías son además potencialmente útiles cuando se desarrollan DSLs durante la fase de análisis donde la captura y representación de conocimiento son elementos claves [5]. Por tanto la definición de SMML se ha basado en esta ontología.

3 Software Measurement Modeling Language (SMML)

SMML (Software Measurement Modeling Language) es un lenguaje que permite definir modelos de medición del software de una manera fácil e intuitiva. Este lenguaje forma parte del Software Measurement Framework (SMF). SMF permite llevar a cabo mediciones genéricas mediante transformaciones a partir de dos modelos de entrada: el de medición del software y el de dominio. La tarea de SMML es facilitar al usuario el modelado de los modelos de medición del software, que es entrada del proceso de medición genérica del software.

El desarrollo de DSL requiere tanto el conocimiento del dominio como la experiencia en el desarrollo del lenguaje [24], por eso se han seguido metodologías para llevar a cabo DSL usables y un correcto desarrollo del DSL.

Por un lado, se ha seguido la metodología de "desarrollo de un DSL" propuesta en [24] que consiste en la realización de cinco fases: decisión, análisis, diseño, implementación y despliegue. A continuación se explican cada una de las fases:

- **Decisión:** se ha considerado de interés desarrollar un DSL para la definición de modelos de medición ya que actualmente no existe ningún lenguaje específico con el mismo propósito. Es posible definir los modelos utilizando UML pero se considera más adecuado utilizar un lenguaje específico con una notación dirigida a los responsables de programas de medición.
- **Análisis:** en esta fase identificamos el dominio del problema y recolectamos el conocimiento del dominio. El dominio del problema es la medición del software, y para extraer el conocimiento del dominio se ha hecho uso de la ontología de medición del software [10].
- **Diseño:** para diseñar el lenguaje se ha utilizado el metamodelo de medición del software [14] que está basado en la ontología. Para definir este metamodelo se ha utilizado el metametamodelo ECORE. El metametamodelo Ecore es un lenguaje común basado en EMOF, que es parte de la especificación MOF 2.0 usado para la definición de metamodelos. Para diseñar el lenguaje se han tenido en cuenta los requisitos de calidad de un DSL presentados en [20]: conformidad, ortogonalidad, portabilidad, integración, longevidad, simplicidad, calidad, escalabilidad y usabilidad.
- **Implementación y Despliegue:** Para construir el editor gráfico del lenguaje SMML, se va a utilizar GMF de eclipse, que permite generar automáticamente código Java a partir del metamodelo del lenguaje. El resultado es un editor gráfico que permite definir el modelo de medición del software basado en el metamodelo de medición del software. Además de la representación gráfica, el editor ofrece una especificación textual basada en XML.

Por otro lado, Feilkas [8] enumera las tareas necesarias para llevar a cabo un DSL usable: definición de una sintaxis abstracta, definición de una sintaxis concreta y definición de una semántica. En los siguientes apartados se describe cómo se han llevado a cabo estas etapas para el desarrollo de SMML.

3.1 Definición de una sintaxis abstracta (Metamodelo de definición del dominio)

La sintaxis abstracta de SMML se ha obtenido a partir de la Ontología de Medición del Software (Software Measurement Ontology, SMO). Esta ontología contiene todos los elementos (conceptos y relaciones) del dominio de medición del software (para mayor detalle ver [10]).

Para el desarrollo del lenguaje interesa tener un Metamodelo de Medición del Software (Software Measurement Metamodel) que permita el modelado de instancias de Medición del Software y que contenga todos los conceptos y relaciones de SMO. Existe un metamodelo de Medición del Software (Software Measurement Metamodel, SMM) [14] que está basado en SMO. Este metamodelo es el metamodelo de dominio específico (Domain Definition Metamodel, DDMM)[21] de SMML, y contiene las entidades básicas del dominio de medición con sus relaciones. Esta base ontológica tiene un papel importante en la definición del lenguaje, en concreto en la definición de la sintaxis abstracta.

El metamodelo de medición del software incluye todas las sub-ontologías de SMO, sin embargo para el desarrollo del lenguaje se han considerado todas las sub-ontologías excepto la sub-ontología Measurement Action. Se ha excluido ésta ya que contiene los elementos relativos al resultado de la medición y no al problema del dominio. En la Fig. 1 se muestra la estructura de paquetes en la que se basa el lenguaje.

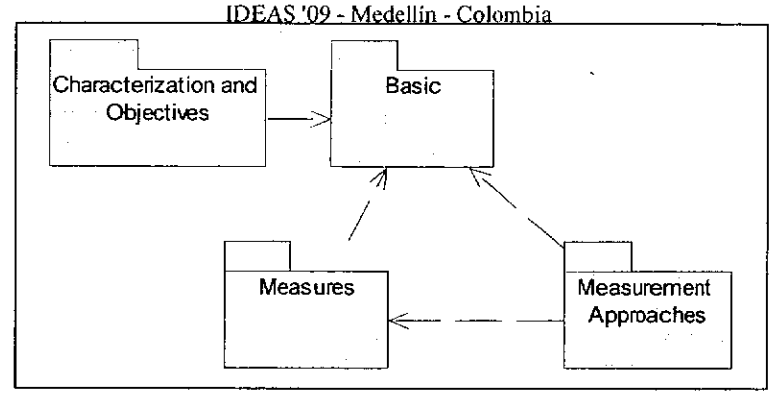


Fig. 1. Estructura de paquetes de SMML.

Tal y como se puede observar en la Fig. 1 el metamodelo está compuesto de un paquete básico que representa las características generales de los constructores básicos de modelos de medición, y de otros tres paquetes (*Characterization and Objectives, Software Measures y Measurement Approaches*), de acuerdo con las tres sub-ontologías de la SMO. Para la construcción de este metamodelo, se ha tenido en cuenta la conceptualización establecida en la Ontología de la Medición Software, pero añadiendo los constructores específicos desde el punto de vista de la implementación.

Todos los elementos que se identifican en la ontología son candidatos a ser los elementos del Metamodelo de Medición del Software en el que se basa el lenguaje SMML, en cambio las relaciones existentes en la ontología no se corresponden con las relaciones necesarias para el lenguaje. Todos los paquetes del Metamodelo de Medición mantienen su definición original [14] excepto el paquete básico, que ha tenido que ser adaptado para representar las relaciones de medición en SMML.

Hay que estudiar los tipos de relaciones que se identifican en la ontología y definirlos para el metamodelo indicando para cada relación los elementos involucrados (un origen y un destino).

Tabla 1. Selección de asociaciones del paquete *Characterization and Objectives* de SMO con las asociaciones de SMML.

Relación SMO	Icono	Descripción	Fuente
Includes	◇——	An *entity class* may include several other *entity classes*. An *entity class* may be included in several other *entity classes*.	UML Agregation
Defined for	- - - - - - - - - ->	A *quality model* is defined for a certain *entity class*. An *entity class* may have several *quality models* associated	UML Dependency
Relates	——>	A *Measurable concept* relates one or more *attributes*. An *Attribute* is related with one or more *measurable concepts*.	UML Association
Has	——	An *entity class* has one or more *attributes*. An *attribute* can only belong to one *entity class*.	UML nonnavigable Association

En [26] se muestran en detalle las relaciones del Metamodelo de Medición del Software que se corresponden con las relaciones de la ontología SMO, en total se han identificado 4

tipos de Relación de Medición (*Measurement Association*): *association, nonnavigable association, agregation* y *dependency.* Estas relaciones se han definido en el paquete básico.

En la Tabla 1 se presentan las 4 relaciones identificadas del lenguaje. Por cada relación en SMO se ha asociado una relación de medición en SMML. En la se muestra un ejemplo de 4 relaciones del paquete Characterization and Objectives de la ontología y el icono utilizado para representarlo.

A continuación se describen los paquetes que componen el metamodelo de medición del software. Puesto que se basa en la ontología que está escrita en inglés, se ha mantenido la terminología inglesa.

- **Paquete básico:** el paquete básico se ha definido con el objetivo de identificar y establecer las características generales de los constructores necesarios para definir modelos de medición. Como se ha comentado en el punto anterior, con respecto al Metamodelo de Medición del Software definido en [14] se han añadido 4 tipos de Relación de Medición (*Measurement Association*): *association, nonnavigable association, agregation* y *dependency.* La Fig. 2 muestra el diagrama de clases UML que representa la estructura de este paquete:

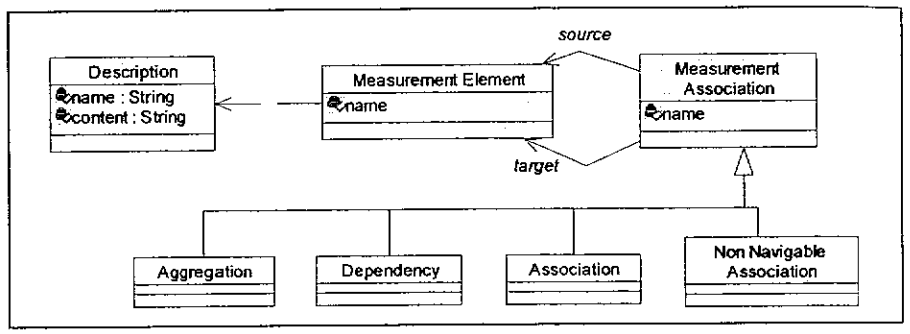


Fig. 2. Paquete Básico.

Como se puede observar en la Fig. 2, el elemento general a partir del cual se construyen modelos de medición es el constructor "*Measurement Element*", y el elemento general a partir del cual se construyen las relaciones de los modelos es el constructor "*Measurement Association*". Un elemento de medición tiene un nombre y puede ser descrito mediante elementos de tipo "*Description*", que aportan información adicional a los elementos de la medición, lo que facilita un mejor entendimiento de los modelos de medición desarrollados. A partir del elemento de medición se especializan los constructores fundamentales de la medición, obtenidos a partir de los conceptos de la Ontología de la Medición del Software. Una relación de medición relaciona dos elementos de medición, un elemento origen y otro destino. A partir de la relación de medición se especializan los constructores de las relaciones de medición definidas para el metamodelo: Asociación, Agregación y Dependencia.

- **Paquete Characterization and Objectives (Caracterización y Objetivos):** este paquete incluye los constructores necesarios para la definición de una vista fundamental del proceso de medición, basada en establecer los objetivos y en identificar y caracterizar los elementos necesarios para cumplir con dichos objetivos. La Fig. 3 muestra un diagrama UML con la estructura de este paquete.

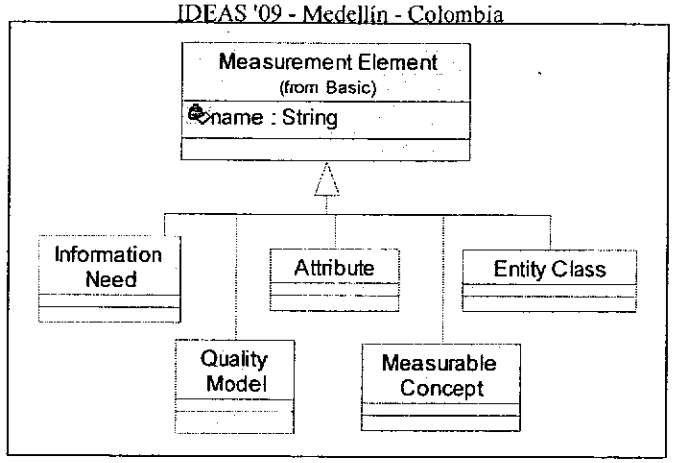


Fig. 3. Paquete Characterization and Objectives.

- **Paquete Software Measures (Medidas Software):** este paquete incluye los constructores necesarios para definir las características generales de las medidas. La Fig. 4 muestra la estructura de este paquete.

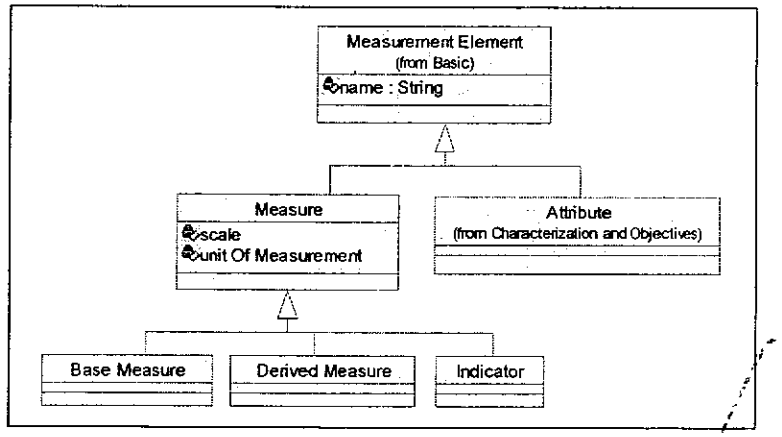


Fig. 4. Paquete Software Measures.

- **Paquete Measurement Approaches (Formas de Medir):** este paquete incluye los elementos de modelado necesarios para la representación de la measurement approaches de los diferentes tipos de medidas La Fig. 5 muestra la estructura de este paquete.

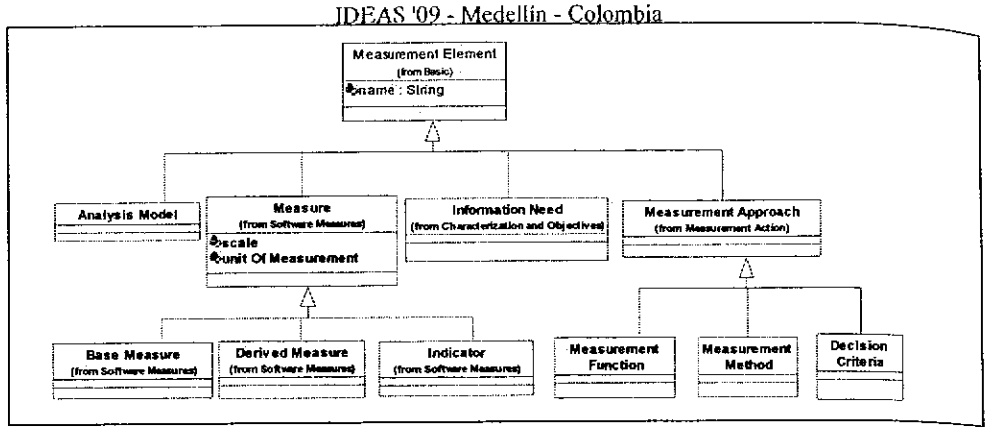


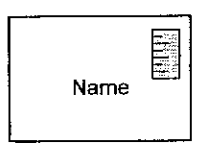
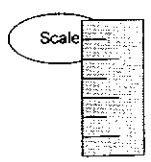
Fig. 5. Paquete Measurement Approaches.

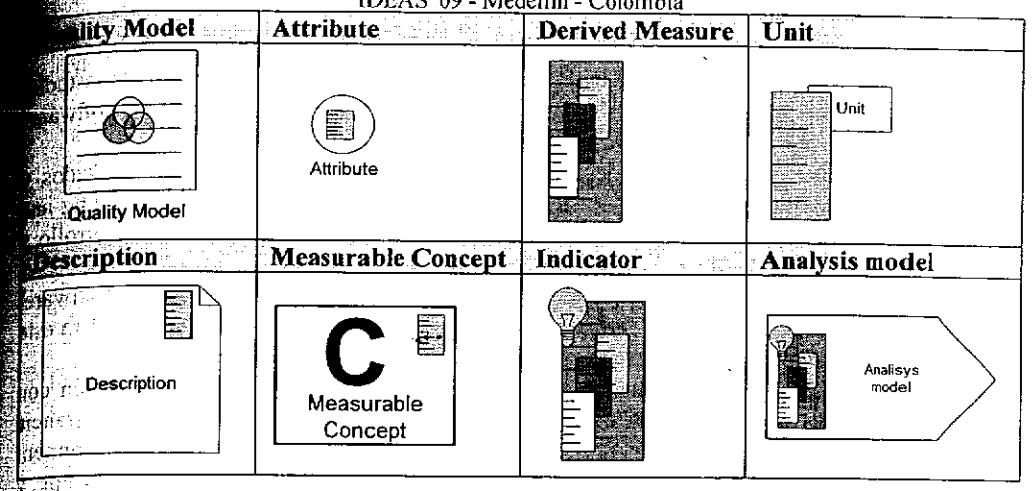
3.2 Definición de un Sintaxis concreta

Para hacer un lenguaje usable, hay que definir una sintaxis concreta. Cada uno de los elementos del lenguaje (todos definidos en el paquete básico) se deben asociar con un icono gráfico que represente el elemento del modelo abstracto. En SMML cada elemento y relación del metamodelo se han asociado con un icono representativo. El símbolo asociado con cada constructor se ha elegido con especial cuidado para que resulte lo más familiar posible a los ingenieros del software y así facilitarles su uso. Por ejemplo, el elemento *Description* es muy similar al de una *nota UML*, con la diferencia de que este añade la imagen de una regla (como símbolo de medición) en la esquina superior derecha. De manera similar, el icono *Entity class* es muy parecido al de una *Entity Class del* diagrama E/R, aunque particularizado para expresar entidades medibles.

En la siguiente tabla, se muestra un ejemplo del icono, definición y fuente de alguno de los elementos del lenguaje SMML.

Tabla 2. Selección de elementos e iconos de SMML.

Information need	Entity	Base Measure	Scale
 Information need	 Name		 Scale



3.3 Definición de la semántica

Uno de los aspectos más importantes de la especificación del lenguaje es la definición de su semántica. Para definir las semántica del lenguaje se han utilizado restricciones OCL [27] en el metamodelo. Las restricciones OCL definen la cardinalidad de las relaciones y los elementos participantes en cada relación de medición. En la Tabla 3 se muestran las restricciones utilizadas para comprobar que los elementos origen y destino (source y target) de las asociaciones son correctos.

Tabla 3. Selección de restricciones OCL de SMML.

Elemento	OCL Constraint
Nonnavigable Association	`self.source.oclIsTypeOf(EntityClass) and self.target.oclIsTypeOf(Attribute)`
Association	`self.source.oclIsTypeOf(MeasurableConcept) and self.target.oclIsTypeOf(Attribute) or self.source.oclIsTypeOf(DerivedMeasure) and self.target.oclIsTypeOf(MeasurementFunction) or self.source.oclIsTypeOf(BaseMeasure) and self.target.oclIsTypeOf(MeasurementMethod) or self.source.oclIsTypeOf(Indicator) and self.target.oclIsTypeOf(AnalysisModel)`
Agregation	`self.source.oclIsTypeOf(EntityClass) and self.target.oclIsTypeOf(EntityClass)`
Dependency	`(self.source.oclIsTypeOf(QualityModel) and self.target.oclIsTypeOf(EntityClass)) or (self.source.oclIsTypeOf(QualityModel) and self.target.oclIsTypeOf(MeasurableConcept)) or (self.source.oclIsTypeOf(MeasurableConcept) and self.target.oclIsTypeOf(InformationNeed)) or (self.source.oclIsTypeOf(AnalysisModel) and self.target.oclIsTypeOf(DecisionCriteria)) or (self.source.oclIsTypeOf(Indicator) and self.target.oclIsTypeOf(InformationNeed)) or (self.source.oclIsTypeOf(Measure) and self.target.oclIsTypeOf(Attribute))`

4 Casos de Estudio

Para ilustrar los beneficios de SMML, se consideran los siguientes casos de estudio: "desarrollo y mantenimiento de aplicaciones de bases de datos en una compañía software" y "definición de una modelos de calidad de datos para portales web".

El primer caso de estudio se detalla en [2]. Este artículo presenta los resultados y lecciones aprendidas en la aplicación de FMESP [12] (Framework for the Modeling and Measurement of Software Processes) en una compañía software dedicada al desarrollo y mantenimiento del software en sistemas de información.

Toda la información del problema se define en cada paquete de Medición del Software: Characterization and Objectives, Software Measures y Measurement Approaches. Para este caso sólo se mostrará el modelado del paquete Characterization and Objectives.

En este ejemplo, se quiere ilustrar como se representaría un modelo de medición con SMML. La Tabla 4 muestra toda la información necesaria para representar una instancia del paquete Characterization and Objectives. Los elementos de medición (Measurement Elements) utilizados son: Information Need, Quality Model, Measurable Concept y Attribute. A partir de esta información se ha definido Fig. 6 utilizando diagramas de objetos UML, y la Fig. 7 mediante SMML.

Tabla 4. Representación textual de la instancia del paquete Characterization and Objectives.

Measurement metamodel elements (MOF classes)	E/R measurement model (instantes M2)	Measurement metamodel relations (MOF associations)	E/R measurement model (links M2)	
Attribute	Size Complexity Length	Has	Entity class Attribute	Relational Schema Size Complexity Length
Entity class	Relational Schema	Defined for	Quality model Entity class	ISO 9126 Relational Schema
Measurable concept	Maintainability	Evaluates	Quality model Measurable concept	ISO 9126 Maintainability
Quality model	ISO 9126	Relates	Attribute Measurable concept	Size Complexity Length Maintainability
Information need	To know the Relational Schemes maintainability	Is associated With	Measurable Concept Information need	Maintainability To know the Relational Schemes maintainability

- **Measurement metamodel elements (MOF classes):** se corresponde con los elementos del metamodelo de medición, clases MOF.
- **E/R measurement model (instantes M2):** son las instancias del metamodelo de medición.
- **Measurement metamodel relations (MOF associations):** descripción de las asociaciones y elementos involucrados en las asociaciones.

- **E/R measurement model (links M2):** valor de cada uno de los elementos involucrados en las asociaciones.

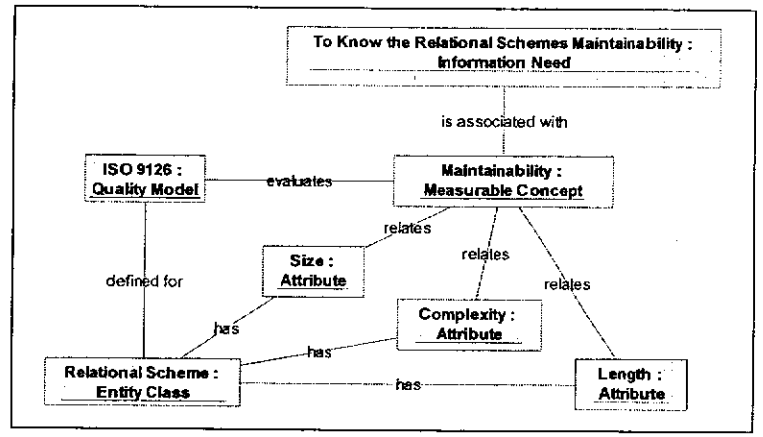


Fig. 6. Instancia del paquete Characterization and Objectives utilizando UML.

Como se puede observar en la siguiente figura (Fig. 7), la representación es más fácil e intuitiva con el lenguaje SMML que utilizando notación textual o UML. Utilizando SMML se evita la utilización de las asociaciones del metamodelo de medición ya que tiene definida asociaciones de medición para cada tipo de asociación. Además, durante la definición de modelos, no se han encontrado errores en los constructores del metamodelo, ni carencias en el lenguaje.

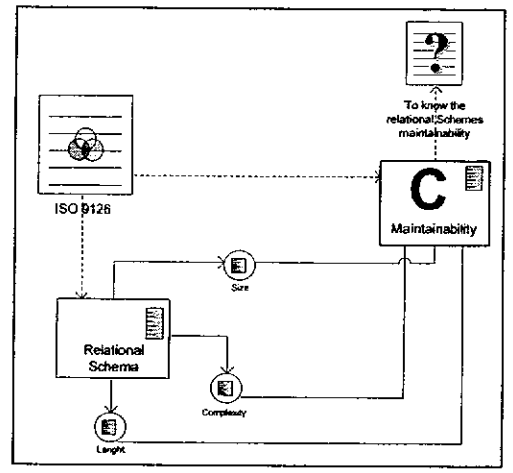


Fig. 7. Instancia del paquete Characterization and Objectives utilizando SMML.

El Segundo caso de estudio se presenta en [13]. Este artículo muestra como se puede instanciar la ontología de medición del software SMO para definir modelos de calidad de datos para portales web; y justifica la utilización de SMO para definir DSL para la medición de entidades software.

La Fig. 8 muestra toda la información necesaria para representar un modelo de medición de PDQM. Los elementos de medición utilizados son: *Information Need, Quality Model, Measurable Concept, Attribute, Base Measure, Derived Measure, Indicator, Measurement Method, measurement Function* y *Analysis Model.*

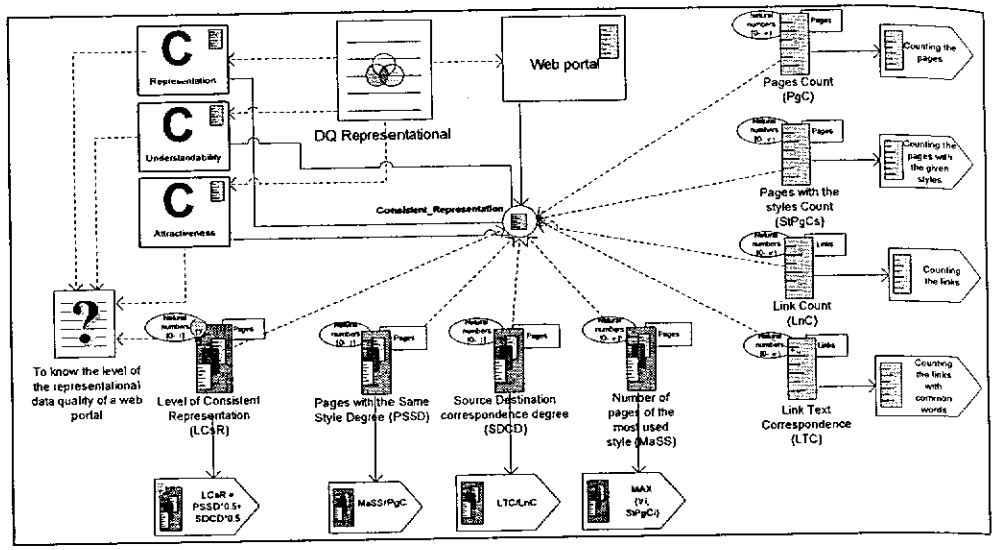


Fig. 8. Modelo de medición de PDQM representado con SMML

En este caso, a pesar de tener que definir numerosos elementos, la representación continua siendo fácil e intuitiva. Además, es fácil identificar los elementos de medición utilizando el lenguaje en vez de cualquier lenguaje de propósito general como UML.

Con respecto a los requisitos esperados [20], se muestran los requisitos que se validan en el lenguaje

- **Conformidad:** los constructores del lenguaje se corresponden con conceptos importantes del dominio.
- **Ortogonal:** cada constructor del lenguaje se utiliza para representar un concepto distinto (*Attribute*, *Base Measure*, etc.) en el dominio.
- **Con soporte tecnológico:** herramientas como MS/DSL Tools o GMF [7] dan soporte al lenguaje SMML.
- **Simple:** el DSL es simple para expresar los conceptos del dominio y dar soporte a sus usuarios.
- **Usable:** los constructores del DSL son expresivos y fáciles de entender.

5 Conclusiones y Trabajo Futuro

En este artículo se ha presentado un DSL para la definición de modelos de medición software, ante la importancia de disponer de una notación específica gráfica e intuitiva para el usuario a la hora de construir modelos de medición. El conjunto de iconos que forman parte del lenguaje han sido seleccionado para que resulten lo más familiar posible a los

ingenieros software. Estos ingenieros serán capaces de utilizar el lenguaje para definir los modelos de medición con facilidad, evitando así el uso de lenguajes de propósito general para definir modelos del dominio de medición. Hasta este momento, no se dispone de representaciones gráficas que permitieran definiciones de dichos modelos.

SMML es un lenguaje completo, con una definición sintáctica y semántica clara y una base ontológica sólida. Además, el lenguaje cumple con los requisitos de un DSL: es usable, conforme a un metamodelo, ortogonal, con soporte y simple.

Este lenguaje juega un papel fundamental en SMF [25] ya que permite definir los modelos de medición del software que son entrada en el proceso de medición del software facilitando que la medición sea genérica, es decir, sobre cualquier tipo de entidad software, y homogénea. La representación visual de los modelos de medición hace a SMF más usable.

Actualmente existe un editor gráfico basado en GMF que permite representar modelos de medición del software, una propuesta importante es la integración del DSL con SMF, de manera que se tenga una única herramienta, llamada Software Measurement Tool (SMT) y permita definir modelos de medición sobre un dominio y ejecutar la medición.

Entre los trabajos futuros, otro trabajo importante es la extensión del metamodelo de medición del software SMM con la jerarquía del paquete de *Measurement Approach* incluido en Software Metrics Meta-Model [28].

Además, se validará la usabilidad del lenguaje por medio de una serie de experimentos basados en el estándar ISO 9126. El estudio se centrará en la usabilidad y mantenibilidad La idea es seleccionar un grupo de expertos en modelado y comprobar la usabilidad de este nuevo lenguaje para definir modelos de medición del software. Por ultimo, se quiere aplicar SMF a entornos reales complejos para obtener un mayor refinamiento y validación.

Agradecimientos

Este trabajo ha sido parcialmente financiado por los siguientes proyectos: INGENIO (Junta de Comunidades de Castilla-La Mancha y Consejería de Educación y Ciencia, PAC08-0154-9262) y ESFINGE (Ministerio de Educación y Ciencia, TIN2006-15175-C05-05).

Referencias

1. Briand, L. C., Morasca, S. y Basili, V. R.; *An Operational Process for Goal-Driven Definition of Measures*, IEEE Trans. Softw. Eng. 28. (2002). pp.1106-1125.
2. Canfora, G., García, F., Ruiz, F. y Visaggio, C. A.; *Applying a framework for the improvement of software process maturity*, Software: Practice & Experience. 36. (2006). pp.283-304.
3. Cook, S. y Frankel, D. S.; *Domain-Specific Modeling and Model Driven Architecture*, MDA Journal. (2004).
4. Champeaux, D., *Object-oriented Development Process and Metrics*, Prentice- Hall, pp (1997).
5. Denny, M.; *Ontology building: A survey of editing tools*. (2003).
6. Deursen, A. v., Klint, P. y Visser, J.; *Domain-Specific Languages: An Annotated Bibliography*, SIGPLAN Notices. 35. (2000). pp.26-36.
7. Eclipse; *Eclipse Graphical Modeling Framework (GMF) Main Page*. In http://www.eclipse.org/gmf/. (2007).

8. Feilkas, M.; *How to represent Models, Languages and Transformations?* Proceedings of the 6th OOPSLA Workshop on Domain-Specific Modeling (DSM'06). (2006). pp.204-213.
9. Fenton, N. y Pfleeger, S. L., *Software Metrics: A Rigorous & Practical Approach, Second Edition*, (1997).
10. García, F., Bertoa, M. F., Calero, C., Vallecillo, A., Ruiz, F., Piattini, M. y Genero, M.; *Towards a consistent terminology for software measurement*, Information and Software Technology 48 (8). (2006). pp.631-644
11. García, F., Bertoa, M. F. y Vallecillo, A., *An ontology for software measurement, Technical Report*, UCLM DIAB-04-02-2, (2004).
12. García, F., Piattini, M., Ruiz, F., Canfora, G. y Visaggio, C. A.; *FMESP: Framework for the modeling and evaluation of software processes*, Journal of Systems Architecture - Agile Methodologies for Software Production. 52. (2006). pp.627-639.
13. García, F., Ruiz, F., Calero, C., Bertoa, M. F., Vallecillo, A., Mora, B. y Piattini, M.; *On the Effective Use of Ontologies in Software Measurement*, The Knowledge Engineering Review (in press). 0. (2008). pp.1-24.
14. García,F., Serrano, M., Cruz-Lemus, J., Ruiz, F. y Piattini, M.;*Managing Software Process Measurement :A Metamodel-Based Approach*, Inf Sciences. Vol 177. pp.2570-2586 (2007).
15. Guerra, E., Lara, J. d. y Díaz, P.; *Visual specification of measurements and redesigns for domain specific visual languages*, Journal of Visual Languages and Computing. (2008).
16. ISO/IEC, *ISO 15939: Software Engineering - Software Measurement Process*, (2002).
17. ISO/IEC; *Software and Systems Engineering - Guidelines for the application of ISO/IEC 9001:2000 to Computer Software*, International Standards Organization. (2004).
18. Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I. y Valduriez, P.; *ATL: a QVT-like Transformation Language*, Companion to the 21th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA 2006. Portland, Oregon, USA. (2006). pp.719-720.
19. Jouault, F. y Bézivin, J.; *KM3: a DSL for Metamodel Specification*, Formal Methods for Open Object-Based Distributed Systems, 8th IFIP WG 6.1 International Conference, FMOODS 2006. 4037. Bologna, Italy. (2006). pp.171-185.
20. Kolovos, D. S., Paige, R. F., Kelly, T. y Polack, F. A. C.; *Requirements for Domain-Specific Languages*, First ECOOP Workshop on Domain-Specific Program Development (ECOOP'06). Nantes, France. (2006).
21. Kurtev, I., Bézivin, J., Jouault, F. y Valduriez, P.; *Model-based DSL Frameworks*, (2006).
22. MacDonell, S. G., Shepperd, M. J. y Sallis, P. J.; *Metrics for Database Systems: An Empirical Study*, IEEE METRICS 1997.
23. McGarry, J., Card, D., Jones, C., Layman, B., Clark, E., Dean, J. y Hall, F., *Practical Software Measurement. Objective Information for Decision Makers*, (2002).
24. Mernik, M., Heering, J. y Sloane, A. M.; *When and how to develop domain-specific languages*, ACM Computing Surveys (CSUR). Volume 37. (2005). pp.316-344.
25. Mora, B., García, F., Ruiz, F., Piattini, M., Boronat, A., Gómez, A., et al. *Software Measurement by using QVT Transformation in an MDA context*, ICEIS 2008. pp.117-124.
26. Mora, B., Ruiz, F., Garcia, F. y Piattini, M., *SMML: Software Measurement Modeling Language*, Technical Report UCLM-TSI-003, (2008). www.esi.uclm.es:8080/tsi/informes
27. OMG, *OCL 2.0 - OMG Final Adopted Specification*, Object Management Group. (2003).
28. OMG, *Architecture-Driven Modernization (ADM): Software Metrics Meta-Model (SMM)*.
29. Özgür, T., Comparison of Microsoft DSL Tools and Eclipse Modeling Frameworks for Domain-Specific Modeling In the context of the Model Driven Development, 2007.
30. Pelechano, V., Albert, M., Javier, M. y Carlos, C.; *Building Tools for Model Driven Development comparing Microsoft DSL Tools and Eclipse Modeling Plug-ins*, DSDM'06.
31. Sánchez, J., García, J. y Menárguez, M.; *RubyTL: A Practical, Extensible Transformation Language.*, ECMDA-FA 2006 4066/2006. (2006). pp.158-172.

Modelado de Líneas de Procesos mediante SPEM v2.0

Tomás Martínez-Ruiz, Félix García, and Mario Piattini

Grupo de Investigación Alarcos, Departamento de Tecnologías y Sistemas de Información, Escuela Superior de Informática, Universidad de Castilla-La Mancha
Paseo de la Universidad, 4, 13071 Ciudad Real, España
{tomas.martinez,felix.garcia,mario.piattini}@uclm.es

Resumen. Recientemente OMG ha propuesto SPEM, una iniciativa para el modelado conjunto de procesos y métodos. Sin embargo, aunque las Líneas de Procesos están en pleno auge, SPEM no considera los mecanismos de variación adecuados para modelarlas. Por ello, en este artículo se proponen nuevos mecanismos para SPEM con los que podrá dar soporte a las Líneas de Procesos Software. Los nuevos mecanismos de variabilidad propuestos aportan a SPEM la capacidad de variación en la que se basan las Líneas de Procesos, mediante la gestión de *puntos de variación* y *variantes* y las relaciones entre ellos. Con la incorporación de estos mecanismos, SPEM soportará el modelado de Líneas de Proceso Software, tal y como se puede ver aplicado al modelo COMPETISOFT. Todo ello facilitará la creación de modelos de procesos adaptados a diferentes contextos organizacionales, propiciando la mejor integración de los procesos dentro de los cuerpos organizacionales.

Palabras clave: SPEM, Línea de Procesos Software, Variabilidad, Punto de Variación, Variante

1 Introducción

La industria dedicada al desarrollo de software está interesada en el modelado de los procesos que se llevan a cabo dentro de las organizaciones como un mecanismo eficaz para conocer y mejorar dichos procesos. En respuesta a esta demanda, OMG ha diseñado SPEM [8], un metamodelo que estandariza el modelado tanto de procesos como métodos.

Sin embargo, los procesos que suceden en empresas distintas son, por definición, distintos. Esto implica que un modelo de procesos genérico como CMMI o ISO/IEC 12207 no se puede implantar de igual forma en cualquier organización [13]. Estos modelos se deben adaptar a las condiciones específicas de cada organización y proyecto, dado que como afirma Humphrey *Al igual que no hay dos proyectos iguales, tampoco hay dos procesos iguales en el mundo* [3]. Como resultado de la adaptación de un proceso a diferentes proyectos y organizaciones específicas en los que se lleva a cabo, se genera un conjunto de procesos muy parecidos pero con pequeñas diferencias entre ellos que los hacen apropiados