Departamento de Tecnologías y Sistemas de la Información

Universidad de Castilla-La Mancha

Tesis Doctoral

SMF: Marco de Trabajo Basado en MDE para la Medición Genérica del Software

Doctoranda: Doña. Beatriz Mora Rivas

Directores: Dr. D. Félix Óscar García Rubio

Dr. D. Francisco Ruiz González

Departamento de Tecnologías y Sistemas de la Información

Universidad de Castilla-La Mancha

Tesis Doctoral

SMF: Marco de Trabajo Basado en MDE para la Medición Genérica del Software

Ciudad Real (España), enero de 2011

Doctoranda: Doña. Beatriz Mora Rivas

Directores: Dr. D. Félix Óscar García Rubio

Dr. D. Francisco Ruiz González

Dedicatoria:

A mis padres, Isabel y Miguel por su confianza, cariño y apoyo incondicional. Gracias Papá y Mamá por cada abrazo y consejo que me han ayudado a continuar y seguir adelante a pesar de las dificultades encontradas.

A mi hermano Miguel, por su envidiable optimismo y alegría que en todo momento me ha transmitido. Gracias Miguel por tu paciencia y enseñarme a ser una buena persona por encima de todo

A Inma mi gran amiga y compañera, a Raquel mi gran prima y confidente, a mis amigas Ana, Lucía, Irene, Loli y Eva. Gracias por multiplicar mis alegrías y dividir mis penas. Por arrancarme cada día una sonrisa. Por tener siempre un hueco para estar conmigo y compartir vuestro tiempo. Gracias por darme la ayuda que he necesitado en cada momento.

Al pequeño Troy, el incansable amigo fiel que tantos paseos ha sacrificado por estar cerca a cambio de nada.

Se dice que las mejores esencias se guardan en frascos pequeños. ¡IMPOSIBLE! ¡Sois demasiado grandes! Gracias a todos por ser los protagonistas de mis mejores momentos.

¡¡Os quiero!!

Agradecimientos

En primer lugar, mi agradecimiento a mis directores de tesis Félix García y Francisco Ruiz. Para mi ha sido todo un orgullo y placer trabajar con dos grandes personas y profesionales. Vuestra ayuda, consejos, conocimiento y confianza me han animado a continuar este trabajo. Gracias por vuestra paciencia y tranquilidad transmitida en todo momento. Sin duda este trabajo nunca podría haber salido adelante si no es gracias a vosotros.

A Artur Boronat, por haberme brindado la posibilidad de trabajar durante un tiempo con él. Por sus comentarios y sugerencias.

A Juan Andrada por su ayuda en la elaboración de la herramienta.

A mis compañeros del Grupo Alarcos por su inestimable ayuda y consejos recibidos a lo largo de la realización de la tesis. A mis más especiales compañeros de Indra-UCLM Ismael, Alfonso, Elvira, Francisco y Tomás. A Goyi y Marisa por vuestra ayuda administrativa. A Luis por brindar soporte tecnológico. A todos os doy las gracias. Ha sido un placer haber trabajado con vosotros.

A mis amigas y compañeras de trabajo, Ana, Cristina y Teresa. Por haber colaborado activamente en la validación de este trabajo. Por vuestros ánimos y apoyo recibido.

Gracias a todas aquellas personas que han formado parte de este trabajo, y que de una manera u otra han ayudado a conseguirlo.

El genio comienza las grandes obras, pero sólo el trabajo las acaba.

Joseph Joubert

Índices.

Índice de Contenidos.

Índice de Figuras.

Índice de Tablas.

Resumen.

En esta tesis se ha abordado el problema de cómo mejorar la medición de los artefactos software de una forma diferente a la manera tradicional, basada en trabajar con métricas/medidas y herramientas útiles sólo para algún determinado dominio y tipo de artefactos software. Para ello se ha ideado y desarrollado un marco de trabajo (framework) para abordar la medición "genérica" del software, de forma que es posible expresar en modelos de medición toda la información necesaria (qué, cómo, cuando, quien, porqué medir) en un contexto dado del mundo real (un proceso o proyecto) y, en base a dichos modelos, obtener automáticamente los valores de las medidas.

Para lograr este objetivo se aprovecha el potencial de genericidad que tiene el paradigma de ingeniería dirigida por modelos (MDE). Como resultado, en esta tesis se define el marco SMF (Software Measurement Framework), formado por los siguientes elementos:

1) **Marco Conceptual** para la representación y gestión del conocimiento relacionado con la medición genérica del software. Constituye la herramienta intelectual necesaria para medir los modelos que representan las entidades software. Está formado de los siguientes elementos:

   - *Arquitectura Conceptual*, que facilita la gestión de los elementos relacionados con la medición del software (el lenguaje de modelado de medición, los metamodelos de dominio y los metamodelos de transformación de modelos) en cuatro niveles de abstracción.
   - *Lenguaje de definición de modelos de medición SMML (Software Measurement Modeling Language)*, que permite representar los modelos de medición de cualquier entidad de manera gráfica y de una forma homogénea. SMML se ha definido a partir de SMM (Software Measurement Metamodel).
   - *Metamodelos para la definición de Entidades Software (Metamodelos de Dominio)*, que permiten representar las entidades software mediante modelos y así poder realizar la medición de las entidades relacionadas con artefactos software.
   - *Metamodelos (Lenguajes) de Transformación de modelos*, para realizar transformaciones entre modelos, con el fin de obtener de forma automática los resultados de la medición. Los lenguajes empleados son MOFScript y QVT Relation.

2) **Método de Trabajo** que proporciona los pasos necesarios para llevar a cabo una medición genérica. El método consiste en la incorporación del metamodelo y modelo de dominio, definición del modelo de medición y en la ejecución automática de la medición.

3) **Formas de medir genéricas** necesarias para que las mediciones sean aplicables a cualquier dominio software. Las formas de medir genéricas permiten trabajar con conceptos más abstractos para conseguir soluciones más generales, es decir, válidas para muchos más casos. Para conseguirlo se trabaja a nivel de metamodelo en vez de modelo.

4) **Entorno Tecnológico** constituido por SMTool, una herramienta que da utilidad real al marco conceptual propuesto. SMTool es un plug-in para la plataforma ECLIPSE que

permite crear y editar modelos de medición software usando el lenguaje gráfico SMML; y obtener de forma automática los valores de las medidas (siempre y cuando se disponga de los metamodelos-modelos necesarios) mediante transformaciones QVT.

La propuesta de la tesis se validó empíricamente mediante un experimento que evaluó la usabilidad del lenguaje SMML y mediante un caso de estudio llevado a cabo en una empresa de desarrollo, en donde se aplicó el marco de trabajo SMF en tres dominios de medición diferentes dentro de la fase de análisis y diseño. Como resultado de la validación se pudo concluir que SMML es un lenguaje usable para representar modelos de medición de forma gráfica e intuitiva y el entorno integrado permite llevar a cabo las mediciones en distintos dominios y la posibilidad de reutilización de los modelos de medición mejorando la productividad del proceso de medición.

Abstract.

This thesis has addressed the issue of improving the measurement of the software artifacts in a different manner with regard to the traditional way, based on working with measures and tools in a particular domain and type of software artifacts. To obtain it, a framework has been designed and developed to address the "generic" software measurement, by means of which it is possible to represent in measurement models all the necessary information (what, how, when, who, why measure) in a given real context (a process or project) and, based on these models, automatically obtain the values of the measures.

To achieve this goal, the potential of the MDE paradigm has been exploited. As a result, this thesis defines the SMF framework (Software Measurement Framework), which is composed of the following elements:

1) **Conceptual Framework** for the representation and management related with the knowledge of the generic software measurement. This element is necessary to measure models which represent software entities and it is composed by the following elements:
   - *Conceptual Architecture*, which facilitates the representation of the elements related to the software measurement (the measurement modeling language, the domain metamodel and models transformation metamodels) at four abstraction levels. .
   - *Software Measurement Modeling Language SMML*, which allows to represent software measurement models of any entity in a graphically manner and in a homogeneous way. SMML is defined from SMM (Software Measurement Metamodel).
   - *Metamodels to Entity Software Definition (Domain Metamodels)*, which permit to represent the software entities by means of models in order to carry out the software measurement of the entities related with the software artifacts.
   - *Models Tranformation Metamodels (Languages)*, which are necessary to carry out transformations between models in order to obtain the measurement results in an automatic manner. The languages used are MOFScript and QVT Relation.

2) **Working Method** which provides the necessary steps to carry out generic software measurements. The method consists of the incorporation of the domain model and metamodel, the definition of the software measurement model and the automatic execution of the measurement.
3) **Generic Measurement Approaches**, which can be applied to measure any software domain. The generic measurement approaches work with more abstract concepts in order to get more general solutions, ie, valid for many cases. To achieve it, it deals with metamodel level instead of model level.
4) **Technological Environment**, which is composed of SMTool, a tool which implements the SMF framework. SMTool is a plug-in for the Eclipse platform and supports to create and edit software measurement models by using the graphical language SMML, and to obtain

measures values (if the models and metamodels are available) by means of by QVT transformations.

The thesis approach was empirically validated through an experiment which evaluated the SMML usability and through a case study which was carried out in a development company, where the SMF framework was applied in three different measurement domains in the context of the analysis and design phase. As a result of the validation, the obtained conclusions were that SMML is a usable language to represent measurement models in an intuitive and graphical manner, and the integrated environment allows to perform measurements in different domains and it facilitates the measurement models reuse and improves the measurement process productivity.

"Una búsqueda comienza siempre con la suerte del principiante y termina con la prueba del conquistador".

(Paulo Coelho)

1. Introducción.

En este capítulo se abordan los aspectos generales relacionados con la tesis doctoral presentando en primer lugar, la motivación al trabajo realizado. En segundo lugar, con el fin de guiar la lectura y comprensión, se presentan las hipótesis y objetivos, el marco de trabajo (en cuanto a proyectos de I+D) y por último, la estructura de este documento.

1.1. Planteamiento.

La actual necesidad de la industria del software por mejorar su competitividad fuerza a la búsqueda de la mejora continua de sus procesos. Para conseguirlo, es necesaria una gestión exitosa de dichos procesos (Florac et al., 2000), lo que implica su correcta definición, ejecución, medición, control y mejora. Entre estas fases del ciclo de vida de los procesos destaca la medición, que ayuda a controlar los errores y carencias dentro del desarrollo y mantenimiento del software facilitando la toma de decisiones. Así, la medición se ha convertido en un aspecto fundamental de la Ingeniería del Software.

Para facilitar y promover la mejora continua de sus procesos software, las empresas requieren llevar a cabo la medición del software de manera efectiva y consistente. Los procesos software constituyen la base a partir de la cual se realiza el trabajo dentro de una organización software (aquella que desarrolla y/o mantiene software). En la práctica, dichos procesos se llevan a cabo en forma de proyectos. Como resultado de la ejecución de proyectos concretos se obtienen productos. Esto implica la necesidad de una disciplina para la medición y análisis de datos (Brown y Dennis, 2004) y la definición, recopilación y análisis de medidas sobre el propio proceso, los proyectos y los productos software.

La gran variedad de tipos de entidades y atributos que son candidatos a ser medidos motiva el interés en disponer de modelos de medición homogéneos, que puedan gestionarse por las empresas de la misma forma, independientemente de cuál sea la entidad a medir. Esto supone la necesidad de una referencia consistente y adecuada para la definición de sus modelos de medición del software así como el soporte tecnológico necesario para integrar la medición de los diferentes tipos de entidades.

Con el fin de satisfacer estas expectativas es muy útil considerar el paradigma MDE (Model-Driven Engineering) (Bézivin et al., 2005), de especial importancia en la actualidad. La filosofía de este paradigma consiste en que los modelos son los principales artefactos de los procesos de Ingeniería del Software. La arquitectura MDA "Model-Driven Architecture" (OMG, 2003c) y sus estándares relacionados[1] proporcionan una base conceptual y tecnológica para llevar a la práctica las ideas de dicho paradigma. De acuerdo al estándar MDA, el proceso de desarrollo del software se puede considerar como una serie de transformaciones de modelos, a partir de un nivel de abstracción alto hasta un nivel más específico. En el nivel más abstracto se pueden ver los requisitos, y en el nivel más específico estaría el código que implementa la aplicación. Así, lo más destacado de la formalización de los modelos y las transformaciones entre ellos es que se facilita la automatización del proceso de desarrollo de software.

El campo de la medición software puede beneficiarse de la nueva filosofía MDE, proporcionando la integración y el soporte necesario a la automatización de la medición de las diversas entidades del proceso software. Esto implica: i) la definición de modelos de medición de manera homogénea y consistente a partir de un metamodelo adecuado; ii) la definición de formas de medir genéricas que puedan aplicarse a cualquier modelo; y iii) el soporte necesario para calcular de forma automática las medidas definidas, almacenar de forma homogénea los resultados y facilitar la toma de decisiones mediante el análisis de los mismos. Por tanto, se habla de homogeneidad en el sentido de que existe una manera general de definir y medir cualquier tipo de entidad o artefacto software (a nivel de producto, proceso o proyecto).

Estos aspectos constituyen el principal interés de la presente tesis, en la que se aplican los principios, normas y herramientas de MDA al campo de la medición del software. El

[1] UML "Unified Modelling Language" (OMG, 2001); MOF "Meta Object Facility" (OMG, 2003b); QVT "Query/View/Transformation" (OMG, 2005a); OCL "Object Constrait Language" (OMG, 2003d); y XMI "XML Metadata Interchange" (OMG, 2002c)

objetivo es desarrollar un entorno genérico para la definición de modelos de medición bien formados respecto a un metamodelo común, y para la medición de cualquier entidad software en base a los metamodelos que las representan (metamodelos del dominio). Por otro lado, la disponibilidad de un lenguaje que permita representar los elementos a tener en cuenta en los procesos de medición puede ser de ayuda para la toma de decisiones y mejora de los procesos. En este sentido es interesante considerar la utilización de los Lenguajes de Dominio Específico (Domain Specific Languages – DSLs), donde el nivel de abstracción es alto, de forma que la solución se especifica directamente con los conceptos del dominio. Por este motivo, se pretende desarrollar como parte del entorno propuesto, un lenguaje para la representación de modelos de medición del software de manera gráfica, más fácil e intuitiva que con lenguajes textuales.

Para lograr estos objetivos se considerarán los siguientes proyectos basados en Eclipse: Eclipse Modeling Framework[2] (EMF) y Medini QVT (Ikv, 2010), que proporcionan el soporte tecnológico necesario para la gestión automática de modelos de acuerdo a MDE y MDA.

1.2. Hipótesis y Objetivos.

La hipótesis del trabajo es:

¿Es factible realizar mediciones genéricas del software aplicando el paradigma MDE?

En consonancia con el título de la tesis y con la hipótesis anterior el **objetivo general** de la tesis queda definido como:

Definir un marco de trabajo basado en MDE para la medición genérica del software

y está sujeto a las siguientes características:

- **Medición Genérica:** por medición genérica se entiende que se debe permitir medir cualquier propiedad de cualquier clase de artefacto software, siendo cualquier elemento de: i) un sistema software (código fuente, código ejecutable, documentación); ii) los procesos para su creación, utilización o modificación; o iii) los proyectos asociados a la gestión de dichos procesos. En realidad, la aplicación del paradigma MDE permitirá aplicar el marco a cualquier tipo de dominio representable en una computadora, independientemente de si se trata de software o no. La diferencia es que el marco de trabajo ofrecerá soporte para la representación (modelado y metamodelado) de cualquier dominio y de sus modelos de medición asociados, pero tan sólo en el caso de software será factible dar soporte también para el propio proceso de medición, es decir, la realización automática de las mediciones.
- **Marco de Trabajo (Framework[3]):** el marco de trabajo está formado por tres tipos de elementos:

[2] Disponible en http://www.eclipse.org/modeling/emf/ El proyecto EMF es un Framework de modelado y generación de código para construir aplicaciones basadas en estructuras de modelos.

i) conceptuales: ontología de la medición del software, modelos de medición y lenguajes gráfico y textual para su representación.

ii) metodológicos: proceso de medición definido para el marco de trabajo.

iii) instrumentales (repositorio basado en XMI).

El marco de trabajo deberá incorporar herramientas para crear y representar modelos de medición software (software measurement models, SMM), basados en un metamodelo general, y que se definen en base a modelos de dominio (software domain models, SDM), que representan el dominio específico de los artefactos software que se quieren medir. En un SMM se especifica qué artefactos software se medirán, qué propiedades de dichos artefactos y cómo, cuando, por qué y por quién se medirá.

El anterior objetivo general se concreta en los siguientes objetivos parciales:

- OP1. Adaptar el metamodelo para la medición del software (García et al., 2007) basado en la ontología de la medición del software de García et al. (García et al., 2006a).
- OP2. Definir formas de medición genéricas que permitan la obtención de medidas independientes del dominio.
- OP3. Definir la sintaxis y la semántica de un DSL gráfico para representar modelos de medición software.
- OP4. Elaborar una herramienta para la creación de SMMs mediante instanciación del metamodelo definido en OP1 y usando el lenguaje definido en OP2.
- OP5. Validar la usabilidad y mantenibilidad del lenguaje mediante experimentos.
- OP6. Validar el marco de trabajo con diversos casos de estudio para comprobar su utilidad en los distintos dominios software. En particular se probará con diferentes dominios software.

1.3. Marco de la Tesis.

En este apartado se presenta el entorno en el que la doctoranda ha desarrollado la tesis. Primero se comenta el grupo de investigación, después se presentan los proyectos de I+D que han permitido la financiación de los trabajos constituyen el contexto en el que se ha realizado la presente tesis doctoral, y por último se muestra un resumen de la organización de la memoria.

1.3.1. Grupo de Investigación.

La autora de la presente tesis doctoral ha realizado el trabajo siendo miembro del **Grupo de I+D Alarcos** (http://alarcos.esi.uclm.es). Este grupo de investigación se creó en septiembre de 1997 por los profesores Mario Piattini y Francisco Ruiz. La autora de la tesis se incorporó al grupo Alarcos a finales del año 2005 (diciembre) como becaria de investigación, con una beca de FPI (Formador Personal Investigador) de la Junta de Castilla – La Mancha. En el año 2008 fue contratada por Indra Software Labs, una empresa de desarrollo software, donde ejerce tareas de análisis funcional y aseguramiento de calidad en los productos software. Desde

[3] En el desarrollo de software, un **framework** es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado.

mayo del 2008 trabaja tanto en el grupo de investigación Alarcos como en la empresa privada, aportando los conocimientos del desarrollo software en su investigación y viceversa.

1.3.2. Proyectos de I+D.

Los proyectos de I+D con financiación en alguna convocatoria oficial de ámbito regional o nacional, que han servido de soporte económico para el desarrollo del trabajo de esta tesis, se resumen cronológicamente en la Figura 1-1.

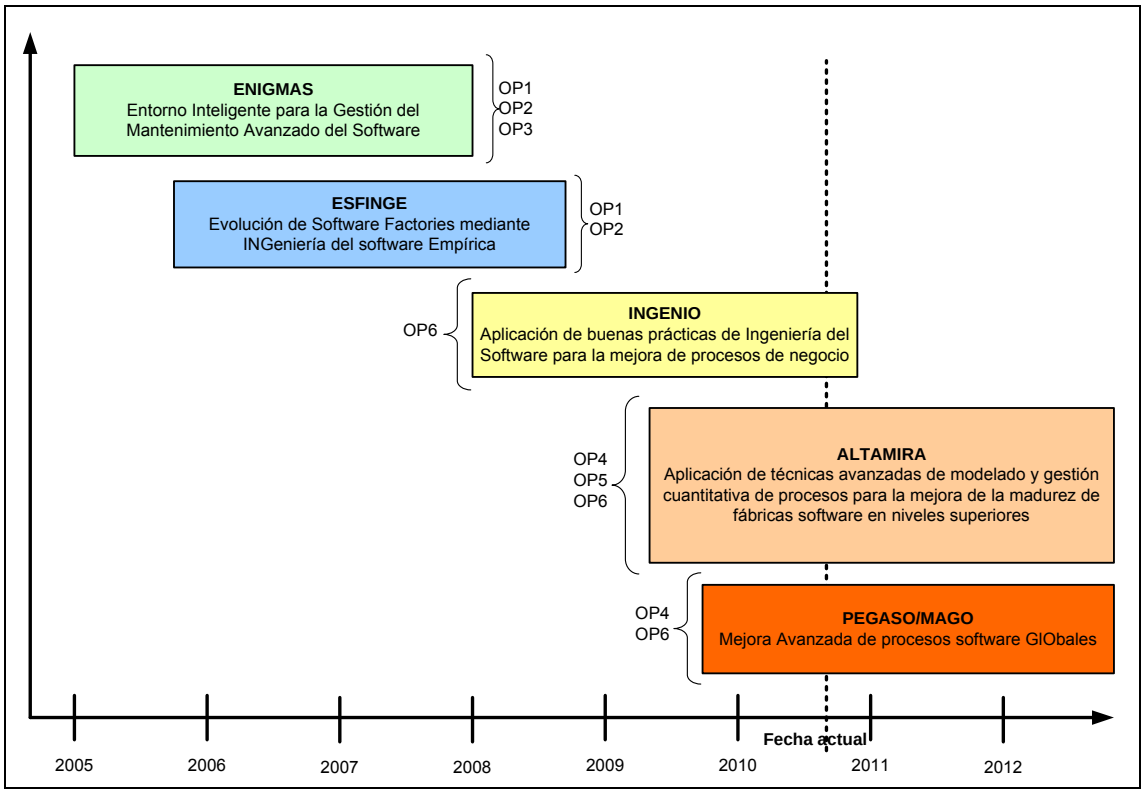


Figura 1-1. Marco de Trabajo de la tesis

En la figura anterior se muestran recuadrados los proyectos I + D. Por cada proyecto se muestra los objetivos cubiertos y financiados por cada uno de los proyectos.

En los siguientes subapartados, se presenta un breve resumen de cada uno de estos proyectos y de las principales contribuciones relacionadas con la tesis doctoral.

1.3.2.1. ENIGMAS.

Este proyecto se enfrentaba a la siguiente pregunta: *¿Cómo mejorar el mantenimiento de los sistemas software desarrollados usando los nuevos entornos tecnológicos y paradigmas surgidos en los últimos años?* Entre estos nuevos paradigmas que las empresas de desarrollo software estaban empezando a utilizar cada vez más, se encontraba el "Model-Driven Software Development" (MDSD) y el "Software as a Service" (SaaS), también conocido como "Service-Oriented Computing" (SOC). Para poder implementar y aplicar estos paradigmas surgieron nuevas tecnologías como los servicios Web o nuevas metodologías y normas como MDA de OMG (Object Management Group). El uso de estas tecnologías y paradigmas permiten construir sistemas software más potentes de forma más eficiente, pero implicaba el surgimiento de nuevos retos y problemas en cuanto a cómo llevar a cabo su mantenimiento. Abordar todos estos problemas nuevos era una tarea demasiado amplia y ambiciosa. Por esta razón, el proyecto

ENIGMAS (Entorno Inteligente para la Gestión del Mantenimiento Avanzado del Software) se centraba en los aspectos referidos a la gestión del mantenimiento. En especial, se abordaba la mejora continua del proceso de mantenimiento de sistemas software que se desarrollen empleando "MDA" y/o "Web Services Architecture" (WSA). Dicha mejora continua estaba basada en el uso de un entorno y un framework integrados que facilitaban herramientas para, entre otras cosas, gestionar el conocimiento necesario para mantener software, definir y medir los aspectos de proceso y producto necesarios, y realizar pruebas de los nuevos artefactos (modelos independientes/dependientes de plataforma, etc.)

Título del proyecto:	**ENIGMAS:** ENtorno Inteligente para la Gestión del Mantenimiento Avanzado del Software
Entidad Financiadora	Junta de Comunidades de Castilla-La Mancha, PBI-05-058
Entidades participantes	Universidad de Castilla La Mancha
Duración	Desde 1/1/2005 hasta 1/12/2007
Investigador Principal	Francisco Ruiz González
Número de investigadores participantes	13
Importe total del proyecto	96800 €

Tabla 1-1. Ficha resumen del proyecto ENIGMAS

1.3.2.2. ESFINGE.

Los objetivos específicos de este proyecto fueron los siguientes.

1. Desarrollar medidas e indicadores para diferentes modelos y arquitecturas de distintos niveles de abstracción, así como un conjunto de valores umbrales y herramientas para su cálculo automático.
2. Definir un marco para la reingeniería y evolución de sistemas en *software factories* basado en la aproximación MDSD.
3. Definir un entorno para las pruebas de software basado en metamodelos.
4. Desarrollar un entorno para la mejora y evolución de modelos de procesos de negocio.
5. Validar diferentes prácticas ágiles para las *software factories*.
6. Definir y validar técnicas y métricas para el desarrollo de software seguro basado en modelos (modelos independientes/dependientes de plataforma, etc.)

Título del proyecto:	ESFINGE: Evolución de Software Factories mediante INGeniería del software Empírica
Entidad Financiadora	Ministerio de Educación y Ciencia, TIN2006-15175-C05-05
Entidades participantes	Universidad Politécnica de Valencia, Universidad de Murcia, Universidad Politécnica de Cartagena, European Software Institute.
Duración	Desde 1/10/2006 hasta 30/9/2009
Investigador Principal	Mario Piattini Velthuis
Número de investigadores participantes	30
Importe total del proyecto	441408 €

Tabla 1-2. Ficha resumen del proyecto ESFINGE

1.3.2.3. INGENIO.

El proyecto INGENIO busca abordar la mejora de los procesos de negocio de forma integrada proporcionando diversas soluciones tanto de carácter metodológico como de carácter tecnológico. Respecto a la parte metodológica, se pretende dotar a las organizaciones de los medios necesarios para la definición efectiva de sus procesos, facilitando la evolución y adaptación de dichos modelos ante entornos de negocio cambiantes y su flexibilidad. Se proporcionan los métodos y mecanismos necesarios para que se puedan llevar a cabo la medición de los procesos y artefactos relacionados de una manera efectiva y consistente. Desde el punto de vista tecnológico, se proporcionan los medios necesarios para la automatización de las actividades organizacionales/empresariales y la comunicación entre sistemas de información automatizados. Las propuestas están basadas en la aplicación en el mundo de la empresa de técnicas que han demostrado ser útiles en el mundo de la Ingeniería del Software. Con todo ello, los objetivos del proyecto son:

- Modelado de Procesos de Negocio.
- Medición de los Procesos de Negocio.
- Automatización de Procesos de Negocio

Título del proyecto:	INGENIO: Aplicación de buenas prácticas de Ingeniería del Software para la mejora de procesos de negocio
Entidad Financiadora	Junta de Comunidades de Castilla-La Mancha y Consejería de educación y Ciencia. PAC08- 0154-9262
Entidades participantes	Universidad de Castilla-La Mancha, Universidad Politécnica de Valencia
Duración	Desde 1/1/2008 hasta 31/12/2010
Investigador Principal	Félix Óscar García Rubio
Número de investigadores participantes	18
Importe total del proyecto	130000 €

Tabla 1-3. Ficha resumen del proyecto INGENIO

1.3.2.4. ALTAMIRA.

En el **proyecto ALTAMIRA** se pretende promover la mejora a niveles altos de madurez en las empresas software mediante la introducción de técnicas relacionadas con el modelado avanzado, la medición genérica y la gestión cuantitativa de los procesos en un entorno de mejora continua. El proyecto ALTAMIRA, tiene como objetivo llevar a cambio mediciones del software de manera efectiva y consistente, para ello se desarrolla un entorno de medición genérica de cualquier modelo software según el paradigma MDE; se define un metamodelo y lenguaje gráfico representar modelos de medición software; se desarrollan las herramientas necesarias y se validará el entorno mediante su aplicación en proyectos reales.

Título del proyecto:	ALTAMIRA: Aplicación de técnicas avanzadas de modelado y gestión cuantitativa de procesos para la mejora de la madurez de fábricas software en niveles superiores
Entidad Financiadora	Junta de Comunidades de Castilla-La Mancha, Fondo Social Europeo (PII2I09-0106-2463)
Entidades participantes	Universidad de Castilla-La Mancha, INDRA SW Labs
Duración	Desde 1/4/2009 hasta 31/12/2012
Investigador Principal	Félix García Rubio
Número de investigadores participantes	11
Importe total del proyecto	275687,51 €

Tabla 1-4. Ficha resumen del proyecto ALTAMIRA

1.3.2.5. PEGASO/MAGO.

En la actualidad el desarrollo de software se lleva a cabo en entornos globales, combinando modalidades de externalización (*outsourcing*) tanto en países alejados (*offshoring*) como cercanos (*nearshoring*), lo que presenta interesantes retos de investigación tanto del punto de vista técnico como de gestión; todo ello con un posible alto impacto en el entorno empresarial.

En el proyecto PEGASO se investiga la aplicación de diferentes técnicas de la ingeniería de procesos software con el fin de **mejorar la calidad del software desarrollado en entornos globales**. Estas técnicas se contrastan experimentalmente utilizando Ingeniería del Software Empírica. Las aportaciones previstas pueden clasificarse en cuatro áreas:

- Definición de procesos software adaptables y flexibles
- Desarrollo de marcos multimodelo para la mejora de la calidad del software
- Diseño de técnicas avanzadas para la gestión cuantitativa de procesos software
- Desarrollo de un método para la ingeniería de requisitos global

Se trata de un proyecto coordinado entre el Grupo Alarcos de la Universidad de Castilla-La Mancha y el Grupo de Investigación de Ingeniería del Software de la Universidad de Murcia. Cuenta con la colaboración de grupos de investigación internacionales, lo que hará posible la experimentación en entornos globales, así como con numerosas empresas, que permitirán contrastar las técnicas propuestas en factorías de software reales.

Título del proyecto:	PEGASO/MAGO: Mejora Avanzada de procesos software GlObales
Entidad Financiadora	Ministerio de Ciencia e Innovación (TIN2009-13718-C02-01)
Entidades participantes	Universidad de Castilla-La Mancha, Universidad de Murcia (proyecto coordinado PEGASO: Procesos para la mEjora del desarrollo GlobAl del Software)
Duración	Desde 1/10/2009 hasta 30/12/2012
Investigador Principal	Mario Piattini Velthuis
Número de investigadores participantes	28
Importe total del proyecto	489100 €

Tabla 1-5. Ficha resumen del proyecto PEGASO/MAGO

1.4. Organización de la Tesis.

La tesis doctoral está estructurada en ocho capítulos, dos apéndices y cuatro anexos. Los contenidos del resto del documento son:

- **Capítulo 2. Método de Trabajo.** Se presenta los métodos de trabajo adoptados para la consecución de los objetivos planteados.

- **Capítulo 3. Estado del arte.** Se presentan de forma resumida los trabajos conocidos relacionados con los aspectos de modelado, medición y mejora de los procesos software.

- **Capítulo 4. SMF: Marco de Trabajo de medición del software**. Se presentan las características del marco SMF (Software Measurement Framework) incluyendo la arquitectura y la metodología de uso.

- **Capítulo 5. DSL y Validación.** Se presenta el DSL de medición del software y la validación de dicho lenguaje.

- **Capítulo 6. Entorno Tecnológico.** En este capítulo se describen las herramientas software que forma parte del SMF que permiten el modelado de modelos software y la realización de mediciones genéricas a partir de los modelos de medición definidos.

- **Capítulo 7. Caso de Estudio.** Se presenta un caso práctico de aplicación del marco de trabajo SMF en la empresa de desarrollo: Indra Software Labs. Aplicándose durante todo el ciclo de desarrollo software.

- **Capítulo 8. Conclusiones y Trabajo Futuro.** Se presentan las principales aportaciones, los resultados obtenidos y las líneas que quedan abiertas para una futura investigación.

- **Apéndices.** Incluyen la lista de acrónimos utilizados en el texto y la lista de referencias bibliográficas citadas.

- **Anexos:** Los anexos incluidos amplían y precisan información o ejemplos útiles para la mejor comprensión de algunos de los aspectos presentados en los capítulos anteriores. La lista de anexos es la siguiente:
  - A – Materiales de los experimentos.
  - B – Gráficas Estadísticas
  - C – Protocolo de Revisión Sistemática
  - D – Cuestionario del Caso de Estudio

Los que se enamoran de la práctica sin la teoría son como los pilotos sin timón ni brújula, que nunca podrán saber a dónde van (Leornardo da Vinci)

2. Método de Trabajo.

En este capítulo se presentan los métodos empleados para la realización de la tesis doctoral: Investigación-Acción (Action-Research) como familia de métodos cualitativos, "Ingeniería del Software Empírica" como familia de métodos cuantitativos, un método concreto para el desarrollo de un DSL y un método para realizar revisiones sistemáticas de la literatura.

En primer lugar se presenta una visión general de los métodos de investigación, cualitativos y cuantitativos, empleados en Ingeniería del Software. A continuación, se describen las características básicas del método cualitativo Investigación-Acción, así como los aspectos especiales que deben tenerse en cuenta al utilizar Investigación-Acción para hacer I+D en Sistemas de Información y, en especial, en Ingeniería del Software. En particular, se describen los participantes que han intervenido y los principales ciclos que se han desarrollado en la presente tesis. En el apartado 2.3 se describe el método que se ha utilizado para desarrollar el lenguaje de modelado de la medición del software SMML. A continuación, en el apartado 2.4, se presentan los métodos cuantitativos aplicados para la validación empírica del entorno propuesto. Por último, en el apartado 2.5, se presenta el método que se ha seguido para realizar revisiones sistemáticas, que ha servido para abordar los trabajos relacionados con los lenguajes de modelado de MDA y la medición genérica y definir el estado del arte.

2.1. Métodos de Investigación en Ingeniería del Software.

Es necesario el aprendizaje para comprender una disciplina, esto significa, observar, reflexionar y encapsular el conocimiento, construir el modelo (del dominio de aplicación, de los procesos para resolver problemas, etc.), experimentar, y evolucionar los modelos con el tiempo (Basili, 2000). En muchas áreas como la física, las matemáticas o la medicina se ha utilizado este método, pero las únicas diferencias entre unas áreas y otras, están en el análisis y construcción de los modelos, y en cómo llevar a cabo la experimentación.

Si se mira desde un punto de vista científico, las áreas que deberían tratarse con un carácter experimental son las áreas de Ingeniería del Software y Sistemas de Información. En estos casos, el objetivo de la investigación es conocer la naturaleza de los procesos, productos y sus relaciones en el contexto de un sistema organizacional (en el caso de Sistemas de Información) o del sistema software. Al igual que en otro campo científico aplicado la Ingeniería del Software tiene dos objetivos fundamentales (Philips, 1998): a) incrementar el conocimiento teórico para comprender el por qué de las cosas en un área particular de interés; b) y mejorar la práctica de manera que los resultados sean útiles y con una aplicación práctica. A veces estos dos objetivos no se cumplen, ya que en muchas áreas se presentan nuevas propuestas, métodos, etc. sin aplicación, existiendo una desconexión entre la investigación teórica y la investigación práctica (Moody, 2000). Es importante eliminar este obstáculo entre la teoría y la práctica, por eso, es necesario que la investigación en la Ingeniería del Software esté orientada a objetivos prácticos, y que la industria del software aplique los resultados obtenidos en la investigación.

Para conseguirlo existen métodos de trabajo en Ingeniería del Software adecuados en función del tipo de investigación que se está realizando. Los métodos de investigación se pueden clasificar de varias formas, la clasificación que divide los métodos en cuantitativos y cualitativos es la más aceptada (Myers, 1997).

Los métodos de **investigación cuantitativos** se utilizan para el estudio de fenómenos naturales, dentro de este tipo de método se incluirían además los métodos deductivos y empíricos. Los métodos de **investigación cualitativos** se utilizan para el estudio de fenómenos sociales y culturales, ya que no se basan en teorías formales ni en experimentos, sino en entrevistas, documentos, cuestionarios, reacciones e impresiones del investigador, etc. Dentro de este tipo de método se incluirían métodos como la etnografía y el método investigación-acción. Las diferencias de estos dos métodos se muestran resumidos en la Tabla 2-1.

Aspecto	Investigación Cualitativa	Investigación Cuantitativa
Propósito	proveer conocimiento sobre una organización y/o un problema y sus soluciones	generalizar los resultados desde un caso de estudio a la población total de interés
Orientación	a la verificación	al descubrimiento
Tamaño de la muestra	Pequeño	grande
Recogida de datos	datos poco estructurados obtenidos mediante entrevistas, observaciones y discusiones en grupo	datos y técnicas muy estructurados
Análisis de datos	no estadístico	estadístico

Tabla 2-1. Investigación cualitativa vs cuantitativa

Como se muestra en la Tabla 2-1 es en la naturaleza de los datos y en el análisis realizado donde se encuentran las principales diferencias entre la investigación cualitativa y cuantitativa. En la investigación cuantitativa los datos son más estructurados y se utilizan técnicas estadísticas para su análisis, al contrario que en la investigación cualitativa que los datos (obtenidos mediante entrevistas, observaciones y discusiones en grupo) son poco estructurados y no se suelen analizar de forma estadística.

Gran parte de la investigación llevada a cabo en las áreas de Sistemas de Información e Ingeniería del Software es de tipo cuantitativo y está basada en técnicas estadísticas.

En los siguientes apartados se describen las características básicas de los métodos de investigación cuantitativos y cualitativos y se expone cómo se han aplicado estos métodos para cumplir con los objetivos marcados en la tesis.

2.2. Investigación-Acción.

Entre los diversos métodos de investigación cualitativa, el método más utilizado en Sistemas de Información e Ingeniería del Software es Investigación-Acción. El término "Investigación-Acción" fue propuesto por primera vez en 1947 por el autor Kurt Lewin. Se trata de una forma de investigación para enlazar el enfoque experimental de la ciencia social con programas de acción social que respondan a los problemas sociales principales. Mediante la investigación-acción se pretende tratar de forma simultánea conocimientos y cambios sociales, de manera que se unan la teoría y la práctica. El concepto tradicional de investigación-acción proviene del modelo Lewin de las tres etapas del cambio social: descongelamiento, movimiento, recongelamiento. El proceso consiste en:

- Insatisfacción con el actual estado de cosas.
- Identificación de un área problemática.
- Identificación de un problema específico a ser resuelto mediante la acción.
- Formulación de varias hipótesis.
- Selección de una hipótesis.
- Ejecución de la acción para comprobar la hipótesis.
- Evaluación de los efectos de la acción.
- Generalizaciones.

Lewin sugería esencialmente que las tres características más importantes de la Investigación-Acción moderna eran: su carácter participativo, su impulso democrático y su contribución simultánea al conocimiento en las ciencias sociales.

En los últimos años este método ha obtenido una gran aplicación y aceptación en la investigación en Ingeniería del Software, desde que fue introducida en el año 1985 por (Wood-Harper, 1985).

Existen diversas definiciones de Investigación-Acción. Algunas de las más significativas son las siguientes:

- Para (McTaggart, 1991) es la manera de preparar las condiciones necesarias para que los grupos de personas aprendan de sus propias experiencias, y hagan que estas experiencias sean accesibles a los demás.
- Para (French y Bell, 1996) es el proceso de recopilar datos de la investigación de forma sistemática sobre un sistema actual con respecto a algún objetivo o necesidad de ese

sistema; de alimentar de nuevo al sistema con esos datos; de llevar a cabo acciones por medio de variables alternativas seleccionadas dentro del sistema, basándose en los datos y en las hipótesis; y de evaluar los resultados de las acciones, recopilando datos adicionales.

- Para (Wadsworth, 1998) consiste en que todas las partes involucradas en la investigación participen, examinando la situación existente (que perciben como problemática), con el objetivo de cambiarla y mejorarla.

A partir de las definiciones anteriores se deduce que la Investigación-Acción tiene una doble finalidad: por un lado beneficiar al investigador y, por otro lado, generar "conocimiento de investigación" relevante (Kock y Lau, 2001). Por tanto, el método Investigación-Acción tiene por objetivo unir la teoría y la práctica entre investigadores y profesionales mediante un proceso cíclico. Este método está orientado a producir teoría que sea útil en la práctica, y se obtiene buscando soluciones en situaciones reales que le ocurren a un grupo de profesionales (*practitioners*) (Avison et al., 1999). La intervención de un investigador en la realidad de este grupo hace posible esta relación entre teoría y práctica. Los resultados de la experiencia deben favorecer al investigador y a los profesionales. Una hipótesis elemental de este método de investigación es que los procesos sociales complejos (y el uso de tecnologías de la información en organizaciones es de este tipo) se pueden estudiar mejor si se introducen cambios en dichos procesos y se observan los efectos de dichos cambios (Baskerville, 1999).

En el área de los Sistemas de Información, normalmente el cliente de una investigación es una organización en la que el investigador provee alguno de los siguientes servicios: desarrollo software, consultoría, ayuda para cambiar, etc. a cambio de tener acceso a datos de interés para la investigación y, en muchos casos, de recibir financiación (Kock y Lau, 2001). En cualquier caso, el investigador que utiliza Investigación-Acción en Sistemas de Información (IA-SI) sirve a dos entidades diferentes: el cliente de la investigación y la comunidad científica de Sistemas de Información. Las necesidades de ambos suelen ser muy diferentes e incluso opuestas. El principal desafío al que se tienen que enfrentar todos los investigadores de IA-SI es intentar satisfacer ambas demandas.

En un análisis más formal de los participantes en Investigación-Acción, Wadsworth (1998) identifica cuatro tipos de roles en este método (en algunas ocasiones la misma persona o equipo puede desempeñar más de un rol):

- El **investigador**, persona o grupo de personas que lleva a cabo el proceso de investigación.
- El **objeto investigado**, es decir, el problema a resolver.
- El **grupo crítico de referencia**, es el grupo que se investiga con el objetivo de resolver el problema que tiene. Este grupo también participa en el proceso de investigación (aunque no tan activamente como el investigador). En él hay tanto personas que saben que están participando en la investigación, como otras que participan sin saberlo.
- El **beneficiario** (*stakeholder*), aquél que se beneficia del resultado de la investigación pero no participa directamente en el proceso. En este grupo, están por ejemplo, las empresas que se benefician de un nuevo método para resolver problemas en tecnologías de la información, o los técnicos que aplican dicha metodología.

French y Bell (1996) proponen cuatro variantes que dependen principalmente de las características de la investigación:

- **Diagnóstico:** el investigador se adentra en una situación problemática, la diagnostica y recomienda al grupo crítico de referencia, pero sin controlar posteriormente sus efectos.
- **Participativa:** el grupo crítico de referencia aplica las recomendaciones del investigador, compartiendo con él sus resultados y sus efectos.

- **Empírica:** el grupo crítico de referencia registra sus efectos y acciones de manera sistemática. Esta característica hace que esta variante difícilmente se aplique.
- **Experimental:** consiste en la evaluación de las diferentes opciones que hay para conseguir un objetivo. El principal problema de esta variante está en la dificultad de poder medir de manera objetiva las diversas opciones, ya que por norma general se aplicarán, o bien en distintas organizaciones con distintas características que enturbian los resultados de la investigación, o bien en una sola organización pero en distintos momentos, con lo que el entorno del experimento habrá cambiado.

Un proceso de investigación que emplea Investigación-Acción se compone de grupos de actividades organizadas formando un ciclo. (Padak y Padak, 1994) identifican las cuatro etapas que se deben seguir en las investigaciones que utilicen este método (véase Figura 2-1).

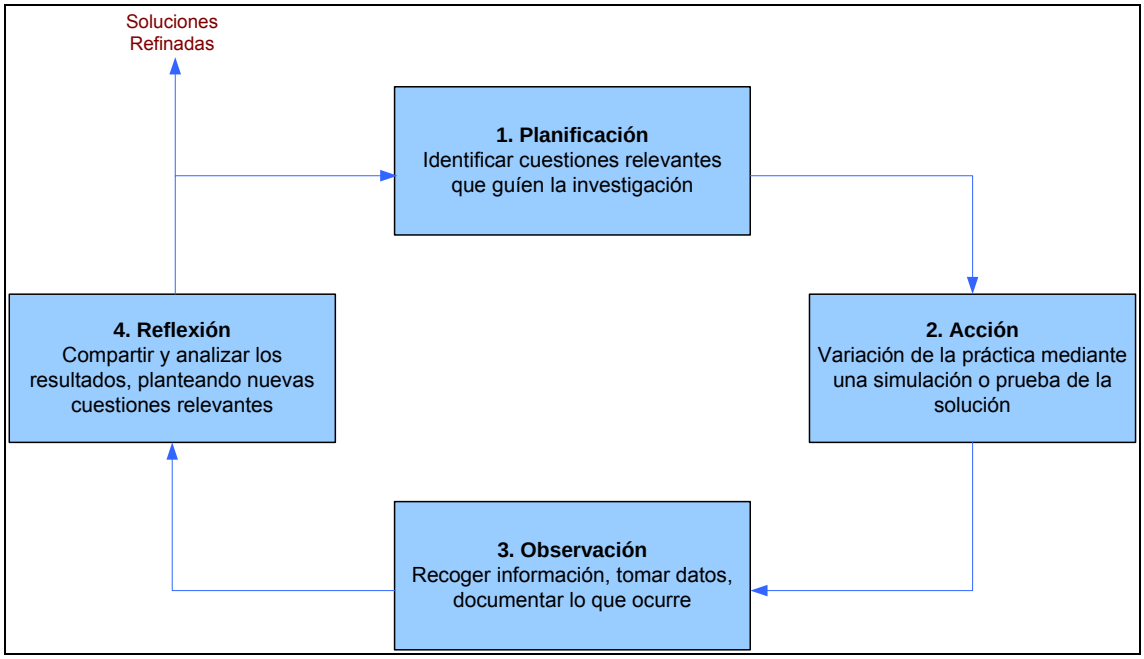


Figura 2-1. Carácter cíclico de Investigación-Acción

Como se muestra en la Figura 2-1, el proceso definido por Investigación-Acción es iterativo, de manera que las soluciones se refinan en cada ciclo, en los cuales se proponen nuevas ideas, que se aplican y se comprueban en el siguiente ciclo. Este ciclo hace que la Investigación-Acción se vea como un proceso reflexivo de aprendizaje y búsqueda de soluciones. El carácter cíclico de Investigación-Acción supone volver a reevaluar o replantear las acciones a seguir ponderando diagnóstico y reflexión.

2.2.1. Aplicación de Investigación-Acción en esta Tesis.

Teniendo en cuenta que el objeto de investigación en esta tesis es el de "Definición de un marco de trabajo basado en MDE para la medición genérica del software" y que se desarrolla en el marco de un proyecto de I+D en colaboración con diversas universidades, se ha considerado la aplicación de métodos cuantitativos y cualitativos.

Desde el punto de vista cualitativo, se ha decidido utilizar la variante participativa de Investigación-Acción como la más adecuada para la definición del marco de trabajo, para lo cual se han considerado los siguientes participantes:

- **Investigador:** el grupo de investigación Alarcos, formado por profesores de la Escuela Superior de Informática de la Universidad de Castilla-La Mancha, en Ciudad Real. La autora de este trabajo es miembro del grupo Alarcos.

- **Objeto investigado:** el objeto investigado es la motivación del proyecto de investigación: "Marco de trabajo basado en MDE para la medición genérica del software".
- **Grupo crítico de referencia (GCR):** constituido en primer lugar por la empresa de desarrollo y mantenimiento de software Indra Software Labs. Esta empresa ha alcanzado el nivel 3 de CMMI y se ha podido aplicar con éxito el marco de trabajo propuesto (véase Capítulo 7). Además, la investigación se enmarca en los proyectos I+D ENIGMAS, INGENIO, ESFINGE y ALTAMIRA en los que colaboran diversas universidades españolas. Por todo ello, el grupo crítico de referencia ha estado constituido por representantes de Indra Software Labs y por representantes de universidades españolas. La autora de este trabajo forma también parte de la empresa Indra Software Labs.
- **Beneficiarios:** serán todas aquellas organizaciones que pueden ser beneficiadas por los resultados del trabajo, es decir, todas aquellas empresas de desarrollo que hacen uso de las medidas para la evaluación de la calidad de la organización. En concreto, la empresa Indra Software Labs, que ha obtenido beneficios importantes en el desempeño de sus mediciones software (véase Capítulo 7).

En la Figura 2-2 se muestra la aplicación de I-A al trabajo de investigación.

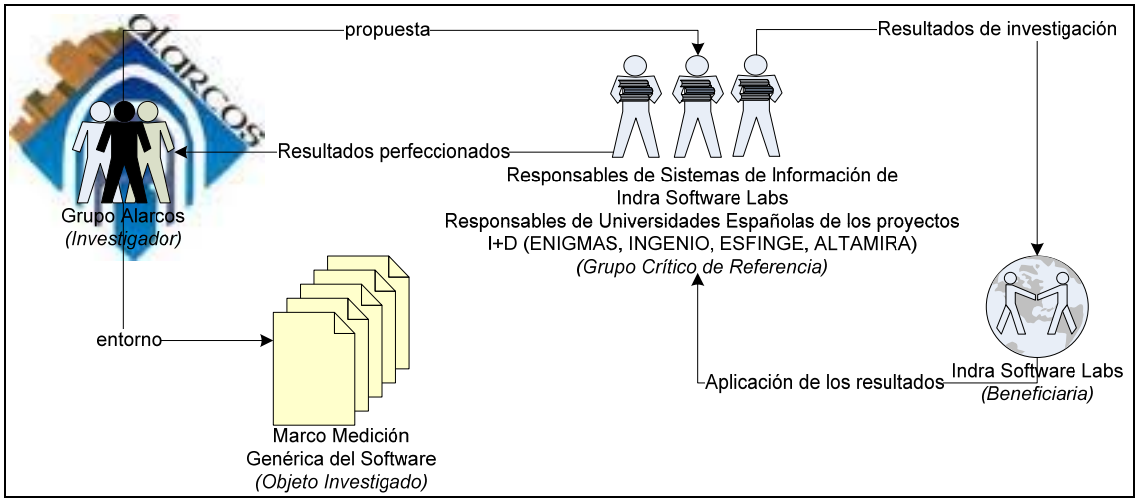


Figura 2-2. Aplicación del método Investigación-Acción al Proyecto de Investigación

La aplicación de IA-SI más evidente es cuando una organización humana interactúa con Sistemas de Información. De hecho, Investigación-Acción es de una las pocas aproximaciones válidas para estudiar los efectos de alteraciones específicas en metodologías de desarrollo y mantenimiento de sistemas en organizaciones humanas. Por tanto, la definición de un marco genérico de medición del software y la definición de modelos de medición es un dominio adecuado para la aplicación de Investigación-Acción. Ello se demuestra con los resultados conseguidos en la aplicación del método:

1. El investigador propuso un marco de trabajo teórico que fue aceptado por el grupo crítico de referencia.
2. El investigador trabajó activamente para que los beneficios fueran mutuos, científicos para el investigador y prácticos para el grupo crítico de referencia.
3. El conocimiento obtenido pudo ser aplicado enseguida.
4. La investigación se desarrolló en un proceso típico cíclico e iterativo combinando teoría y práctica.

La puesta en marcha de Investigación-Acción durante el proceso investigador de este trabajo ha supuesto una realimentación entre el investigador y el grupo crítico de referencia. Se podrían identificar tres grupos de ciclos básicos en la aplicación de investigación en acción para nuestro marco de trabajo que nos han permitido ir obteniendo soluciones cada vez más refinadas generadas de forma participativa.

- Ciclo general inicial.
- Ciclos generales intermedios.
- Ciclos específicos finales.

Estas soluciones se concretaban en componentes del marco conceptual propuesto, definición y validación del lenguaje de medición del software y desarrollo o mejora de los componentes del entorno de Ingeniería del Software. Podemos resumir este proceso en los siguientes ciclos:

1. *Ciclo general inicial*: investigadores y grupo crítico de referencia definieron la problemática general de la medición del software en relación a la gran cantidad de tipos de entidades a medir (**planificación**). Se procedió a la búsqueda de toda la información de posible interés al respecto (**acción**). Su análisis posterior (**observación**) permitió descubrir que el objeto de estudio tenía una complejidad importante dado que debían tenerse en cuenta los múltiples dominios en un proceso de medición. El razonamiento y puesta en común (**reflexión**) entre los investigadores y el grupo crítico de referencia permitió detectar que la solución podría consistir en aplicar un paradigma de Model-Driven (MDE, MDD). En resumen, este ciclo supuso un estudio sobre el paradigma Model-Driven y cómo su aplicación puede ayudar a solucionar la problemática planteada. Como resultado de este ciclo general se identificaron los dos ciclos principales de la investigación (conceptual y tecnológico)

2. *Ciclos generales intermedios*: a partir del ciclo general se identificaron los siguientes grupos cíclicos intermedios:
   a. **Ciclo Conceptual**: ¿Qué es necesario para gestionar de forma genérica la medición del software?; ¿Cómo representar toda la información necesaria para llevar a cabo la medición?; ¿Cómo se aplica el paradigma Model-Driven? ¿Qué es necesario para llevar a cabo la definición de modelos de medición basados en el paradigma Model-Driven?.
   b. **Ciclo Tecnológico**: ¿Qué herramientas software son útiles para gestionar de forma genérica el proceso de medición de software?; ¿Cómo se aplican las transformaciones MDA para conseguirlo?

3. *Ciclos específicos finales*: a partir del momento en que las respuestas anteriores quedaron claras, tanto para los investigadores como para el grupo crítico de referencia, se procedió a realizar ciclos específicos de Investigación-Acción para cada uno de los tres componentes principales. En los capítulos 4, 5 y 6 se muestran los resultados obtenidos y su validación.

Para la definición del lenguaje de modelado de medición del software se ha utilizado un método específico y para la validación de las propuestas se ha hecho uso de métodos empíricos que se describen en los siguientes apartados.

2.3. Método para Desarrollar el DSL.

En esta apartado se describe el método utilizado para el desarrollo de un DSL (*Domain Specific Language*).

Para el desarrollo de un DSL se requiere el conocimiento del dominio y cierta maestría en desarrollo de lenguajes. El proceso del desarrollo se puede facilitar usando las herramientas de DSM (*Domain Specific Modeling*). El desarrollo del DSL consiste en cinco fases (Mernik et al., 2005): decisión, análisis, diseño, implementación (puesta en práctica) y despliegue (ejecución). No es un proceso secuencial ya que, a menudo, fases finales dependen de anteriores, y viceversa. En este apartado se presentan también las etapas propuestas en (Feilkas, 2006) para hacer un DSL usable.

2.3.1. Etapas de Desarrollo de un DSL.

A continuación se presentan las etapas de desarrollo de un DSL propuestas por (Mernik et al., 2005).

2.3.1.1. Decisión.

El diseño, construcción y mantenimiento de los DSLs sin tener en cuenta la herramienta DSM (MS/DSL, eclipse u otras) utilizada en el desarrollo es una tarea costosa. Por tanto, en la fase de diseño, se debe realizar un análisis de costes y beneficios para determinar si es más beneficioso o no, utilizar DSLs en vez de lenguajes de propósito general (*General Purpose Language – GPL*).

Como se muestra en la Figura 2-3, en el desarrollo de un DSL los costes son más altos debido a la complejidad del diseño y a la puesta en ejecución de los mismos. Sin embargo, se reducen los costes de ciclo de vida del desarrollo debido al aumento de la productividad (Christensen, 2003). En este sentido, la inversión en el desarrollo del DSL se compensa por un desarrollo y mantenimiento más económico del software en las últimas aplicaciones (Mernik et al., 2005). Una vez que se ha llevado a cabo la fase de decisión, el siguiente paso es definir el alcance y tamaño del lenguaje, y seleccionar las herramientas apropiadas.

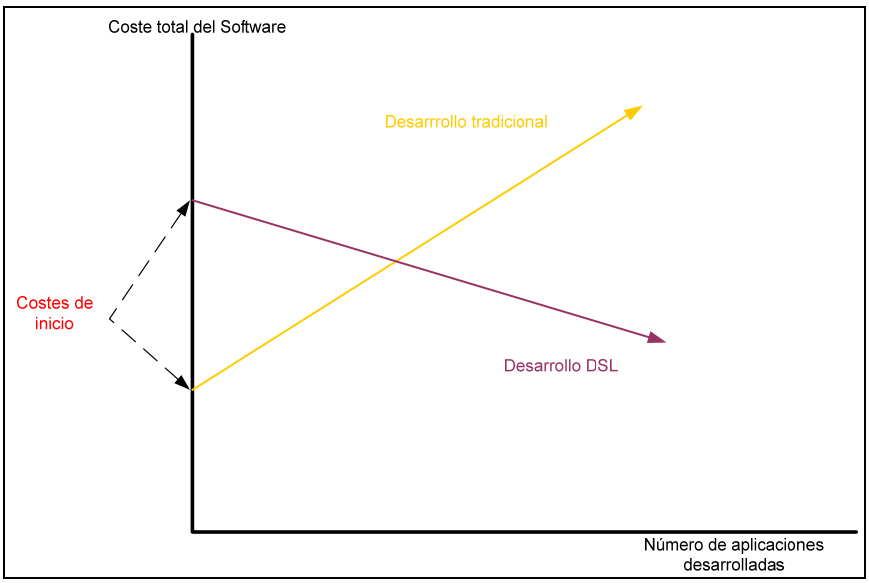


Figura 2-3. Rentabilidad del desarrollo de un DSL (Christensen, 2003)

2.3.1.2. Análisis.

En la fase de análisis del desarrollo del DSL, se identifica el dominio del problema y se recolecta el conocimiento del dominio. Para extraer el conocimiento del dominio se necesitan fuentes implícitas o explícitas, que pueden ser: los documentos técnicos existentes; el conocimiento proporcionado por expertos del dominio, los modelos existentes definidos en idiomas de uso general, y las encuestas sobre cliente. Además, las prácticas de la ingeniería de conocimiento tales como capturar y representar el conocimiento pueden facilitar la elicitación del conocimiento del dominio. Como salida del análisis del dominio se obtiene un modelo del dominio que consiste en una definición del alcance del dominio, de la terminología del dominio, de la descripción de los conceptos del dominio y de las concordancias, de variabilidades y de dependencias entre ellas (Mernik et al., 2005). (Völter et al., 2006) citan que para encontrar un DSL apropiado para un determinado dominio, se debe considerar siempre la cantidad requerida de variabilidad. Para expresar variantes en los modelos, sirve de gran ayuda herramientas que den soporte como los diagramas de la característica (Völter et al., 2006).

2.3.1.3. Diseño

La manera más fácil de diseñar un DSL es basarlo en un lenguaje existente y ampliarlo con las nuevas características que tratan conceptos del dominio. Una vez que la relación entre los lenguajes existentes se haya resuelto, para especificar el diseño antes de la implementación, se debe utilizar un diseñador de DSL (Mernik et al., 2005).

Se distinguen entre diseños formales e informales. En un diseño informal la especificación se da normalmente en lenguaje natural, probablemente incluyendo un conjunto de programas DSL ilustrativos. Un diseño formal consiste en una especificación escrita utilizando una semántica de definición de métodos.

En (Kolovos et al., 2006) se cita que los atributos de calidad de un DSL y de su entorno de soporte (compiladores, IDE) tienen un impacto en los atributos de la calidad del proceso total del desarrollo y del producto resultante. En la fase de diseño del desarrollo del DSL, se deben especificar los requisitos de calidad del DSL y se deben llevar a cabo las compensaciones necesarias. Los requisitos de un DSL son (Kolovos et al., 2006):

- **Conformidad**: los constructores del lenguaje se deben corresponder con los conceptos importantes del dominio.
- **Ortogonalidad:** cada constructor del lenguaje se utiliza para representar exactamente un concepto distinto (entidad de negocio, regla de negocio…) en el dominio.
- **Portabilidad:** las herramientas del DSL deben proporcionar la ayuda y soporte para la gestión y transformación del modelo.
- **Integración:** es esencial integrar las herramientas del DSL con otras instalaciones utilizadas en el proceso de desarrollo. Las herramientas del DSL deben ser extensibles para dar soporte a constructores y conceptos adicionales.
- **Longevidad:** para asegurar el soporte de la herramienta y cuantificar la rentabilidad del DSL, el DSL se debe utilizar por un período de tiempo no trivial. Se asume que el dominio del software persiste por un período de tiempo lo suficientemente largo como para justificar el coste de construcción del DSL y sus herramientas.
- **Simplicidad:** un DSL debe ser tan simple como sea posible para expresar los conceptos del dominio y dar soporte a sus usuarios.

- **Calidad**: el DSL debe proporcionar los mecanismos generales para construir sistemas de calidad que incluyan constructores del lenguaje para mejorar la fiabilidad, seguridad, etc.
- **Escalabilidad:** los constructores del DSL deben poder manejar grandes descripciones.
- **Usabilidad:** los constructores del DSL se deben entender fácilmente.

2.3.1.4. Implementación y Despliegue

Una vez que se ha diseñado un DSL ejecutable, se debe elegir la forma más adecuada de implementación (intérpretes, compiladores, generadores de aplicaciones, etc.). Los compiladores y generadores de aplicaciones proporcionan una sintaxis cercana a la notación utilizada por los expertos del dominio, de modo que facilita el análisis, la verificación y optimización del dominio específico. Sin embargo el esfuerzo necesario para diseñar compiladores y generadores de aplicaciones es muy grande en comparación al beneficio obtenido (Mernik et al., 2005). Normalmente los generadores se utilizan para los aspectos estructurales de un sistema, y los intérpretes se utilizan para los aspectos de comportamiento. Con respecto a los intérpretes, el principal problema para la implementación de un DSL es que los intérpretes son intrínsecamente más lentos que el código generado. Por otra parte, tienen la ventaja que permiten el modificar dinámicamente el código con el cambio del modelo, sin necesidad de reconstrucción alguna. En el caso de los generadores, si se modifica el modelo es necesario regenerar el código.

Con respecto a la generación del código, una idea interesante es combinar código generado y código manuscrito. Para evitar los problemas de la inconsistencia que aparecen durante la modificación del código generado. Otra solución es guardar en archivos separados el código generado automática y manualmente (Völter et al., 2006).

2.3.2. Cómo hacer usable un DSL.

Para hacer un DSL usable se propone en (Feilkas, 2006) las siguientes etapas que deben llevarse a cabo:

- **Definición de una sintaxis abstracta:** la mayoría de las herramientas de DSL permiten la definición de una sintaxis abstracta como un metamodelo. Este metamodelo se define por medio de técnicas de modelado (el meta-metamodelo) similares a los diagramas de clases o diagramas ER.
- **Definición de una sintaxis concreta:** para hacer un lenguaje usable se tiene que definir una sintaxis abstracta. Para cada elemento de un lenguaje existe un icono gráfico (en el caso de los lenguajes gráficos) que representa el elemento del modelo abstracto. Realizada la correspondencia de los elementos del lenguaje con los iconos, se necesita un entorno de desarrollo para proporcionarlo. En el caso de los lenguajes textuales la sintaxis puede ser descrita por una gramática. Una gramática describe tanto la sintaxis abstracta como la concreta por medio de especificación de terminales, no-terminales y reglas de producción.
- **Definición de la semántica:** posiblemente la parte más importante de la especificación del lenguaje es la formulación de la semántica. Se debe dar una descripción informal del lenguaje en lenguaje natural describiendo su dominio, y otra descripción en lenguaje formal.

2.4. Ingeniería del Software Empírica.

Los estudios empíricos son necesarios para comprobar y entender las implicaciones relacionadas con la medición de las entidades software. Esto se consigue con la aplicación de la teoría en el mundo real para obtener datos empíricos. Se pueden llevar a cabo tres tipos principales de estrategias empíricas (Robson, 1993): experimentos, casos de estudio y encuestas. En los siguientes apartados se describen con mayor detalle las estrategias aplicadas en la presente tesis: experimentos y casos de estudio.

2.4.1. Experimentos.

Un experimento es un procedimiento o examen formal mediante el cual se trata de comprobar (confirmar, verificar) una o varias hipótesis relacionadas con un determinado fenómeno, mediante la manipulación de la/s variable/s que presumiblemente son su causa. Un experimento puede variar mucho según las disciplinas, pero persiguen el mismo objetivo: excluir explicaciones alternativas (diferentes a la variable manipulada) en la explicación de los resultados. Este aspecto se conoce como validez interna del experimento, la cual aumenta cuando el experimento es replicado por otros investigadores y se obtienen los mismos resultados.

También pueden realizarse experimentos "en línea", esto significa que la investigación se realiza en el campo de trabajo en condiciones normales (Babbie, 1990). Controlar las situaciones en línea es más complicado, ya que no es posible controlar todos los factores. El objetivo es manipular una o más variables y mantener las otras variables a niveles fijos.

La ventaja de un experimento es que puede determinar las situaciones en las que determinadas afirmaciones son ciertas y puede proporcionar el contexto en el que se recomiendan determinados estándares, métodos y herramientas. Sólo si se realiza adecuadamente el experimento, es posible obtener conclusiones acerca de las relaciones entre la causa y el efecto para la cual se formulan la hipótesis (Wohlin et al., 2000). Los experimentos necesitan planificarse correctamente para que proporcionen resultados útiles y significativos (Juristo y Moreno, 2001).

(Basili et al., 1999) comentan que la experimentación en la Ingeniería del Software es necesaria, pero bastante complicada. Una razón de esta dificultad es la gran cantidad de variables del contexto que hay, que implica que para comprender adecuadamente los resultados de los experimentos, se necesita un mecanismo para explicar los estudios e incorporar los resultados.

Además es importante realizar **réplicas** de los experimentos, porque con los resultados de un único experimento es difícil apreciar si los resultados se pueden generalizar y poder así concluir que sus resultados son válidos. Una estrategia fundamental para realizar estas réplicas es mediante la creación de "paquetes de laboratorio" (*laboratory packages*) (Basili et al., 1999). Estos paquetes recogen información de todo el material experimental, es decir: el diseño experimental, los artefactos, los procesos utilizados en el experimento, los métodos utilizados durante el análisis experimental y las decisiones tomadas.

2.4.1.1. Proceso para la Realización de Experimentos.

El proceso presentado en este apartado se centra en la experimentación, aunque en cualquier estudio empírico se deberían seguir los mismos pasos básicos. La principal diferencia es el trabajo dentro de una actividad específica, por ejemplo, el diseño de un cuestionario difiere

del de un experimento o un caso de estudio, pero todos ellos necesitan diseñarse. De tal manera que, el proceso básico se puede utilizar para llevar a cabo otros tipos de estudios. El proceso para realizar experimentos consta de seis etapas principales (Wohlin et al., 2000), como muestra la Figura 2-4.

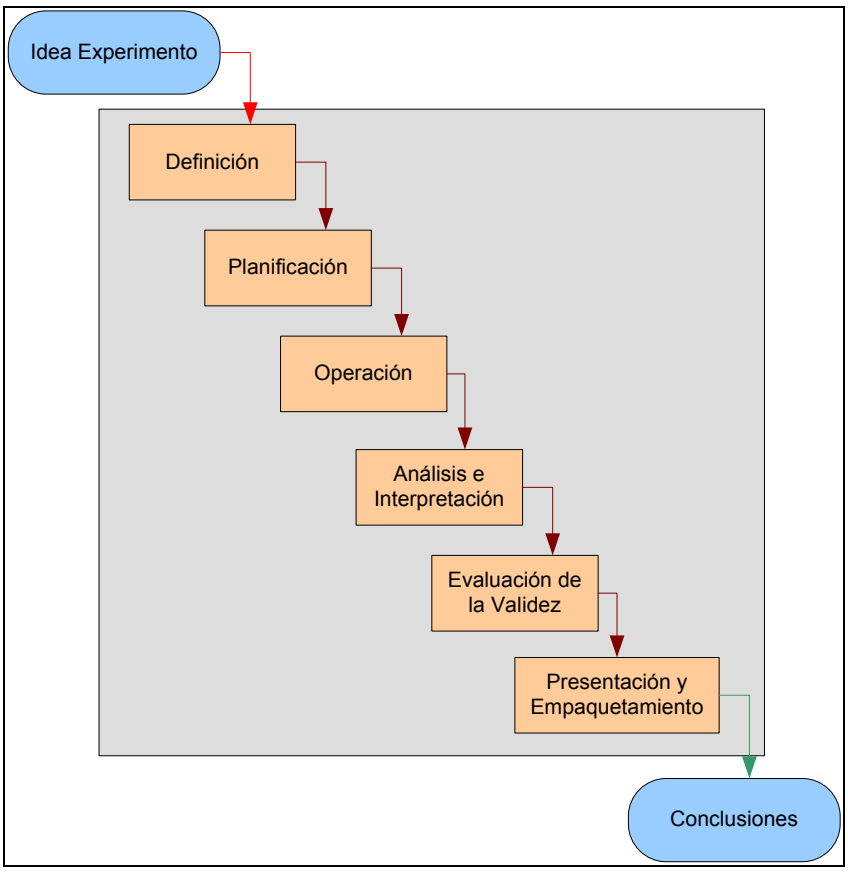


Figura 2-4. Visión general del proceso de realización de experimentos

- **Definición:** en esta etapa se determina la base del experimento, "el porqué" del experimento. La finalidad de esta fase es definir los objetivos de un experimento formulado a partir de un problema a resolver. De acuerdo con las sugerencias de (Briand et al., 2002) y (Lott y Rombach, 1996) se utilizará la plantilla GQM (Goal Question Metric) para recoger los objetivos de un experimento. Los elementos de la plantilla son los siguientes:
  - El **objeto del estudio** es la entidad que se estudia en el experimento, esta puede ser un producto, un proceso, un recurso, un modelo, una medida o una teoría.
  - El **propósito** define cuál es el objetivo o finalidad del experimento.
  - El **enfoque** de la calidad es el efecto primario que se estudia en el experimento.
  - La **perspectiva** sirve para definir el punto de vista desde el cual se van a interpretar los resultados del experimento.
  - El **contexto** es el "entorno" en el cual se lleva a cabo el experimento. El contexto está formado por los sujetos implicados y los artefactos software que se usan en el experimento.
- **Planificación:** Una vez que se ha definido un experimento, el siguiente paso es la planificación. La planificación sirve para establecer cómo se llevará a cabo el experimento. Esta etapa se puede dividir en los siguientes seis pasos:
  - **Selección del contexto**. El contexto del experimento se caracteriza de acuerdo con cuatro dimensiones:

  - Off-line vs. On-line.
  - Estudiantes vs. Profesionales.
  - Simulación vs. Problemas reales.
  - Específico vs. General.
- **Formulación de hipótesis**. La manera de formalizar la definición del experimento es por medio de hipótesis. Para ello se deben formular dos hipótesis: una **hipótesis nula** (H_0) y una **hipótesis alternativa** (H_1).
- **Selección de variables**.
- **Selección de sujetos.**
- **Diseño del experimento**.
- **Instrumentación**. Puede ser de tres tipos: **objetos particulares**, **instrucciones** e **instrumentos de medición**.

- **Operación:** cuando un experimento se ha diseñado y planificado debe llevarse a cabo con el fin de recoger los datos que van a ser analizados. Esto es lo que se llama operación de un experimento, es el punto donde el experimentador se encuentra con los sujetos. En la etapa operacional de un experimento, los tratamientos se aplican a los sujetos. La etapa operacional se divide en tres fases:
  - **Preparación**. Antes de que se inicie el experimento, es necesario buscar el personal que se comprometa a participar en él.
  - **Ejecución**. Es la puesta en marcha del experimento.
  - **Validación de los datos**. Una vez que los datos se han recogido, el experimentador debe comprobar si los datos son razonables y se han recogido correctamente.
- **Análisis e Interpretación:** una vez recogidos los datos empíricos, se deben analizar correctamente. Hay tres elementos principales a considerar cuando se eligen las técnicas de análisis: la naturaleza de los datos que se han recogido, por qué se ejecuta el experimento, y el tipo de diseño experimental. Dependiendo del objetivo del experimento, se pueden utilizar distintas técnicas para probar las hipótesis. El objetivo de probar las hipótesis es comprobar si es posible rechazar la hipótesis nula, H_0, en base a una muestra de una distribución estadística.

  (Pfleeger, 1994) realizó un estudio en el que detalla los diferentes tests estadísticos que pueden aplicarse dependiendo de los objetivos que se pretenden satisfacer. En la Figura 2-5 se muestra el árbol de decisión que es el resultado del estudio desarrollado con el objetivo de facilitar la selección del test estadístico a aplicar en cada caso.

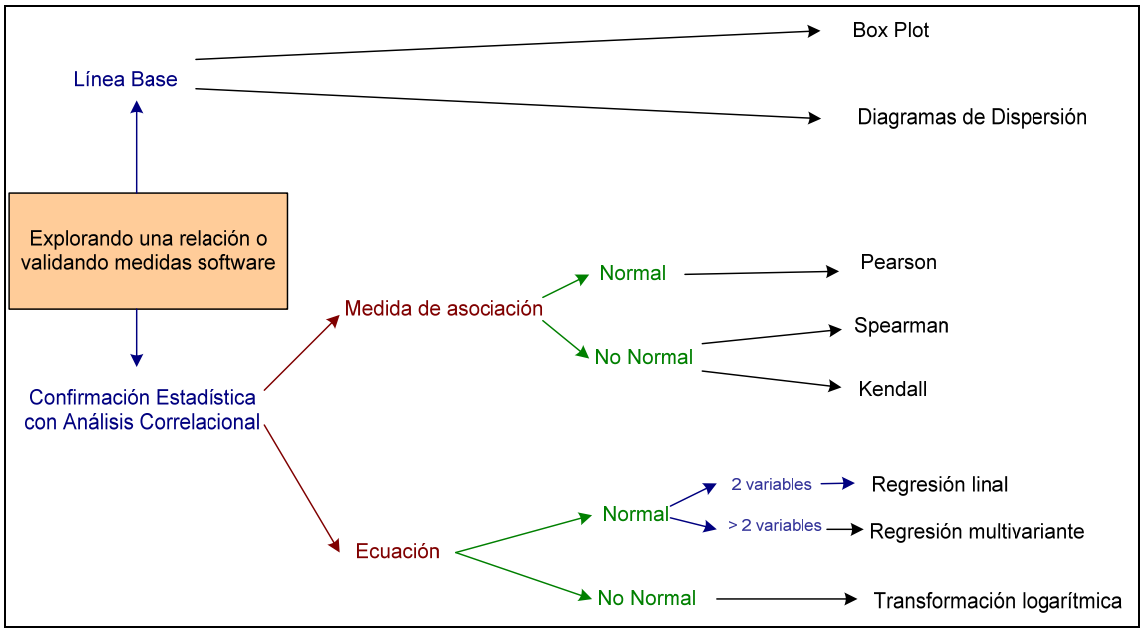


Figura 2-5. Árbol de decisión para técnicas de análisis (Pfleeger, 1994)

En los diferentes estudios empíricos que se han realizado en la presente tesis doctoral se ha utilizado la herramienta SPSS (Statistical Package For Social Science) (2008) para aplicar automáticamente los diferentes métodos estadísticos a los datos recogidos.

- **Evaluación de la validez:** una cuestión fundamental relacionada con los resultados del experimento es la validez de esos resultados. El grado de credibilidad de un experimento depende de la validez de las conclusiones que se obtengan. Al principio de la fase de planificación es importante considerar la cuestión de la validez para planificar adecuadamente la validez de los experimentos. En la Tabla 2-2 se presenta una lista de las diferentes amenazas a la validez de los experimentos (Cook y Campbell, 1979), que se deben controlar para que los resultados en cualquier estudio empírico sean válidos. Todas las amenazas que aquí se recogen no son aplicables a todos los experimentos, pero puede servir como lista de comprobación.

Amenazas a la Validez	Descripción
Validez de la conclusión	Baja potencia estadística, violar las suposiciones de los tests estadísticos, finalización y ratio de error, Fiabilidad de las métricas, Fiabilidad o tratamiento de implementación, Irrelevancias aleatorias en el ambiente experimental.
Validez de constructo	Interpretación pre-operacional inadecuada de los constructores, Sesgo por Mono-operación, Sesgo por Mono-método, Confusión entre constructores y niveles de constructores, Interacción de diferentes tratamientos, Interacción de experimentación y tratamientos, Generalización restringida a través de constructores, Conjeturar hipótesis, Aprensión en la evaluación, Expectativas del experimentador.
Validez interna	Historia, Madurez, Experimentación, Instrumentación, Regresión estadística, Selección, Mortalidad, Ambigüedad sobre el sentido de la influencia causal, Interacción con la selección, Difusión de duplicados del tratamiento, Igualación compensatoria de los tratamientos, Rivalidad compensatoria, Desmoralización resentida.
Validez externa	Interacción de la selección y el tratamiento, Interacción del ambiente y el tratamiento, Interacción de la historia y el tratamiento.

Tabla 2-2. Amenazas a la validez de los experimentos

- **Presentación y Empaquetamiento:** en muchas ocasiones, una vez realizado el experimento, se pretende presentar los resultados. Esto se podría hacer en un artículo para un conferencia, un informe para la toma de decisiones, o como material educacional. Es esencial no olvidar, para la presentación y difusión de un experimento, los aspectos importantes y la información necesaria que necesitamos para llevar a cabo las réplicas y obtener beneficios del experimento y del conocimiento obtenido a través de él.

2.4.2. Casos de Estudio.

Los casos de estudio están relacionados con la investigación acción y se utilizan para monitorizar proyectos, actividades o asignaciones. Los datos recogidos en los casos de estudio se recogen para un objetivo específico de la investigación y se pueden aplicar como una estrategia de investigación comparativa a los resultados de usar otra aproximación. En los casos de estudio el nivel de control es bajo con respecto a nivel de control requerido en un experimento. Los casos de estudios son estudios observacionales (al contrario que en los experimentos que son estudios controlados). Se entiende por estudio observacional aquel que se lleva a cabo mediante la observación de un proyecto o actividad que está en marcha. (Zelkowitz y Wallace, 1998). Existe más información acerca de los casos de estudio se puede en (Robson, 1993; Pfleeger, 1995; Stake, 1995; Yin, 2002).

Por otro lado, los experimentos y los casos de estudio se parecen en que la mayoría de los aspectos considerados en ambos son iguales como es el establecimiento de la hipótesis. La definición de la hipótesis es especialmente importante ya que sirve de guía para medir y analizar los resultados obtenidos en el caso de estudio. En los casos de estudio es necesario crear una base sólida para evaluar los resultados de un caso de estudio, de esta manera se evitan desviaciones y se asegura la validez interna. También es necesario minimizar los efectos de los factores confusos. Se entiende por un factor confuso aquel que hace imposible distinguir los efectos de un factor de los efectos de otro. Esto es importante ya que no se tiene el mismo control sobre un caso de estudio al que se tiene en un experimento. Por ejemplo, puede ser difícil determinar si unos resultados mejores dependen de la aplicación usada o de la experiencia del usuario sobre la aplicación.

Los casos de estudio tienen tanto ventajas como desventajas. En las siguientes tablas (Tabla 2-3 y Tabla 2-4) se muestran algunas de las ventajas y desventajas identificadas por los distintos autores**.**

Autor	Ventajas
(Marcelo y Parrilla, 1991)	• Conectan directamente con la realidad relacionando la teoría con la práctica. • Reconocen la complejidad del caso vinculándolo con su contexto. • Permiten construir una base de datos para múltiples propósitos más allá de la búsqueda en sí. • Conducen a la acción porque las ideas que generan pueden ser utilizadas de manera inmediata por las personas que formen parte del caso. • Producen resultados que son accesibles a muchas audiencias. • Ofrecen comprensión de fenómenos humanos complejos, en si contexto real y deforma holística, donde hay implicadas muchas variables que no se pueden estudiar de forma independiente.
(Stake, 2005)	Será la mejor metodología si la investigación va dirigida a la comprensión de fenómenos complejos para incrementar la convicción sobre los conocimientos del objeto o proceso investigado.
(Eisenhardt, 1989)	• Estrategias de investigación que permite con mayor probabilidad el desarrollo de nuevas teorías y la generación de nuevos conocimientos. • El caso de estudio, al basarse en un proceso de razonamiento inductivo, de "abajo a arriba", a partir del estudio de unas pocas unidades bien seleccionadas, permite que el investigador se ponga en contacto directo con la realidad investigada y que alcance un conocimiento profundo de la misma, favoreciendo en mayor medida la creatividad y la generación de nuevo conocimiento.

Tabla 2-3. Ventajas de los casos de estudio

Autor	Desventajas
(Stake, 2005)	• El caso de estudio no será la mejor opción si el objetivo de la búsqueda es la obtención de leyes generales constituidas por proposiciones explicativas del tipo causa-efecto. • Parece que los estudios de caso son una base pobre para poder generalizar.
(Yin, 2002)	• Normalmente se investiga un fenómeno contemporáneo en su contexto real. • Las fronteras entre el fenómeno y su contexto no están claramente establecidas. • Se han de integrar múltiples fuentes de datos utilizando varias técnicas para la recopilación de datos. • Se tiene que partir de una teoría sólida que guíe el diseño del caso de estudio. • Se tienen que superar las limitaciones de una metodología de búsqueda aún no desarrollada del todo.

Tabla 2-4. Desventajas de los casos de estudio

Para documentar el caso de estudio de la presente tesis se ha utilizado la plantilla presentada en (Brereton et al., 2008), y se ha seguido el método propuesto en (Yin, 2002).

2.4.2.1. Método de Realización de Casos de Estudio.

En este apartado se resumen las etapas para el desarrollo de casos de uso propuestas por Yin (Yin, 2002), las etapas son las siguientes (véase Figura 2-6):

- Diseño y planificación del caso de estudio.
- Preparación de la colección de datos, procedimientos y protocolos para la colección de datos que se va a definir.
- Recopilación de los datos
- Análisis de los datos recogidos
- Presentación de informes

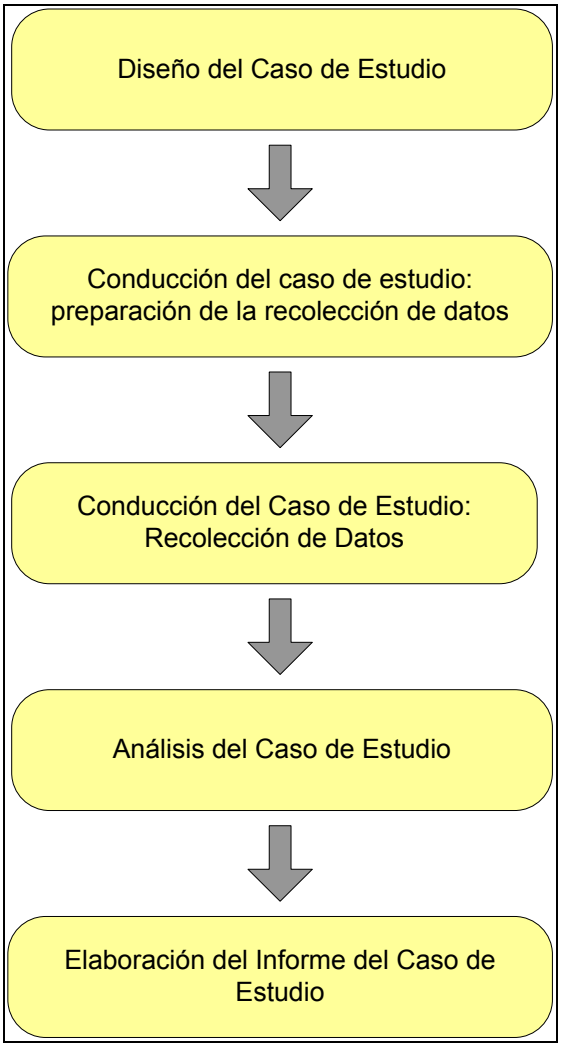


Figura 2-6. Etapas del desarrollo de casos de estudio (Yin, 2002)

2.4.2.1.1. Diseño y planificación del caso de estudio.

El diseño de la investigación consiste en unir los datos que van a recolectarse con las preguntas iniciales del estudio (Yin, 2002). Para realizar un diseño de investigación se proponen los siguientes cinco componentes:

1. ***Preguntas de estudio:*** existen 3 tipos de preguntas dependiendo del objetivo.
   a. ¿Qué? Cuando el objetivo es identificar la hipótesis y proposiciones de la investigación los casos de estudio intentan responder a preguntas de este tipo, por ejemplo "¿qué se puede aprender?"
   b. ¿cómo?, ¿por qué? cuando el propósito es explicativo, es decir tienen el objetivo de establecer relaciones de causa y efecto
   c. ¿quién? ¿dónde? ¿cuántos? ¿cuánto? Para este tipo de preguntas otros métodos de investigación como las encuestas son más adecuados.
2. ***Proposiciones de la misma, si las hay:*** una proposición se centra en lo que se va a estudiar (investigar).
3. ***Unidades de análisis:*** tiene por objetivo definir lo que es el caso de estudio y se deriva de las preguntas en los casos de estudio. Si el caso de estudio está formado de muchas sub-unidades pueden existir varias unidades de análisis. Por ejemplo, si la organización de un hospital es el caso en estudio, podría haber otras subunidades que deben ser analizados, como los servicios clínicos, el personal, etc.
4. ***La relación lógica entre las preguntas y las proposiciones***
5. ***Criterios para interpretar los resultados:*** una vez obtenidos los resultados se necesitan criterios de decisión para analizar los resultados obtenidos.

Seguir de manera metodológica los pasos anteriores facilita el desarrollo de un buen diseño de caso de estudio. Para que un diseño de un caso de estudio se considere bueno debe garantizar las siguientes características:

- Coherencia. Validez de los conceptos
- Validez interna
- Validez externa
- Fiabilidad

En la siguiente tabla se muestra un resumen de cómo conseguir los aspectos deseados de un diseño (Yin, 2002).

Aspecto	Estrategia	Etapas de investigación que aplica
Coherencia. Validez de los conceptos	Usar múltiples fuentes de datos. Establecer la cadena de pruebas. Hacer un informe de cada caso y entregar a los informantes clave para su revisión	Recolección de datos Análisis de datos
Validez interna	Usar técnicas divergentes para el análisis comparativo entre casos: comparación, construcción de explicaciones, uso de series temporales.	Análisis de datos
Validez externa	Usar lógica de replicación en la selección de los casos a estudiar.	Diseño de la investigación (selección de casos)
Fiabilidad	Usar protocolos definidos para el estudio de campo y crear una base de datos para la información recopilada.	Recolección de datos

Tabla 2-5. Estrategias para conseguir los aspectos deseables para un diseño de un caso de estudio

2.4.2.1.2. Preparación de la colección de datos, procedimientos y protocolos.

El primer paso para llevar a cabo esta etapa, es comprender y adquirir las habilidades para ser un buen investigador de casos de estudio, para ello el investigador debe cumplir los siguientes requisitos:

- Un investigador de caso de estudio, debe poder ser capaz de hacer buenas preguntas e interpretar las respuestas.
- Un investigador debe ser un buen oyente y ser objetivo, es decir no dejarse llevar por sus propias ideologías o ideas preconcebidas.
- Un investigador se debe adaptar y ser flexible, de modo que si encuentra nuevas situaciones las vea como oportunidades y no como amenazas.
- Un investigador debe tener una comprensión firme de las cuestiones que se están estudiando.
- Una persona debe ser imparcial a nociones concebidas, incluso aquellas que derivan de la teoría. Así, una persona es más sensible y receptiva a las pruebas contradictorias.

El siguiente paso para los investigadores sería el entrenamiento del caso de estudio. Los investigadores deben tener un profundo conocimiento sobre la metodología del caso de estudio, de la teoría y el caso a estudiar.

El protocolo de un caso de estudio se explica a continuación. Un protocolo contiene el cuestionario de la recogida de los datos y los procedimientos y normas que se deben seguir cuando se utiliza el protocolo. Las partes principales de un protocolo son las siguientes:

- Visión general del proyecto del caso de estudio (objetivo, cuestiones y lecturas relevantes del tema que es objeto de la investigación).
- Procedimientos del campo (presentación de credenciales, el acceso al caso de estudio "sitios", en general fuentes de información, restos de procedimiento)
- Preguntas del caso de estudio (las preguntas específica que el investigador del caso de estudio debe tener en cuenta a la hora de la recogida de los datos)
- Guía para la presentación de un caso de estudio (esquema, formato de los datos, uso y presentación de otro tipo de documentación, e información bibliográfica).

El siguiente paso será la identificación del caso a estudio entre todas las posibilidades; a veces se deben realizar pequeños casos de estudio con el objetivo de poder seleccionar el más adecuado.

Por último, si es necesario, llevar a cabo un caso de estudio piloto con el objetivo de mejorar las preguntas previstas del caso de estudio, el protocolo de recogida de recogida de datos o incluso el diseño.

2.4.2.1.3. Recopilación de los datos.

Los datos de los casos de estudios pueden venir de muchas fuentes. Las seis más importantes se muestran a continuación:

- **Archivos de registros:** registros de un determinado servicio, por ejemplo: registro del número de clientes atendidos durante un determinado período de tiempo; registro de la organización, etc.
- **Entrevistas**: se trata de una de las fuentes de información más importantes para un caso estudio.

- **Observación directa**
- **La observación participante**: esto ocurre cuando el investigador asume un rol dentro del caso de estudio.
- **Artefactos físicos:** los dispositivos tecnológicos, herramientas, instrumentos, etc.
- **Documentación.**

Además, hay tres principios para la recogida de datos de los casos de estudios:

- El uso de múltiples fuentes de datos: datos de dos o más fuentes, pero que convergen en el mismo conjunto de hechos y conclusiones.
- Base de datos del caso de estudio, para almacenar los datos del caso de estudio.
- Una cadena de datos: donde se debe establecer la relación entre las preguntas formuladas, los datos recogidos y las conclusiones.

2.4.2.1.4. Análisis de los datos recogidos.

El análisis de datos consiste en examinar, categorizar, tabular, datos o la combinación tanto de los datos cuantitativos como los cualitativos con el objetivo de hacer frente a la proposición inicial del estudio. Es muy importante definir una estrategia de análisis antes de la preparación de la recogida de datos con el fin de garantizar que los datos sean analizables. Sin una estrategia, es difícil de obtener a una interpretación razonable de los datos. Hay tres estrategias generales:

- Basándose en las estrategias teóricas: la primera y más preferible estrategia es la de seguir una propuesta teórica para llevar a cabo el caso de estudio.
- Pensar en una explicación opuesta: el objetivo es definir y probar explicaciones opuestas.
- Desarrollar la descripción de un caso: consiste en desarrollar un marco de trabajo descriptivo con el cual organizar el caso de estudio.

En cualquiera de estas estrategias se pueden utilizar varias técnicas analíticas que ayudarán a obtener validez interna y externa para el caso de estudio, como son: la coincidencia de patrones, análisis de series de tiempo, modelos lógicos, etc.

2.4.2.1.5. Presentación de informes.

La presentación de los casos de estudio significa presentar los resultados y conclusiones del caso. Un caso de estudio ejemplar debe cumplir las siguientes condiciones:

- El caso de estudio debe ser significativo.
- El caso de estudio debe ser completo
- El caso de estudio también deben mostrar pruebas suficientes
- El caso de estudio debe tener en un atractivo de manera que atraiga al lector

2.4.3. Aplicación de los métodos cuantitativos.

La aplicación de los métodos cuantitativos presentados se ha centrado en dos tipos de validaciones empíricas:

- **Experimentos:** para evaluar la usabilidad y modificabilidad del lenguaje de modelado de medición del software. Este apartado se explica en más detalle en el Capítulo 5.
- **Casos de estudio:** para validar de manera práctica el uso del entorno de medición genérica de software en diferentes dominios. Se explican con más detalle en el Capítulo 7.

2.5. Revisiones Sistemáticas.

Barbara Kitchenham presenta en (Kitchenham, 2004) un método para la realización de revisiones sistemáticas en el contexto de la Ingeniería del Software. Antes de esta propuesta en la Ingeniería del Software no se disponía de ningún método eficiente para realizar estudios exhaustivos, por lo que éste es una buena solución para paliar esta carencia.

Una revisión sistemática de la literatura es una forma de identificar, evaluar e interpretar todas las investigaciones relevantes disponibles a una pregunta de investigación particular, un área temática o un fenómeno de interés. La mayor parte de las investigaciones comienzan con algún tipo de revisión de la literatura existente. Sin embargo, a menos que esa revisión sea rigurosa e imparcial, tendrá poco valor científico. Esto incrementa la posibilidad de detectar más efectos reales que los que pueden detectar revisiones de menor dimensión. Ésta es la principal motivación para emprender revisiones sistemáticas. De esta forma, las revisiones sistemáticas sintetizan el trabajo existente de manera imparcial. La propuesta de Kitchenham para la realización de una revisión sistemática involucra diferentes actividades independientes, para lo cual el método propone tres etapas fundamentales (ver Tabla 2-6).

Etapa 1	**Planificación de la revisión**
	• Identificación de la necesidad de la revisión • Desarrollo de un protocolo de revisión
Etapa 2	**Desarrollo de la revisión**
	• Identificación de la investigación • Selección de los estudios primarios • Evaluación de la calidad del estudio • Extracción y monitoreo de datos • Síntesis de datos
Etapa 3	**Publicación de los resultados**

Tabla 2-6. Método para la realización de revisiones sistemáticas de Kitchenham (2004)

2.5.1. Aplicación de las Revisiones sistemáticas.

En esta tesis las revisiones sistemáticas se han utilizado para llevar a cabo el estado del arte de los lenguajes de definición de MDA presentado en el Capítulo 3. El procedimiento se muestra en la Figura 2-7 y el protocolo seguido se describe en el anexo C.

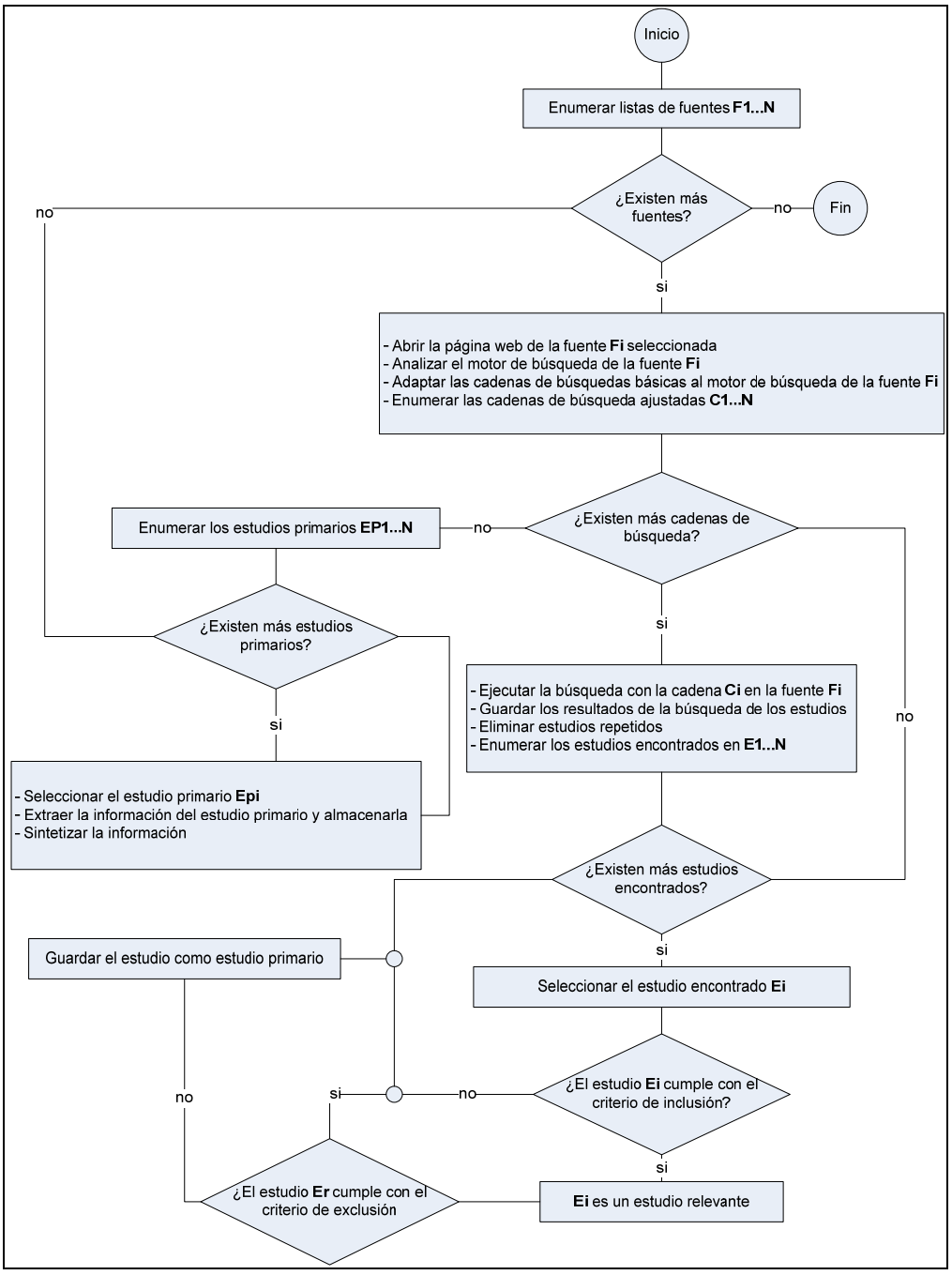


Figura 2-7. Procedimiento para obtener los estudios primarios y sintetizar su información

“El progreso y el desarrollo son imposibles si uno sigue haciendo las cosas tal como siempre las ha hecho” (Wayne W. Dyer)

3. Estado del Arte.

En este capítulo se presentan las diversas propuestas y trabajos relacionados con los objetivos de la Tesis.

En el primer apartado se presenta el paradigma MDE (Model Driven Engineering). En el segundo apartado se describe el modelado de dominios específicos. En tercer lugar se aborda el campo de la medición del software en términos generales. En el apartado cuarto se presentan propuestas relacionadas con la medición de modelos software. Por último, en el apartado quinto se detallan los antecedentes de la presente tesis.

3.1. Model Driven Engineering.

MDE (Model Driven Engineering) es un paradigma de desarrollo de software que promueve el uso de modelos con diferentes niveles de abstracción. La iniciativa MDE más conocida es MDA, que es una marca registrada de la OMG.

A continuación, en base a los resultados obtenidos con una revisión sistemática de la bibliografía (Véase Anexo C) se presenta: una breve introducción a MDE, los objetivos, la aplicación de un proceso MDE, los estándares y herramientas de MDE y la diferencia entre los conceptos MDD, MDE y MDA. Por último se presenta el modelado específico de dominio (Domain Specific Modeling).

3.1.1. Introducción.

Actualmente, el desarrollo de los sistemas software es cada vez más complejo y distribuido. Por esto, los desarrolladores están obligados a tener conocimiento de una amplia gama de tecnologías. A pesar de estos cambios en el mercado, los usuarios finales esperan resultados rápidos, fiables y escalables (Cook, 2004).

La tendencia dominante actual en el desarrollo software es que los modelos representen el problema software con un alto nivel de abstracción, tanto los elementos del sistema como las relaciones dentro de él. A partir de ellos (desarrollados en las primeras fases) se lleva a cabo el desarrollo del software. Los modelos por tanto se convierten en artefactos esenciales en MDE (Bézivin, 2004). Los modelos sirven como entrada y salida en todas las fases del desarrollo del sistema hasta que se obtiene el sistema software (Balasubramanian et al., 2006).

MDE combina los siguientes conceptos (Schmidt, 2006; Selic, 2006):

- **Lenguajes de dominio específico (DSL[4], Domain Specific Language).** Utilizando los lenguajes de modelado se consigue un alto nivel de abstracción en las especificaciones, de manera que la solución está más cercana al dominio del problema. Los lenguajes de modelado formalizan la estructura de la aplicación, el comportamiento y los requisitos dentro de un dominio particular. La descripción de estos lenguajes se hacen utilizando metamodelos y sus modelos se definen a partir de este metamodelo. Los metamodelos definen los elementos y las relaciones entre ellos dentro de un dominio concreto. Mediante un DSL se consiguen notaciones de modelado distintas para cada tipo de sistema, las cuales están definidas formalmente por su metamodelo. De esta manera, el ingeniero del software tiene herramientas específicas para cada tipo de sistema, lo cual le permite modelarlos de una manera más detallada y de acuerdo al dominio al que pertenecen. Para más información véase el apartado 3.1.6.
- **Motores de transformación y generadores.** Utilizando la tecnología para unir los conceptos semántica e implementación se consigue un alto nivel de abstracción en la automatización, de manera que se permite obtener la implementación (código generado) a partir de una especificación (modelo). Los motores de transformación y generadores analizan ciertos aspectos de los modelos y después crean varios tipos de artefactos, tales como código fuente, entradas de simulación, descripciones de uso XML (Extensible Markup Language), o representaciones alternativas de dicho modelo. Mediante los motores de transformación se facilita la evolución de modelos, transformando de unos modelos a otros, según la reglas de transformación definida entre metamodelos. En

[4] Los conceptos DSL y DSML (Domain Specific Modeling Language) se refieren a lo mismo, por tanto se pueden utilizar indistintamente.

MDE cualquier concepto debe ser modelado. De esta manera, cualquier cambio o nueva propiedad del sistema debe ser mostrado en su modelo correspondiente. Con MDE, la parte de escritura de código es una parte más del proceso de construcción de sistemas (quizás la menos importante), la cual se sugiere que se realice automáticamente.

Las herramientas MDE utilizan estos conceptos para dar soporte a la evolución del software, tanto en su lógica como en su tecnología.

Los efectos de la práctica de estas dos tendencias se traducen en una mejor calidad del producto y un aumento en la productividad del desarrollo. A pesar de ello, antes de introducir MDE en una empresa de desarrollo software hay que considerar con detenimiento los costes de las nuevas herramientas, experiencia del personal y la gestión.

3.1.2. Objetivos de MDE.

La principal motivación del paradigma MDE es mejorar la productividad. Actualmente en las empresas de desarrollo cada vez existe más diversidad en las plataformas y tecnología utilizadas, y la creación de aplicaciones compatibles con las nuevas tecnologías se convierte en una labor muy compleja, quedándose las aplicaciones obsoletas con respecto a las nuevas tendencias tecnológicas del mercado. MDE tiene el objetivo de aumentar la rentabilidad en una empresa desde el punto de vista del esfuerzo en el desarrollo de software. Este beneficio se presenta en dos formas básicas (Atkinson y Kühne, 2002):

- Mediante la mejora de la productividad a corto plazo de los desarrolladores. Esto se consigue aumentando las funcionalidades ofrecidas por los artefactos software.
- Mediante la mejora de la productividad a largo plazo de los desarrolladores. Reduciendo el porcentaje de los artefactos software que se quedan obsoletos.

El objetivo principal de la mayoría de las herramientas MDE es garantizar la primera ventaja: permitir generar nuevos artefactos de software a partir de modelos, apoyando así a los desarrolladores en su productividad. Sin embargo, después del proceso inicial de construcción no ofrecen soporte en la gestión del ciclo de vida del artefacto de software y los cambios tienen que hacerse en el código fuente generado o en partes del modelo, lo que dará lugar a problemas de todo tipo.

Debido a la rápida evolución de las tecnologías en el desarrollo software de las empresas, la segunda ventaja se hace cada vez más importante. Según pasa el tiempo los productos software se van quedando obsoletos y el coste de adaptación es muy alto. De acuerdo con (Atkinson y Kühne, 2002) una segunda y estratégica ventaja importante del MDE es la reducción de la sensibilidad de los artefactos primarios al cambio.

En resumen, el objetivo de MDE es aumentar tanto la productividad a corto plazo (cantidad de funcionalidad presentada en un artefacto software), como la productividad a largo plazo (reducción de la sensibilidad de los artefactos de software para los cambios de personal, requisitos, plataformas de desarrollo y despliegue).

3.1.3. Aplicación de un proceso MDE.

La idea que promueve MDE es usar modelos a diferentes niveles de abstracción para desarrollar los sistemas. De esta manera, la principal actividad de los desarrolladores MDE es diseñar modelos, como los que usaban para desarrollar código, pero ahora guiados por una metodología.

La ventaja de tener un proceso MDE es que éste debe definir claramente cada paso a dar, forzando a los desarrolladores a seguir la metodología definida. Debe especificar la

secuencia de modelos a desarrollar, y cómo derivar un modelo a partir de otro del nivel de abstracción inmediatamente superior. Proporcionando a los desarrolladores una metodología como ésta, podrán saber en cualquier momento a lo largo del proceso de desarrollo, qué se debe hacer en cada paso de desarrollo y cómo conseguirlo.

La aplicación de un proceso MDE se muestra en la Figura 3-1. El sistema en desarrollo es descrito en primer lugar por un modelo a un alto nivel de abstracción, esto es, ignorando cualquier tipo de dependencia de la plataforma. Posteriormente, se deben realizar una serie de refinamientos interactivos (transformaciones) con el objetivo de hacer el sistema más específico de la plataforma en cada paso.

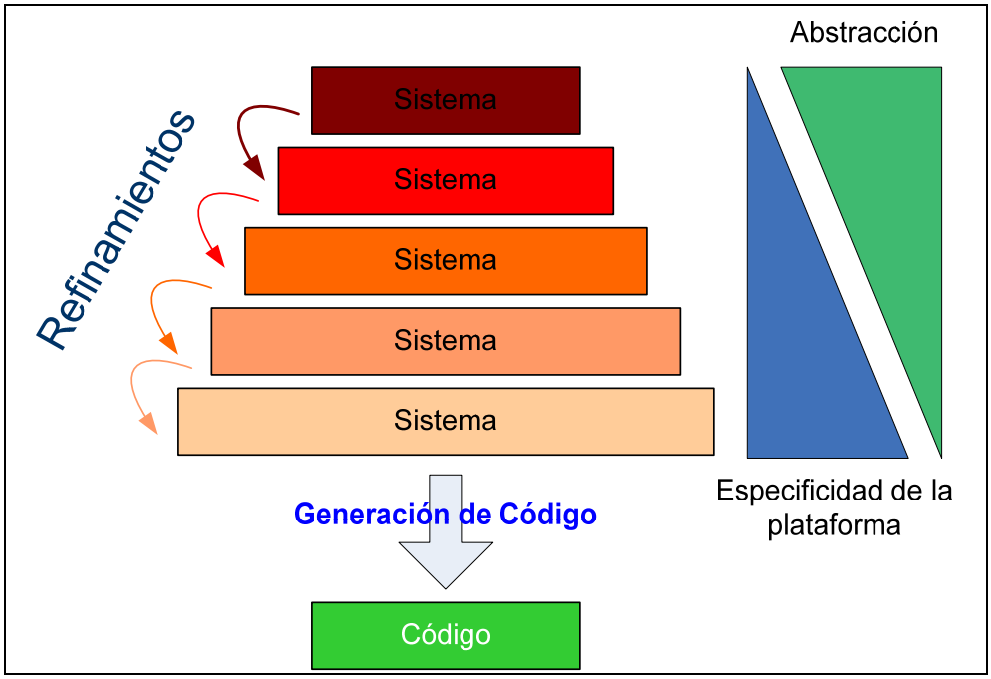


Figura 3-1. Aplicación de un proceso MDE.

Una de las ventajas más importantes de usar el proceso MDE es su adaptabilidad a los cambios. Cuando un cambio ocurre, bien en el mayor nivel de abstracción o bien en el menor nivel de abstracción, su impacto está localizado y las partes que no están afectadas por el cambio se reutilizan.

Desde el punto de vista de la medición del software la aplicación el proceso MDE se puede ver como un conjunto de transformaciones de modelos, a partir de un nivel de abstracción alto hasta un nivel más específico. En el nivel más abstracto estarían los requisitos, y en el nivel más específico estaría resultado de la medición.

3.1.4. Estándares y Herramientas de MDE.

Hoy en día los estándares más significativos de MDE son MDA de OMG, Software Factories de Microsoft y Model Integrated Computing (MIC) (Véase Figura 3-2).

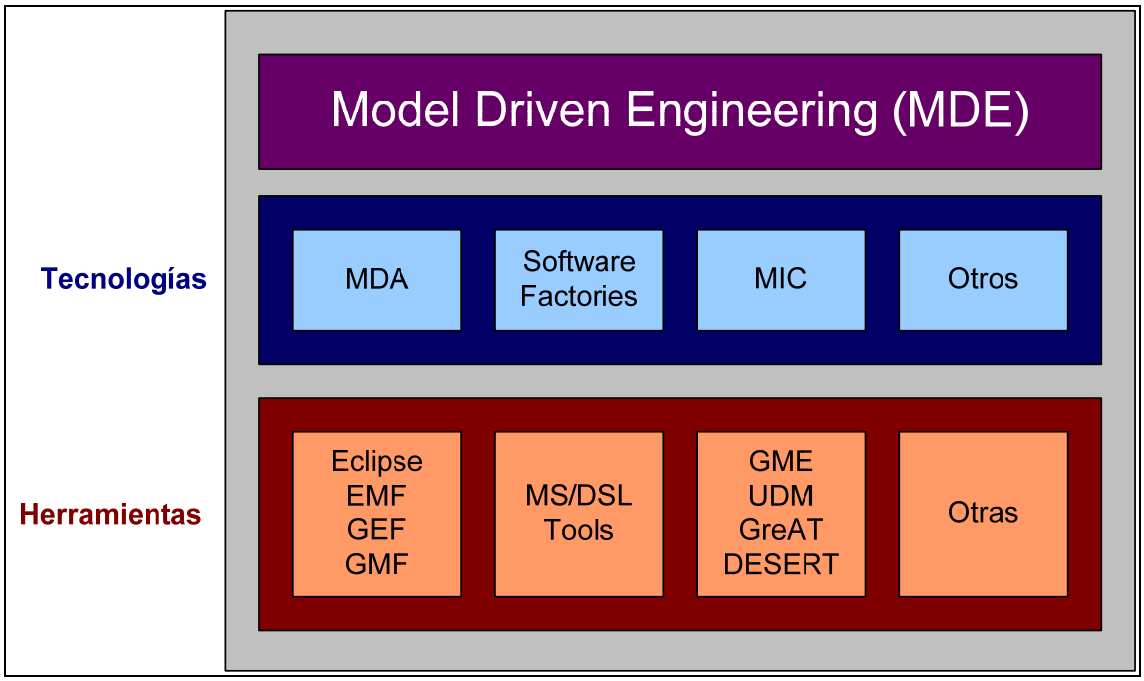


Figura 3-2. Model-Driven Engineering

El estándar MDA es el nombre que la OMG ha dado a su nueva arquitectura. MDA, se basa en el lenguaje de modelado unificado UML, el lenguaje de modelado tan popular de la OMG. De hecho, la nueva arquitectura de la OMG surge en un nivel de abstracción superior en un esfuerzo de encontrar un mecanismo universal de integrar aplicaciones. La base del estándar MDA de la OMG es UML y Meta Object Facility (MOF) (OMG, 2003b) que hacen posible modelar no sólo UML sino también otros metamodelos, incluyendo CORBA y CWM (Common Warehouse Meta Model) (OMG, 2003a). Al mismo tiempo, la OMG ha estandarizado el intercambio de metadatos mediante la propuesta XML Metadata Interchange (XMI) (OMG, 2005b).

UML, MOF y XMI son tres tecnologías clave para el desarrollo de software bajo el enfoque de MDA. Usadas de forma conjunta proporcionan grandes ventajas que hacen que los modelos sean más claros y fácilmente mantenibles. Estas tecnologías definen una forma estándar de almacenar e intercambiar modelos, bien sean de negocio o de diseño. Esto permite a los constructores de herramientas CASE establecer un lenguaje común que proporcione grandes beneficios para el desarrollador. Una vez que las herramientas implementen estos estándares se pueden automatizar y estandarizar numerosos procesos del desarrollo que simplificarán muchas tareas, que antes eran manuales o se realizaban de forma automática por medio de alguna característica propia de la herramienta, que en muchos casos hacía imposible el intercambio de información con otras herramientas del mercado.

Otros estándares que forman parte de MDA son Object Constraint Language (OCL) (OMG, 2002b) que permite añadir restricciones a los modelos y el Software Process Engineering Metamodel (SPEM) para modelar procesos software.

Al contrario que MDA, MIC emplea DSLs para representar los elementos del sistema y sus relaciones, y las transformaciones a los artefactos específicos de la plataforma (Balasubramanian et al., 2006). Por otro lado, las Factorías Software (*Software Factories*), un nuevo paradigma dentro del desarrollo del software, es un entorno de desarrollo preparado para dar soporte al desarrollo rápido de un tipo específico de aplicación. Las Software Factories hacen uso del DSM utilizando DSLs para modelar varios aspectos de un sistema de software (Greenfield, 2004). Microsoft e IBM desarrollan sus propias tecnologías: Microsoft desarrolla las MS/DSL Tools utilizan el estándar Software Factories e IBM, contribuye con Eclipse Modeling Framework (EMF) (Eclipse, 2010b), Graphical Editing Framework (GEF) (Eclipse,

2007) y Graphical Modeling Framework (GMF) (Eclipse, 2010a) que utilizan el estándar de MDA (OMG, 2003c). Generic Modeling Environment (GME), Model Management tool suite (UDM), Model Transformation tool suite (GReAT) y Design Space Exploration tool suite (DESERT) son los componentes genéricos de la arquitectura MIC (ISIS, 2010), y están desarrollados por ISIS (Institute of Software Integrated Systems) de la universidad de Vanderbilt. Por último las herramientas EMF/GMF y MS/DSL Tools ofrecen diferentes propuestas para dar soporte al modelado específico de dominio.

Para abordar la presente tesis se utiliza el estándar MDA que proporciona la base conceptual y tecnológica necesaria para llevar a la práctica el objetivo principal de la tesis. De acuerdo a MDA, la medición del software sobre un dominio concreto se puede considerar como una transformación de un modelo de medición en otro extendido con los resultados de la medición, tomando como entradas el propio modelo de medición y el modelo de dominio en el que se está llevando a cabo la medición. De esta manera se podrá realizar una medición independientemente de cuál sea el dominio sobre el que se realiza.

3.1.5. MDE, MDD y MDA.

Existe cierta confusión acerca de la relación entre los conceptos MDA, MDD (Model-Driven Development) y MDE, a continuación se da una breve descripción de cada uno de ellos desde el más específico al más genérico:

- **MDE:** es el paradigma que ofrece una propuesta del uso de modelos frente a la incapacidad de los lenguajes de tercera generación para disminuir la complejidad de las plataformas y expresar conceptos de dominio de manera eficaz (Schmidt, 2006). Con esta definición se puede entender que MDE es la evolución de las herramientas CASE.
- **MDD:** es el paradigma MDE pero aplicado al desarrollo (no mantenimiento). Se trata de una propuesta para el desarrollo de software basado en el modelado del sistema software y su generación a partir de los modelos. Sólo proporciona una estrategia general a seguir en el desarrollo de software, pero no define ni técnicas a utilizar, ni fases del proceso, ni ningún tipo de guía metodológica.
- **MDA:** es la iniciativa de OMG que propone definir un conjunto de normas para especificar tecnologías interoperables con la finalidad de llevar a cabo desarrollo dirigido por modelos impulsado con transformaciones automatizadas.

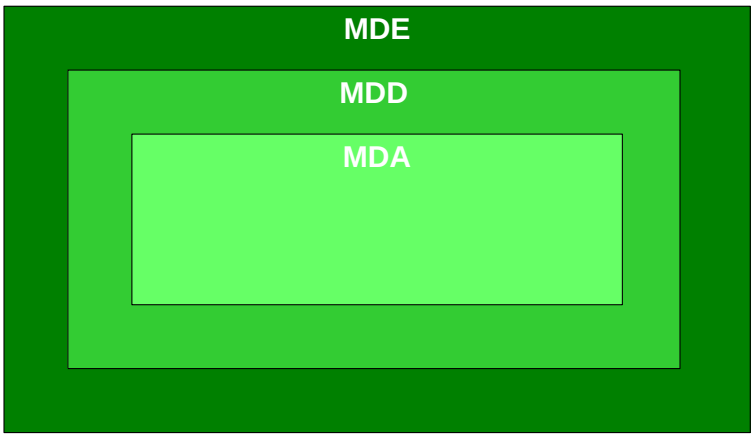


Figura 3-3. Representación de iniciativas Model-Driven

En resumen, tal como se observa en la Figura 3-3, MDD es un paradigma de desarrollo que considera los modelos software como principal elemento del proceso de desarrollo. A partir de estos modelos se genera, normalmente, de manera semiautomática el código. MDA es la visión particular que tiene OMG de MDD y por lo tanto se basa en el uso de los estándares OMG. Por tanto se puede ver MDA como una instanciación de MDD con la aplicación de unas tecnologías.

En cambio, MDE sería un superconjunto de MDD ya que se trata de una Ingeniería de Model-Driven. Por tanto MDE va más allá de las actividades de desarrollo e incluye también otros procesos adicionales de la Ingeniería del Software como la evolución del sistema, siempre basándose en el uso de modelos

3.1.6. Domain Specific Modeling.

El modelado específico de dominio (Domain Specific Modeling –DSM) es una metodología de Ingeniería del Software para desarrollar y diseñar sistemas. Esta metodología implica el uso sistemático de un lenguaje específico de dominio (DSL) para representar el dominio. El nivel de abstracción dentro de los lenguajes DSM es muy alto, más que los lenguajes de propósito general como puede ser UML, lo que significa un menor esfuerzo ya que se centra en el dominio específico de un determinado sistema. La metodología DSM incluye la generación automática de código ejecutable a partir de los modelos DSM, esta es más fiable que la manual ya que reduce el número de errores en los programas finales, mejorando así la calidad.

Como se ha introducido en el apartado 3.1.1, un DSL es un lenguaje específico de dominio que está diseñado para utilizarse en dominios o problemas específicos, a diferencia de los lenguajes de propósito general (General Purpose Language -GPL-). Tienen un mayor nivel de abstracción que los lenguajes base y expresan los conceptos de dominio específico en un nivel de representación más alto. Debe considerarse como un lenguaje de tamaño pequeño, muy centrado en resolver algunos problemas claramente identificables a los que debe enfrentarse un analista, arquitecto, responsable de pruebas o administrador del sistema. El empleo de DSLs es una técnica adecuada para desarrollar software en menor tiempo y con una mejor calidad (Feilkas, 2006). Algunos ejemplos de DSLs conocidos son SQL[5] (Structured Query Language) o HTML[6] (HyperText Markup Language).

La construcción de un DSL no es tarea fácil, se requiere experiencia en el lenguaje y conocimiento sobre el dominio específico. Tal como se ha descrito en el Capítulo 2, es importante seguir un enfoque sistemático para desarrollar un DSL. En esta tesis se ha seguido la metodología de desarrollo propuesta por Mernik (Mernik et al., 2005). Por otro lado es necesario que el DLS sea usable (Feilkas, 2006).

3.1.6.1. Beneficios y Riesgos de los DSLs.

Los beneficios que puede aportar un DSL son los siguientes:

- Permiten que las soluciones estén expresadas a nivel del dominio; dominio que los expertos pueden entender, validar, modificar e incluso desarrollar modelos de dicho dominio mediante DSLs (Furtado, 2006).

[5] SQL es un lenguaje de consultas estructurado que proporciona una interfaz a los sistemas de gestión de base de datos.

[6] HTML es un lenguaje de marcas para la creación de páginas Web.

- Son concisos, no necesitan estar excesivamente documentados y se pueden reutilizar para distintos fines (Furtado, 2006).
- Son lenguajes pequeños y muy centrados en las tareas específicas. Tienen muy bien definida la semántica lo que permite la facilidad de uso (Greenfield, 2004).
- Permiten a los desarrolladores separar previamente las actividades de desarrollo del entorno software. De esta manera, pueden centrarse en una única tarea y así obtener mejores resultados y un proceso de desarrollo más eficiente (Czarnecki y Eisenecker, 2000).
- Mejoran la productividad (Christensen, 2003), mantenibilidad (Greenfield, 2004), fiabilidad (Christensen, 2003), portabilidad (Furtado, 2006) y reutilización de los artefactos software.
- Permiten la expresión, validación y optimización de conceptos en un nivel de abstracción del dominio del problema (Furtado, 2006).
- Facilitan la construcción de casos de prueba complejos, mejorando así la prueba del software (Furtado, 2006).

Por otro lado, existen ciertos riesgos asociados a los DSLs:

- Alto coste de diseño, implementación y mantenibilidad de los DSL y las herramientas (Greenfield, 2004).
- Alto coste en la migración del desarrollo del software convencional al desarrollo del software basado en DSLs (Greenfield, 2004; Furtado, 2006).
- Dificultad en encontrar el ámbito concreto de un DSL (Furtado, 2006). Los DSLs están limitados en su alcance (se refieren a un dominio particular) y capacidad (carecen de características que son básicas en los lenguajes de propósito general) (Fowler, 2006).
- Difícil equilibrio entre los constructores de lenguajes de programación específicos de dominio con los de propósito general (Furtado, 2006).
- Perdida de eficiencia en comparación con el código fuente escrito a mano (Christensen, 2003; Furtado, 2006).

3.1.6.1.1. Herramientas para Modelar un DSL.

Para desarrollar un DSL es de gran ayuda la utilización de una herramienta que de soporte al modelado. En este apartado, se presenta una breve comparativa entre las herramientas industriales más importantes de MDE que permiten modelar un DSL: Microsoft DSL Tools[7] y Eclipse[8].

En la Tabla 3-1 se presenta una comparativa llevada a cabo por Pelechano et al (Pelechano et al., 2006) en la que se comparan las facilidades básicas que cada herramienta MDD (MS/DSL Tools y Eclipse) proporciona.

[7] http://www.microsoft.com/

[8] http://www.eclipse.org

Criterio	MS/DSL Tools	Eclipse
Metamodelado	Notación del propietario (modelo de dominio)	EMF (Ecore)
Repositorio	Ficheros XML	XML, XMI
Notación Gráfica	Manipulación directa con ficheros XML	GMF
Model a Model	No dispone de una técnica explícita	mediniQVT, ATL, etc.
Model a Texto	Lenguaje plantilla del propietario	MOFScript, etc.
Validación del Modelo	Validadores pragmáticos	*Framework* de validación (Proyecto Tecnológico EMF)

Tabla 3-1. DSL Tools vs. Eclipse (Pelechano et al., 2006)

Además, en el experimento realizado por (Pelechano et al., 2006) con un grupo de alumnos en el que se compararon ambos lenguajes, se obtiene que Eclipse tiene una mayor aceptación por parte de los alumnos que Microsoft MSL Tools.

En segundo lugar, en la Tabla 3-2 se presenta una comparación de las etapas de desarrollo de un DSL para cada una de las herramientas (Özgür, 2007).

	Etapa	MS/DSL Tools	Eclipse EMF/GEF/GMF
1	Definición del modelo de dominio (*Domain Model Definition*)	Creación del fichero .dsl	Creación de lo ficheros .ecore y ecore_diagram
2	Definición gráfica (*Graphical Definition*)	Creación del fichero .dsl.diagram	Creación del fichero .gmfgraph
3	Definición de la herramienta (*Tooling Definition*)	La herramienta editor (Editor Tools) se define vía *DSL Explorer Windows*	Creación del fichero .gmftool
4	Correspondencia con los elementos del diagrama (*Diagram Element Mappings*)	La correspondencia se define vía *DSL Details Windows*	Creación del fichero .gmfmap
5	Generación del código (*Code Generation*)	Todas las plantillas de texto (fichero .tt) deben transformarse después de cada modificación	El fichero .gmfgen debe crearse después de cada modificación

Tabla 3-2. Comparación de las etapas en el diseño de DSL (Özgür, 2007)

Basándonos en las comparativas previas y en el hecho de que Eclipse se trata de una plataforma de código abierto con la posibilidad de integrar multitud funcionalidades (plugin), la herramienta seleccionada para desarrollar un DSL para la medición del software ha sido el proyecto GMF de Eclipse, que proporciona el soporte necesario para desarrollar editores gráficos basados en EMF y GEF.

3.2. Medición del Software.

La medición del software se ha convertido en un aspecto fundamental de la Ingeniería del Software (Fenton y Pfleeger, 1997). Está demostrando ser muy eficaz en la construcción de sistemas de predicción de alta calidad para grandes proyectos de bases de datos (MacDonell et al., 1997) en la comprensión y mejora de los proyectos de desarrollo y mantenimiento del software (Briand et al., 2002), en la evaluación y garantía de calidad de sistemas, evidenciando las áreas problemáticas (Champeaux, 1997), en la determinación de mejores prácticas de trabajo con el fin de ayudar a los usuarios e investigadores en su trabajo (Champeaux, 1997). Además, las medidas software ayudan en la evaluación y en la institucionalización de la Mejora del Proceso Software (Software Process Improvement) en 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 (ISO/IEC, 2004) 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 medidas que intentan dar información para responder a estas preguntas.
- **Practical Software Measurement (PSM):** la metodología PSM (McGarry et al., 2002) se basa en la experiencia obtenida de organizaciones para saber cuál es la mejor manera de implementar un programa de medición de software con garantías de éxito.
- **IEEE 1992 (Metodología para Medidas 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 de un sistema mediante una lista de atributos de calidad del software requeridos por el propio sistema.
- **ISO/IEC 15939:** este estándar internacional (ISO/IEC, 2002) 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.

En general, el objetivo de todo proceso de medición es recoger indicadores cuantitativos sobre entidades software. Una entidad software es todo elemento software sobre el que se puede aplicar un proceso de medición. Puede ser un objeto físico (como un programa), un evento que ocurre en un determinado instante de tiempo (hito) o una acción que se lleva a cabo en un periodo de tiempo determinado (una actividad, como la codificación o las pruebas). Las entidades se clasifican en productos y procesos y están caracterizadas por una serie de atributos (tamaño de un programa, tiempo requerido para la codificación, etc.). Para realizar la medición es necesario identificar tanto las entidades como los atributos a medir, es decir, no se puede medir una entidad o un atributo de forma aislada, como por ejemplo medir un programa o medir el tamaño, sino que se tienen que medir de forma conjunta, especificando que lo que se quiere medir es el tamaño de un programa (Morasca, 2001). Por tanto, para llevar a cabo mediciones, hay que estudiar las entidades software implicadas y los atributos característicos de dichas entidades.

Los tipos característicos de entidades software que pueden ser objeto de medición son:

- **Medición del Proceso**, basado en el estudio y control de la capacidad de los procesos, así como en la gestión de los cambios en el proceso. Las medidas del proceso de software se utilizan para propósitos estratégicos. Los indicadores de procesos permiten a una organización tener una visión profunda de la eficacia de un proceso ya existente; por ejemplo: el paradigma, las tareas de Ingeniería del Software, los productos del trabajo e hitos. Permiten que los administradores evalúen lo que funciona y lo que no.

  Las medidas de proceso se recopilan de todos los proyectos y durante un largo periodo de tiempo. El objetivo es proporcionar indicadores que lleven a mejoras de los procesos software a largo plazo. Las medidas de proceso también se extraen midiendo las características de tareas específicas de la ingeniería de software y obteniendo como resultados medidas sobre los errores detectados antes de la entrega del software, defectos detectados e informados por los usuarios finales, productos de trabajo entregados, el esfuerzo humano y tiempo consumido, ajuste con la planificación , etc.

- **Medición del Proyecto**, basado en la gestión de proyectos. Dichas medidas son tácticas, es decir, las medidas de proyectos y los indicadores derivados de ellos son utilizadas por un administrador de proyectos y por un equipo de software para adaptar el flujo de trabajo del proyecto y las actividades técnicas. **Los indicadores de proyecto** permiten al administrador de software (Pressman, 2001):
  - Evaluar el estado del proyecto en curso.
  - Realizar un seguimiento de los riesgos potenciales.
  - Detectar las áreas de problemas antes de que se conviertan en "críticas"
  - Ajustar el flujo y las tareas de trabajo.
  - Evaluar la habilidad del equipo del proyecto en controlar la calidad de los productos de trabajo de la Ingeniería del Software.

- **Medición del Producto**, centrado en su calidad y aspectos técnicos. Está centrada en evaluar la calidad de los entregables producidos a lo largo del ciclo de vida del software

En el contexto de medición genérica del software de esta tesis y con el objetivo de definir el estado actual de medición del software ha sido necesario hacer una búsqueda en la bibliografía de trabajos que presenten propuestas de medición relacionadas con la medición genérica o con el uso de modelos para llevar a cabo la medición. El resultado de esta búsqueda bibliográfica se muestra en los siguientes apartados. En primer lugar se presentan las propuestas que definen mediciones en dominios específicos y en segundo lugar los trabajos relacionados con la medición genérica del software.

3.2.1. Medición de Dominios Específicos.

En este apartado se presentan propuestas de mediciones en dominios específicos. En primer lugar se presentan mediciones cuyas medidas han sido definidas con el objetivo de medir un metamodelo en un dominio concreto, en segundo lugar se presentan trabajos en los que se definen medidas mediante el uso de lenguajes ya existentes, y por último se muestran herramientas para llevar a cabo mediciones software.

3.2.1.1. Medidas sobre Metamodelos de Dominios Específicos.

Diversos autores han desarrollado trabajos en los que se definen metamodelos para llevar a cabo mediciones en dominios específicos.

En los trabajos presentados en (Misic y Moser, 1997) y en (Mens y Lanza, 2002) se presenta una propuesta basada en metamodelos para especificar medidas genéricas en sistemas orientados a objetos (OO). En estos trabajos se definen metamodelos que incluyen conceptos abstractos del dominio, como clase o atributo. Se define una medida genérica por medio de los conceptos del metamodelo, y se personaliza para un lenguaje específico por medio del mapeo de los conceptos del lenguaje y del metamodelo. Ambos trabajos demuestran la genericidad de las propuestas. En el trabajo de Mens y Lanza se presenta además un mecanismo de extensión y composición de medidas.

En otro trabajo presentado en (Abounader y Lamb, 1997) se publica un informe técnico en el que se enumeran las principales medidas OO existentes en la literatura, y unen los conceptos necesarios para la especificación de un metamodelo para medir modelos OO. En este trabajo no se aborda la especificación e implementación de medidas.

Tichelaar et al. (Tichelaar y Demeyer, 1998) definen un lenguaje genérico que sirve de interfaz entre lenguajes OO (Java, C+, Ada, Smalltalk) y herramientas, de forma que las herramientas pueden utilizar modelos independientemente del lenguaje de programación utilizado. Este trabajo es aplicable a la medición ya que la propuesta permite definir e implementar medidas en un determinado lenguaje OO y ser utilizadas por herramientas de medición. En Lanza y Ducasse (Lanza y Ducasse, 2002) hacen uso de esta propuesta definiendo tres tipos de medidas genéricas utilizando el lenguaje OO *Smalltalk* y por medio de la herramienta *CodeCrawler*. Las medidas genéricas definidas son las siguientes: *NodeCount* que cuenta los nodos que satisfacen un predicado, *EdgeCount* que cuenta los arcos que satisfacen dos predicados que une el grafo (nodo origen y nodo destino) y *PathLenght* que calcula la longitud de una cadena de arcos.

En la misma línea, Reissing et al. (Reissing, 2001) presentan un metamodelo formal para el diseño OO llamado ODEM (Object-oriented DEsign Model). Este metamodelo puede servir de base para la definición formal de medidas del diseño OO. ODEM extiende el metamodelo UML y proporciona un modelo formal de los diseños orientados a objetos expresados en UML.

Tang y Chen (Tang y Chen, 2002) presentan una metodología que recoge las especificaciones UML para obtener información del diseño y calcular las medidas de diseño en las primeras etapas del ciclo de desarrollo software. En este trabajo se utilizan diagramas obtenidos por la herramienta Rational Rose y se calculan medidas OO que se consideran buenos indicadores para identificar los fallos de la funcionalidad OO. Estos indicadores se utilizan para realimentar el modelo UML con los fallos identificados.

Siguiendo en el dominio OO, existen trabajos de mediciones sobre metamodelos. Ma et al. (Ma et al., 2004) comparan diferentes versiones del metamodelo UML para aplicar las medidas Orientadas a Objetos (OO) definidas en (Bansiya y Davis, 2002). En el trabajo de presentado en (Bansiya y Davis, 2002) se describe un modelo jerárquico para la evaluación de diseño de alto nivel de atributos de calidad en diseño de orientación a objetos. En este modelo se evalúa el diseño de comportamiento y estructural de propiedades de clases, objetos y sus relaciones. Relaciona propiedades de diseño como la encapsulación, modularidad, acoplamiento y cohesión con atributos de calidad de alto nivel como son la reutilización, flexibilidad, y complejidad.

En el trabajo de Ma et al. (Ma et al., 2005) se definen patrones vinculados con el ciclo de vida de las meta-clases, y los estudian sobre diferentes versiones de metamodelo UML. Este trabajo está inspirado en los trabajos de (Mattsson y Bosch, 1999; Bansiya, 2000; Gîrba et al., 2005).

Dentro del dominio de UML, existen autores cuyos trabajos se basan en la medición de modelos UML para los distintos artefactos UML (clases, componentes, casos de uso, etc.). Para la medición de los diagramas de clases, (Genero et al., 2005b) han realizado un estudio completo de las medidas aplicadas a modelos de diagramas de clases. Otros autores han llevado a cabo mediciones de modelos dinámicos (Genero et al., 2002; Baroni, 2005), de modelos de componentes (Mahmood y Lai, 2005), modelos de casos de uso (Bernárdez et al., 2004; Cruz-Lemus et al., 2009; Cruz-Lemus et al., 2010) y expresiones OCL (Reynoso et al., 2003; Cabot y Teniente, 2006).

Por otro lado, existen propuestas de mediciones de modelos de procesos. Berenbach et al. (Berenbach y Borotto, 2006) presentan una lista de medidas aplicadas en proyectos reales para el desarrollo de requisitos dirigidos por modelos, además se presentan buenas prácticas a tener en cuenta. La aplicación de medidas aplicadas a procesos MDE se pueden ver en los diversos trabajos desarrollados por el proyecto Modelware (Modelware, 2006a; Modelware, 2006b; Modelware, 2006c).

Otros autores centran sus trabajos en mediciones de transformaciones de modelos. La transformación de modelos es un pilar en MDE (Schmidt, 2006), y por tanto la posibilidad de medir transformaciones de modelos, evaluando su complejidad y el coste de los modelos puede resultar de gran interés. Puesto que una transformación está definida como un modelo es posible aplicar este punto en el contexto MDD. Así lo aplica Saeki et al. (Saeki y Kaiya, 2006) que proponen una técnica relacionada con la definición de medidas en un contexto MDD. Más concretamente se centran en los siguientes puntos:

- Aplicación de una técnica de meta modelado para especificar formalmente medidas para modelos específicos. La definición de las medidas se realizan en OCL sobre el metamodelo.
- Definición de medidas que se ocupan de aspectos semánticos de los modelos (medidas de semántica) utilizando ontologías de dominio.
- Especificación técnica de medidas de transformaciones de modelo basadas en sistema de reescritura de grafos.

Esta propuesta es genérica en el sentido que se pueden definir medidas independientes del lenguaje de transformación que se utilice, ya que lo que se hace es definir medidas OCL sobre el metamodelo.

3.2.1.2. Lenguajes.

Muchos autores proponen utilizar lenguajes ya existentes con el objetivo de definir medidas. Las principales propuestas a utilizar son SQL, lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones; XQuery, conocido por hacer consultas sobre ficheros XMI, y OCL conocido por ser un lenguaje de modelado de restricciones sobre modelos UML.

Con respecto a la utilización se SQL para la definición de medidas, (Harmer y Wilki, 2002) exponen un diseño para calcular medidas de código fuente de programas OO. Para llevar a cabo el estudio, los autores crean una base de datos relacional para almacenar el código fuente del lenguaje OO. Proponen utilizar rutinas C con lenguaje SQL integrado para calcular medidas de complejidad para cada clase almacenada en la base de datos (véase la Figura 3-4).

```
void SQL_select_class_links_table(char *qry, char *cls_id)
{
   strcpy(qry, "select * from links_class");
   strcat(qry, cls_id);
} ;
```

Figura 3-4. Ejemplo de medidas definidas con rutinas C y SQL integrado

En el trabajo presentado en (Lavazza y Agostini, 2005) se sigue la misma propuesta pero para medir modelos UML. Con respecto al trabajo de Hammer y Wilki, en este trabajo son capaces de obtener medidas para las etapas del ciclo de vida de desarrollo software. Con el fin de dar soporte cuenta la medición de modelos UML, definen un esquema E-R que recoge los principales conceptos del metamodelo UML. Una herramienta definida en Java lee los modelos UML en formato XMI y lo reemplaza en la base de datos. Las medidas se escriben en SQL y las consultas se efectúan a través de la herramienta estándar de consulta MySQL Control Center. Como todas las medidas son en SQL, se fuerza ejecutar consultas de la forma "número de elementos que satisfacen una petición". Un ejemplo de consulta expresada en SQL puede ser: *SELECT COUNT (*) FROM CLASS WHERE name=@name.*

XQuery es un lenguaje funcional y declarativo que permite manipular documentos XML. El Wakil et al (ElWakil et al., 2005) utilizan XQuery para definir medidas en modelos UML que se procesan sobre ficheros XMI. Los ficheros XMI ofrecen una forma estándar de representar diseños OO, especialmente diagramas UML. En este trabajo se demuestra que son capaces de definir 65 medidas en XQuery. Además, en este trabajo también se presenta y aplican la herramienta software Design-Metrics Crawler para ejecutar las medidas definidas.

En el trabajo de Eichberg et al. (Eichberg et al., 2006) se presenta **QScope**. Qscope es un framework de medidas extensibles que está desarrollado bajo el marco de trabajo *framework Magellan*. QScope es abierto en el sentido que permite al usuario definir nuevas medidas por medio de un lenguaje de consultas declarativo. Esta propuesta permite expresar medidas (hasta 18) sobre un conjunto de artefactos utilizando XQuery.

Otros autores utilizan **OCL** para definir medidas. Baroni et al. (Baroni et al., 2002) describen como se ha llevado a cabo la formalización de medidas orientadas a objetos utilizando OCL. Para ello utilizan su propio metamodelo (basado en UML) expuesto en un trabajo previo. Esta aportación permite la definición de medidas no-ambiguas (unívocas) Y también es posible establecer comparaciones entre los conjuntos formalizados de medidas. Siguiendo la misma propuesta, en el trabajo presentado en (McQuillan y Power, 2006) se utiliza el metamodelo *Dagstuhl Middle Metamodel* (DMM) como base para definir medidas software que están expresadas con consultas OCL sobre el metamodelo. En la Figura 3-5 se muestra un ejemplo de cómo expresar medidas en OCL.

```
// Número total de clases en un paquete
Package:: TC (): Integer
post: result = allClasses ()-> size ()

 // Conjunto total de clases que pertenecen al paquete actual
 Package :: allClasses (): Set( Class )
 post: result = self . contents ()->
       iterate (elem:ModelElement;
             acc:Set( Class )=oclEmpty (Set( Class ))
       |elem . oclIsTypeOf ( Class ) implies
acc -> union (acc -> including ( elem . oclAsType ( Class ))))
```

Figura 3-5. Ejemplo de medidas definidas mediante OCL

Reynoso et al. (Reynoso et al., 2006) aportan avances en la misma dirección utilizando el metamodelo OCL. Describen un conjunto de medidas que capturan las propiedades

estructurales de las expresiones especificadas con OCL de su metamodelo OCL. Es decir se utiliza OCL con dos fines, el primero como un lenguaje para definir medidas y el segundo con el objetivo de evaluar la complejidad de expresiones OCL a través de medidas. Además, dada la relevancia de los modelos MDE, el trabajo se puede extender para la definición formal de las medidas para cada uno de los modelos UML.

Como contrapartida, diversos autores proponen definir medidas con *lenguajes orientados* al problema. Por *lenguaje orientado* entendemos la definición de conceptos, de una sintaxis (textual o gráfica) y la definición de la ejecución asociada[9]. Ejemplo de algunos lenguajes son los siguientes.

- **GraphLog**: el lenguaje GraphLog (Consens y Mendelzon, 1990) está basado en una representación por grafos, y permite manipular datos y definir medidas en forma de consultas. Las medidas resultantes son de la forma *"numero de resultados de una consulta"*. Esta propuesta muestra, por un lado la posibilidad de utilizar un lenguaje de peticiones para especificar medidas, y por otra parte la especificación gráfica de medidas (mediante grafos).
- **SDMetrics:** SDMetrics (SDMetrics, 2006) es una herramienta comercial para la medición de modelos UML. En SDMetrics una especificación es un conjunto de medidas sobre UML que se traducen a un fichero XML, que la herramienta carga a partir de la petición del usuario. Se puede considerar por tanto el fichero XML como un lenguaje orientado en la especificación de medidas (véase Figura 3-6). Desde esta perspectiva el DTD (Document Type Definition) representa el metamodelo de especificación de medidas.

```
<!--a single metric -->
<metric name="NumPubOps" domain="class" category="Size">
  <description>The number of public operations in a class</description >
  <projection relset ="ownedoperations" condition ="visibility ='public'"/ >
</metric >
<!-- intermediate helper set -->
<set name ="AssocOut" domain ="interface">
  <description>
    The set of associations with navigability away from the interface.
  </description >
 <projection relset ="ownedattributes" condition ="association!=''"
   element =" association " />
</set >
```

Figura 3-6. Ejemplo de medida especificada con la herramienta SDMetrics

- **CQL(Code Query Language):** el lenguaje CQL (Smacchia, 2006) forma parte de la herramienta comercial NDepend[10]. CQL permite escribir consultas sobre una aplicación .NET independientemente de cuál sea el lenguaje utilizado (C++, C#, VB). CQL es similar a SQL ya que tiene a misma estructura de consulta de SQL.

[9] Por lenguaje orientado al problema por DSL

[10] En http://www.ndepend.com

- **SAIL:** Marinescu et al (Marinescu et al., 2005) proponen este DSL destinado a dar soporte a la definición de medidas de diseño OO. La justificación de este lenguaje reside en la obligación de automatizar la medición en programas de gran tamaño. En la propuesta se implementan 40 medidas utilizando SAIL. La propuesta SAIL se puede concebir como un DSL orientado a las medidas de diseño orientados a objetos.
- **RML (Relation Manipulation Language):** Beyer et al (Beyer et al., 2005) especifican un nuevo lenguaje RML donde uno de los objetivos es la de especificar medidas. Este lenguaje es similar GraphLog. RML está basado en expresiones relacionales, es decir manipulando expresiones como conjuntos de tuplas. RML también contiene expresiones numéricas.
- **GMEF (Generic Metric Extraction Framework):** el trabajo presentado en (Alikacem y Sahraoui, 2006) consiste en definir un metamodelo de la lógica orientada a objetos y un lenguaje de especificación de medidas. Dicho lenguaje permite navegar en un modelo que está definido a partir de su metamodelo y contiene operadores clásicos para la manipulación de cantidades y conjuntos.
- **SLAMMER:** Guerra et al. (Guerra et al., 2008) presentan un Framework para la creación de lenguajes visuales específicos de dominio (Domain Specific Visual Languages - DSVL), en el que se ha desarrollado el lenguaje SLAMMER como caso de estudio para representar de modelos de medición. Este lenguaje 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.
- **MDSL:** se trata de de un trabajo interesante en el área de medición es el Measurement-Domain Specific Language (MDSL) (Arpaia et al., 2009). Este lenguaje es un lenguaje formar especialmente diseñado para un dominio específico de medición y prueba.

3.2.1.3. Herramientas.

Existen muchas publicaciones que mencionan las herramientas que dan soporte a la medición como importantes factores de éxito en los esfuerzos de la medición del software (Komi-Sirviö et al., 2001), proporcionando entornos de trabajo y aproximaciones generales (Kempkens et al., 2000), o dando arquitecturas de soluciones más específicas (Jokikyyny y Lassenius, 1999).

Como consecuencia, en la bibliografía existe una gran variedad de herramientas que dan soporte a la creación, control y análisis de medidas software. Una lista extensa puede consultarse en (Dumke y Grigoleit, 1997).

En el trabajo presentado en (Brown y Dennis, 2004) se incluye una lista de herramientas que dan soporte a la creación, control y análisis de las mediciones del software.

Por otro lado, Auer examina en (Auer et al., 2003) distintas herramientas de medición del software en entornos heterogéneos, como son: MetricFlame, MetricCenter, Estimate Profesional, CostXPert y ProjectConsole.

También se pueden encontrar en la bibliografía algunas propuestas en las que se aborda la medición de software más integrada y menos específica que en las herramientas anteriores. En (Palza et al., 2003) se propone MMR, que es una herramienta basada en el modelo CMMI para la evaluación de procesos software.

En la literatura pueden encontrarse más herramientas (Harrison, 2004; Scotto et al., 2004; Lavazza y Agostini, 2005), sin embargo están restringidas a un dominio concreto o a evaluar características de calidad específicas.

En (Harrison, 2004) se presenta una herramienta que trata un proceso como una serie de transformaciones de artefactos, cada uno con entradas que producen salidas. La utilización de esta propuesta soporta una gran variedad de medidas de software.

Lavazza y Agostini presentan una herramienta de medición en UML que no sólo da soporte a las mediciones de modelos UML utilizando la medidas tradicionales, sino que también proporciona soporte para que el usuario pueda definir nuevas medidas ya sean para el análisis, procesos, etc.

Por otro lado existe WebMetrics (Scotto et al., 2004) que es una herramienta automatizada que contiene una colección de medidas software. Su uso, como capa intermedia, es un conjunto de relaciones que describen la estructura del código fuente. Estas relaciones se almacenan en una base de datos con el objetivo de calcular la medidas mediante ejecución de consultas SQL.

Sin embargo, muchas de estas herramientas están restringidas a dominios o modelos de evaluación de calidad específicos, lo que reduce su generalidad y alcance.

3.2.2. Medición Genérica.

En este apartado se presentan trabajos relacionados con la medición genérica de modelos. La genericidad se refiere a la capacidad de poder aplicar medidas independientemente del tipo de artefacto. En términos de MDE, consiste en aplicar las medidas en multitud de metamodelos.

En 1992 Tsalidis et al (Tsalidis et al., 1992) propusieron el entorno **Athena,** que permite generar automáticamente medidas para un lenguaje de programación dado. Para llevarlo a cabo proponen un metalenguaje. En la Figura 3-7 se resumen el proceso de la propuesta.

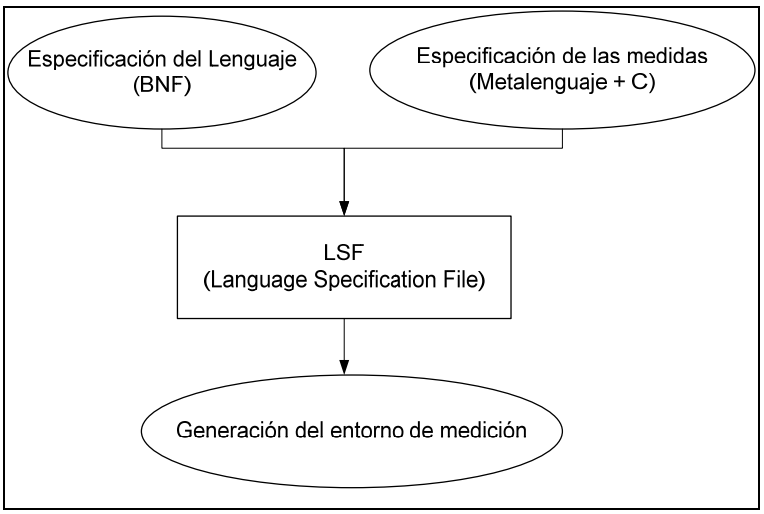


Figura 3-7. Propuesta de Athena

Como se observa en la figura anterior para obtener el entorno de medición, es necesario especificar las medidas en un metalenguaje específico[11] para las medidas. Este lenguaje es una mezcla de sentencias declarativas y código imperativo escrito en C. Una vez que se ha especificado el lenguaje y las medidas se obtiene un fichero que especifica el lenguaje (Language Specification File – LSF). La generación del entorno de medición se hace en dos fases.

[11] En 1992 no se utilizaba el concepto Domain Specific Language pero si Design Specific Language.

- Primero, a partir del fichero LSF, un compilador genera un fichero utilizable en la herramienta YACC.
- A continuación la herramienta YACC genera el código C.

Monperrus et al. (Monperrus et al., 2010) crean una especificación de medidas con el objetivo de medir cualquier dominio gracias a la generación de código (idea previa definida en (Monperrus et al., 2008) que es una réplica de la idea general de medición genérica propuesta en el entorno SMF (Mora et al., 2008d; Mora et al., 2009a), presentado en esta tesis pero usando un entorno tecnológico diferente.

Por su parte, Vépa et al. (Vépa et al., 2006) describen un trabajo en el que se calculan medidas mediante transformaciones de modelos, utilizando el lenguaje de transformación ATL. Presentan un metamodelo que permite el almacenamiento de datos de medición, y un conjunto de transformaciones para llevar a cabo la medición de modelos definidos con este metamodelo. Este artículo se centra en los aspectos tecnológicos necesarios para implementar la medición del software con la tecnología ATL, ofreciendo al usuario una gran variedad de representaciones gráficas de los resultados de medición obtenidos. A pesar de utilizar transformación de modelos para obtener los resultados de las medidas, la propuesta de Vépa es menos genérica que la presentada en la tesis porque las medidas están definidas de partida, al contrario que con SMF donde las medidas las define el usuario en cada momento.

Con respecto a la existencia de un referente común que permite definir modelos de medición existe un metamodelo de medición, Software Metrics Meta-Model (OMG, 2007) desarrollado por la OMG. Este metamodelo promueve un formato de intercambio común que permite la interoperabilidad entre las herramientas existentes, servicios y sus respectivos modelos. A pesar de la existencia de este metamodelo, en el desarrollo de la tesis se ha optado por utilizar el metamodelo de medición del software de FMESP (García et al., 2006b) ya que está basado en una ontología previa, lo que aporta 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). Como trabajo futuro se alineará la propuesta para ser compatible con dicha propuesta de OMG.

En el siguiente apartado se proporciona un resumen de FMESP, que proporciona los antecedentes que dieron lugar a la realización de la presente tesis doctoral. La propuesta presentada en esta tesis parte de la base conceptual sólida establecida en la ontología de la medición del software y los mecanismos de medición genérica básicos definidos en FMESP (García et al., 2006b), y extiende dicha propuesta: adaptándola a MDE para aprovechar los beneficios de dicho paradigma, proponiendo nuevos métodos de medición genéricos para la definición de medidas software sobre cualquier metamodelo y desarrollando y validando un lenguaje gráfico para la definición de modelos de medición software.

3.3. Fundamentos Previos.

En este apartado se presentan los trabajos anteriores que han sido realizados por el grupo de investigación Alarcos, y cuyos resultados se aprovechan en esta tesis. Fundamentalmente son tres: la ontología de medición del software, el metamodelo de medición de software y, por último, el marco de trabajo FMESP.

3.3.1. Ontología de Medición del Software.

Ante la diversidad terminológica existente en el campo de la medición software, como paso previo a la creación de la primera versión del metamodelo de Medición (García et al., 2007) se desarrollo una ontología (García et al., 2006a) Software Measurement Ontology (SMO). Esta ontología permitió establecer y aclarar los elementos involucrados mediante la identificación de todos los conceptos, las definiciones precisas de todos los términos, y la aclaración de las relaciones entre ellos. Basándose en los conceptos y relaciones de esta ontología, se construyó el metamodelo de medición.

Para facilitar la comprensión, en el proceso de representación de la ontología de la medición del software se optó por dividirla en cuatro sub-ontologías, basándose en el modelo propuesto en (García, 2004). Así, la SMO se organiza de la siguiente forma:

a) **Characterization and Objectives *(Caracterización y Objetivos)***, que incluye los conceptos necesarios para establecer el ámbito y los objetivos del proceso de medición software. El principal objetivo del proceso de medición software es satisfacer ciertas *necesidades de información* identificando las *entidades* (que pertenecen a *categoría de entidad*) y los *atributos* de estas *entidades* (que son el enfoque del proceso de medición). *Atributos* y *necesidades de información* están relacionados por medio de *conceptos medibles* (que pertenecen a *modelo de calidad*).

b) **Software Measures *(Medidas Software)***, que trata de establecer y aclarar los elementos clave en la definición de una *medida software*. Una *medida* relaciona un *método de medición* definido y una *escala* de medición (que pertenece a *tipo de escala*). La mayoría de las *medidas* pueden ser expresadas en una *unidad de medición*, y pueden ser definidas para más de un *atributo* (Medidas nominales son ejemplos de medidas que no pueden ser expresadas en unidades de medición). Las *medidas* pueden ser de 3 tipos: *medidas base, medidas derivadas e indicadores.*

c) **Measurement Approaches *(Formas de Medir)***. Esta sub-ontología introduce el concepto de *forma de medir* para generalizar los diferentes "métodos" usados por los tres tipos de *medidas* para obtener sus respectivos *resultados de medición*. Una *medida base* aplica un *método de medición*. Una *medida derivada* usa una *función de cálculo* (que se apoya en otra medida base y/o derivada). Por fin, un *indicador* utiliza un *modelo de análisis* (basado en un *criterio de decisión*) para obtener un *resultado de medición* que satisfaga a una *necesidad de información*.

d) **Measurement Action *(Acción de Medir)***. Establece la terminología relacionada con el acto de medir el software. Una *medición* (que es una acción) es un conjunto de operaciones cuyo objetivo es determinar el valor de un *resultado de medición* para un dado *atributo* de una *entidad*, utilizando una *forma de medir*. Los *resultados de la medición* se obtienen a través de la acción de llevar a cabo una *medición*.

Cada una de las sub-ontologías que componen la ontología de la medición del software se describe con mayor detalle en (García et al., 2006a). En la Figura 3-8 se muestra el diagrama UML completo de la ontología de la medición del software, mostrando todos sus términos, atributos e interrelaciones.

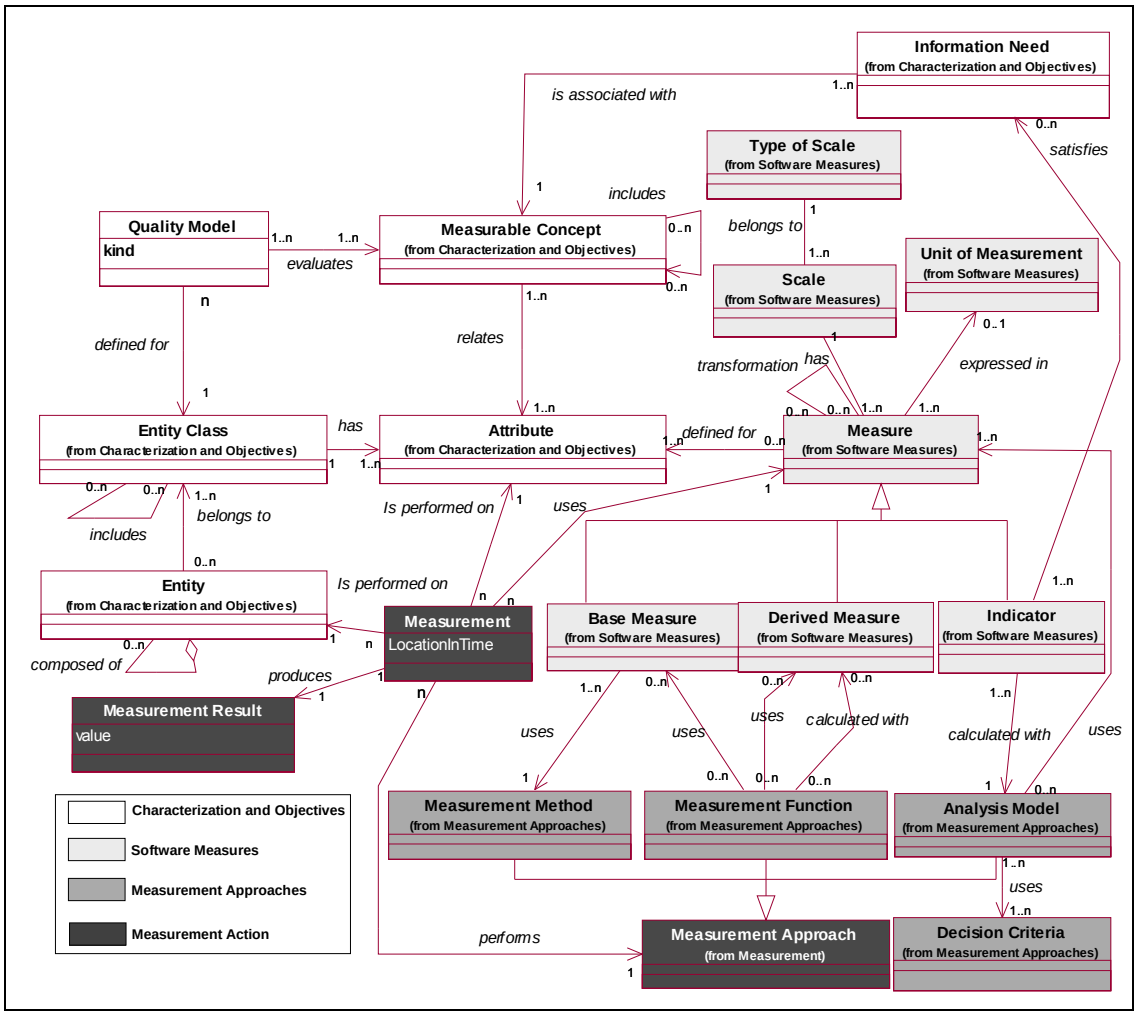


Figura 3-8. Diagrama de la Ontología de la Medición del Software.

A través de esta ontología, además de establecer y clarificar los conceptos y relaciones relacionados con la medición del software, se proporciona una terminología común que facilita el entendimiento entre miembros del equipo responsables de llevar a cabo mediciones en una empresa y la posibilidad de registrar los resultados de dicho proceso de una forma consistente e integrada (García, 2004).

3.3.2. Metamodelo de Medición del Software.

Basándose en los conceptos y relaciones de la ontología de medición del software descrita en el apartado anterior, García et al. (García et al., 2007) definieron el metamodelo de medición: Software Measurement Metamodel (SMM).

En primer lugar es interesante saber la diferencia entre una ontología y un metamodelo. Las ontologías son conceptualizaciones de un dominio identificando qué conceptos existen y cuáles son sus relaciones. Los metamodelos son modelos para crear otros modelos y se usan en el contexto de la arquitectura de MOF y pertenecen a nivel M2. Las ontologías son independientes de arquitecturas conceptuales y tecnologías mientras que los metamodelos no.

La Figura 3-9 muestra la estructura de paquetes en la que se basa el metamodelo de la medición.

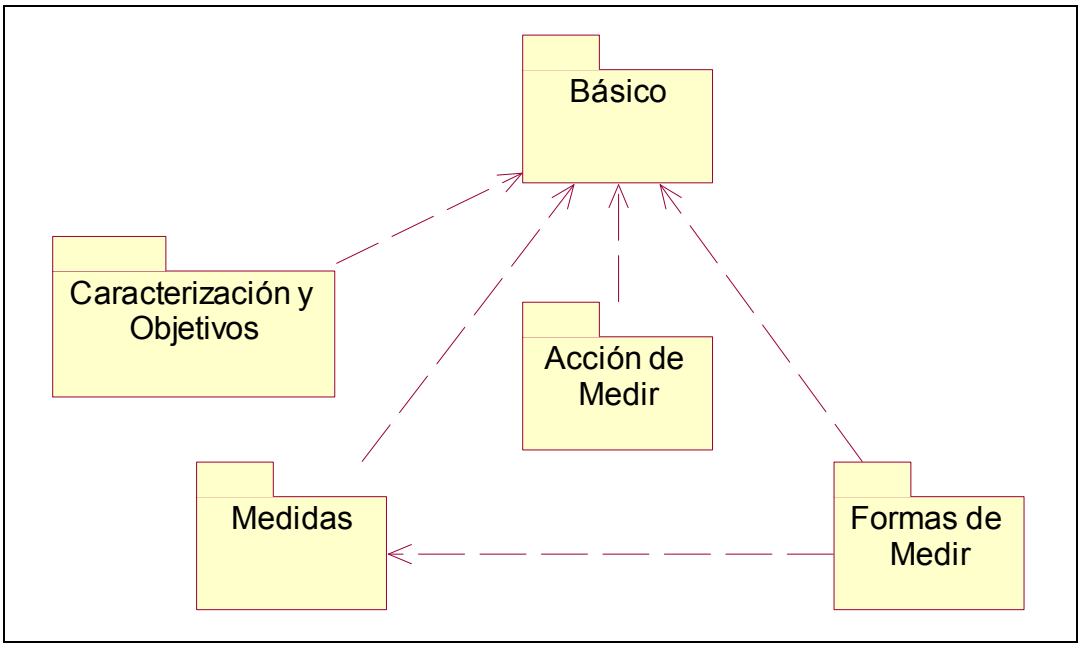


Figura 3-9. Estructura de Paquetes del Metamodelo de la Medición.

Tal y como se puede observar en Figura 3-9, el metamodelo de la medición está compuesto de un paquete básico que representa las características generales de los constructores básicos de modelos de medición, y de otros cuatro paquetes (Caracterización y Objetivos, Acción de Medir, Formas de Medir y Medidas), de acuerdo con las cuatro sub-ontologías de la SMO. Para la construcción de este metamodelo, se tuvo 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.

Como la mayoría de los conceptos de SMO pertenecen al nivel de metamodelado M2, no fue necesario realizar una transformación significativa de la misma para la construcción del metamodelo, excepto a la hora de incorporar un paquete orientado a la implementación de los conceptos de la ontología, mediante su definición como constructores de medición.

La Figura 3-10 muestra el diagrama de clases UML que representa la estructura del paquete básico. Este paquete se ha definido con el objetivo de identificar y establecer las características generales de los constructores necesarios para definir modelos de medición.

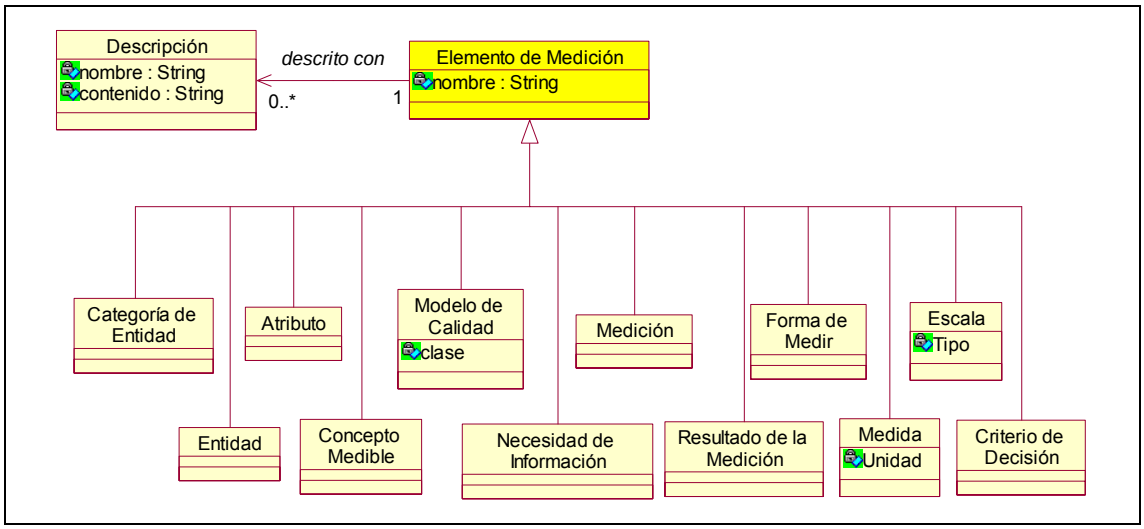


Figura 3-10. Paquete Básico.

Como se puede observar en la Figura 3-10, el elemento general a partir del cual se construyen modelos de medición es el constructor "**Elemento de Medición**". Un elemento de

medición tiene un nombre y puede ser descrito mediante elementos de tipo "**Descripción**", 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.

Más información del metamodelo de medición del software puede encontrarse en (García et al., 2007).

3.3.3. FMESP.

El marco de trabajo **FMESP** (*Framework for the Modeling and Evaluation of Software Processes*) (García et al., 2006b) se desarrolló con el objetivo de proporcionar el soporte conceptual y tecnológico necesario para la representación y medición integrada de los procesos software con el fin de facilitar su mejora. Los elementos que constituyen FMESP son:

1. Un **Marco Conceptual** para representar y gestionar el conocimiento relacionado con los procesos software desde el punto de vista de su modelado y medición. Con el fin de representar y medir los procesos de una forma integrada el marco conceptual está constituido por los siguientes elementos:

- **Arquitectura Conceptual de Gestión de Metadatos en 4 niveles de abstracción**. Para facilitar la gestión integrada del modelado y de la medición de los procesos software se utilizó una arquitectura conceptual de cuatro niveles de abstracción propuesta por el estándar MOF. Esta arquitectura conceptual incluye los lenguajes de modelado (metamodelos) necesarios para la definición de procesos y de medidas, y los metamodelos necesarios para representar cualquier entidad software relacionada con la medición del proceso software. También incluye los modelos concretos de definición y medición de los procesos.
- Una colección de **Ontologías (**ontología del modelado descriptivo de los procesos software y una ontología de la medición del software**)**. Desarrolladas con el objetivo de facilitar el modelado y la medición de los procesos software para que todos los modelos y metamodelos utilizados estén basados en una misma conceptualización.
- Una colección de **Metamodelos**. Con el fin de facilitar la automatización del modelado y la medición de los procesos software y, representar dicho conocimiento de forma procesable por un computador. Los metamodelos que forman parte del marco conceptual son los siguientes:
    a. Un **Lenguaje de Modelado de Procesos (LMP)**. El lenguaje de modelado de procesos es el metamodelo SPEM (*Software Process Engineering Metamodel Specification*) desarrollado por el consorcio OMG. Considerado el más adecuado para modelar de forma descriptiva los procesos software gracias a su amplia aceptación en la industria y su gran generalidad.
    b. Un **Metamodelo para la Definición de Modelos de Medición**. Descrito en el apartado 3.3.2.
    c. **Metamodelos** para la definición de **Entidades Software**. Necesarios para poder realizar la medición de las entidades relacionadas con el proceso software. A estos metamodelos se les denomina Metamodelos del Dominio.

En la Figura 3-11 se muestra el esquema de los elementos que constituyen el marco conceptual de FMESP:

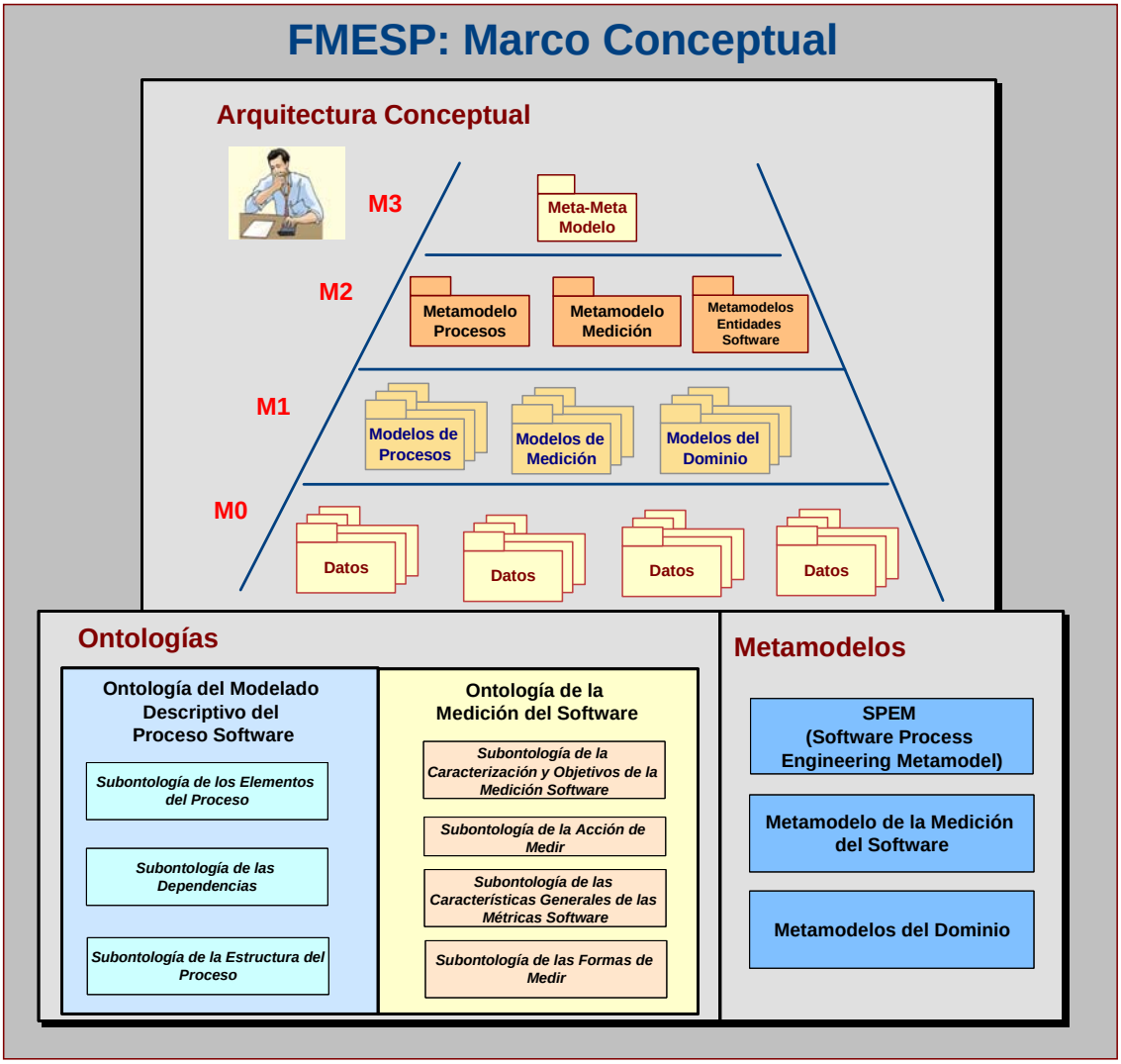


Figura 3-11. Marco Conceptual de FMESP

- Un **Entorno de Ingeniería del Software (EIS)**. El EIS da soporte al marco conceptual. Está compuesto por dos herramientas: **METAMOD** para la gestión de la arquitectura conceptual; y **GenMETRIC** para la gestión de la medición del software. El entorno propuesto es flexible en el sentido de que puede incorporar una gran variedad de herramientas, dado que dispone de un repositorio de metadatos que puede ser compartido entre todas ellas. En la Figura 3-12 se muestra la estructura del EIS propuesto.

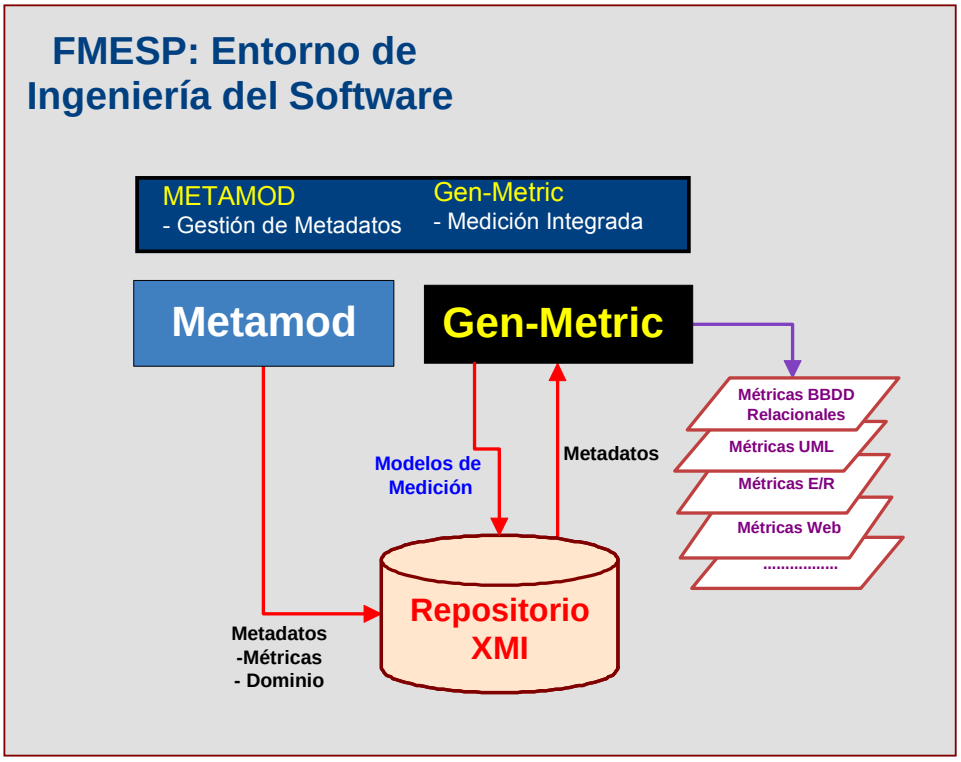


Figura 3-12. Entorno de Ingeniería del Software de FMESP

En (García et al., 2006b) se describen con mayor detalle los elementos que constituyen el marco conceptual de FMESP.

3.4. Conclusiones.

En este capítulo se han estudiado las principales aportaciones existentes en la literatura relacionadas con MDE y la medición de modelos software. Tal y como se ha presentado, existen propuestas relacionadas con la medición de metamodelos, procesos, modelos, etc. Pero ninguno de estos trabajos aprovecha el potencial de poder definir un entorno completo que permita realizar mediciones en cualquier momento del ciclo de vida del producto y para cualquier dominio utilizando una base conceptual sólida, una representación de modelos de medición usable y adecuada y que aproveche los beneficios del paradigma MDE.

Teniendo en cuenta las necesidades anteriores, surge la presente tesis doctoral en la que se desarrolla un entorno conceptual y tecnológico que permite llevar a cabo mediciones genéricas basadas en MDA.

“Si tu intención es describir la verdad, hazlo con sencillez y la elegancia déjasela al sastre” (Albert Einstein)

4. SMF: Marco de Trabajo.

En este capítulo se presenta SMF, marco de trabajo de Medición del Software genérico para la definición de modelos de medición bien formados respecto a un metamodelo común, y para la medición de cualquier entidad software en base a un metamodelo de dominio. En el primer apartado se justifica la necesidad de disponer de un marco conceptual integrado para modelar y ejecutar mediciones software, presentando las características generales de la propuesta. En los siguientes apartados se describen con más detalle los componentes del marco conceptual.

Los contenidos de este capítulo se complementan con el Capítulo 5, en el que se presenta un lenguaje específico de dominio para representar modelos de medición, con el Capítulo 6, donde se describen las características del entorno tecnológico desarrollado como soporte al marco conceptual, y con el Capítulo 7, que presenta un caso práctico de aplicación del marco de trabajo propuesto.

4.1. Características del Marco de Trabajo.

La **medición del software** se ha convertido en uno de los objetivos estratégicos fundamentales en las organizaciones a la hora de mejorar la calidad de los productos. Es una actividad de ingeniería que permite obtener información cuantitativa con respecto a los procesos de ingeniería o sistemas que se están desarrollando, y es un elemento clave en el ciclo de vida de desarrollo del software ya que da soporte a la planificación, monitorización, control y evaluación de los procesos software. Por ello, la medición ha llegado a ser un aspecto fundamental en la Ingeniería del Software (Fenton y Pfleeger, 1997).

Dada la importancia de la medición en las organizaciones, es interesante y útil establecer un marco de trabajo adecuado que permita que cada organización defina y lleve a cabo de forma efectiva la medición de sus artefactos software (procesos, proyectos, productos, recursos, etc.).

Para poder gestionar de forma efectiva la medición del software hay que considerar, por tanto, los siguientes aspectos fundamentales:

- **Definición de modelos de medición homogéneos.** Debido a la gran variedad de tipos de entidades software y atributos que son candidatos para la medición, existe la necesidad de disponer de modelos de medición homogéneos, que puedan gestionarse por las empresas de una misma forma común, independientemente de cual sea la entidad a medir. Esto implica la necesidad de una referencia consistente y adecuada para la definición de los modelos de medición del software, así como el soporte tecnológico necesario para integrar la medición de los diferentes tipos de entidades. En el apartado 1.1 se describen las características del marco de trabajo, basado en MDE, que da soporte a las mediciones genéricas.
- **Lenguaje visual para la definición de modelos de medición.** Para que la definición de las mediciones se convierta en una tarea fácil e intuitiva para el experto de medición, es interesante la existencia de un lenguaje específico de medición que permite expresar una medición de manera gráfica. Este modelo de medición gráfico se convierte en un punto clave dentro del proceso de medición ya que es el que contiene toda la información de la medición que se quiere llevar a cabo. En el Capítulo 5 se presenta el lenguaje visual y la validación empírica de dicho lenguaje.
- **Método.** Para asegurar que el proceso de medición del software se lleva a cabo de una manera productiva y consistente, además de proporcionar una infraestructura es necesario proveer un método que defina cómo llevar a cabo el proceso de medición genérica (véase apartado 4.4).
- **Transformaciones de modelos.** Para poder obtener los resultados de los modelos de medición del software de manera genérica, beneficiándose de MDE, es necesario que se apliquen transformaciones sobre los modelos con el objetivo de calcular las medidas definidas en un dominio concreto. Por tanto, las transformaciones de modelos se convierten en uno de los puntos clave en el marco de medición genérica. Las transformaciones genéricas definidas para tal fin se presentan en el apartado 4.5.
- **Soporte necesario para calcular de forma automática las medidas definidas.** Para que una medición se pueda llevar a cabo es necesario, además de tener una base conceptual bien definida, una herramienta que de soporte a las mediciones de software. En el Capítulo 6 se presenta la herramienta que da soporte a las mediciones genéricas permitiendo la definición de modelos de medición (de forma gráfica) y el cálculo automático de medidas definidas sobre cualquier dominio que sea candidato a ser medido.

4.2. Elementos del Marco de Trabajo.

A partir de los requisitos establecidos en el apartado anterior, se ha desarrollado el marco de trabajo SMF (Software Measurement Framework) con el objetivo de proporcionar el soporte conceptual y tecnológico necesario para la medición genérica del software. Los elementos que constituyen SMF son: marco conceptual, método de trabajo, formas de medir genéricas, y entorno tecnológico. Se presentan a continuación:

a) Marco Conceptual.

Es la base conceptual para la representación y gestión del conocimiento relacionado con la medición genérica del software y constituye la herramienta intelectual necesaria para medir los modelos que representan las entidades software. Con este fin, está constituido por los siguientes elementos:

- **Arquitectura Conceptual en niveles de abstracción**. Con dicha arquitectura se pretende facilitar la representación de los elementos relacionados con la medición del software, en diferentes niveles de abstracción. La arquitectura conceptual debe facilitar la medición genérica incluyendo el lenguaje de modelado de medición necesario para definir medidas sobre un dominio en concreto, así como los metamodelos necesarios para representar cualquier entidad software relacionada con la medición del software (productos, proyectos, recursos, etc.) y los metamodelos de transformación de modelos. En el apartado 4.3 se describen con mayor detalle las características de la arquitectura conceptual.
- **Lenguaje de definición de modelos de medición.** Con el fin de poder medir cualquier entidad software y al mismo tiempo representar la medición de cualquier entidad de una forma homogénea, es necesario que exista un referente consistente y adecuado para la definición de los modelos de medición del software. Tal y como se ha presentado en el Capítulo 3, existe un Metamodelo de Medición del Software (SMM) (García et al., 2007) a partir del cual se pueden definir modelos de medición que garantizan una base conceptual sólida ya que SMM fue definido a partir de la ontología SMO (García et al., 2006; García et al., 2009), que establece y clarifica los elementos (conceptos y relaciones) involucrados en el dominio de medición del software. Además, la definición de modelos a partir de SMM permite especificar medidas para cualquier dominio (UML, ER, Java, etc.). Sin embargo, para facilitar la usabilidad de dicho metamodelo se debe definir un lenguaje de modelado adecuado. Por ello, para poder definir modelos de medición del software de manera gráfica y facilitar esta tarea al usuario, SMF integra un lenguaje específico de dominio (DSL), llamado SMML (Software Measurement Modeling Language), que está definido a partir de SMM y se presenta en el Capítulo 5.
- **Metamodelos** para la definición de **Entidades Software**. Para poder realizar la medición de las entidades relacionadas con artefactos software es necesario conocer las características de dichas entidades. Por ello, se debe disponer de metamodelos que permitan representar las entidades software mediante modelos. A estos metamodelos les denominamos **Metamodelos de Dominio**.
- **Metamodelos (Lenguajes) de Transformación de modelos.** Para realizar transformaciones entre modelos, con el fin de obtener de forma automática los resultados de la medición, es necesario disponer de lenguajes de transformación adecuados. En SMF se han empleado los siguientes:

- **MOFScript.** El lenguaje de MOFScript [12] permite llevar a cabo transformaciones de modelo a texto. El subproyecto MOFScript da soporte a este lenguaje en términos de edición, parseo y ejecución.
- **QVT Relation.** Es un lenguaje de transformación de modelos definidos acorde a un metamodelo. El resultado de una transformación es otro modelo generado a partir de uno o varios modelos.

En la Figura 4-1 se muestra el esquema de los elementos que constituyen el marco conceptual de SMF:

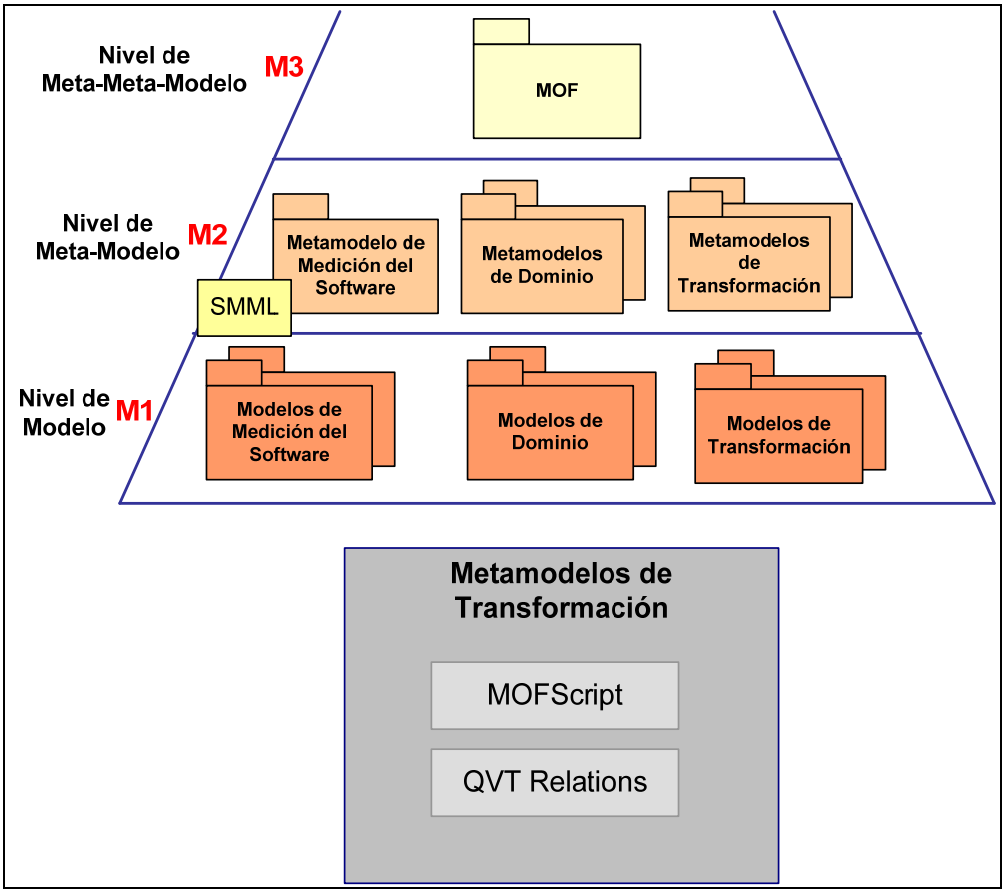


Figura 4-1. Marco Conceptual de SMF

b) Método de Trabajo.

Precisa los pasos necesarios para llevar a cabo una medición genérica. Se detalla en el apartado 4.4.

c) Formas de Medir Genéricas.

Para que las mediciones sean aplicables a cualquier dominio software es necesario definir formas de medir genéricas, que se explican en el apartado 4.6.

[12] Disponible en http://www.eclipse.org/gmt/mofscript/

d) Entorno Tecnológico.

Está constituido por SMTool (Software Measurement Tool), una herramienta que da utilidad real al marco conceptual propuesto y, a la vez, sirve de aval y prueba de su factibilidad en la práctica. En la Figura 4-2 se muestra la estructura del entorno tecnológico desarrollado, que se describe con más detalle en el Capítulo 6.

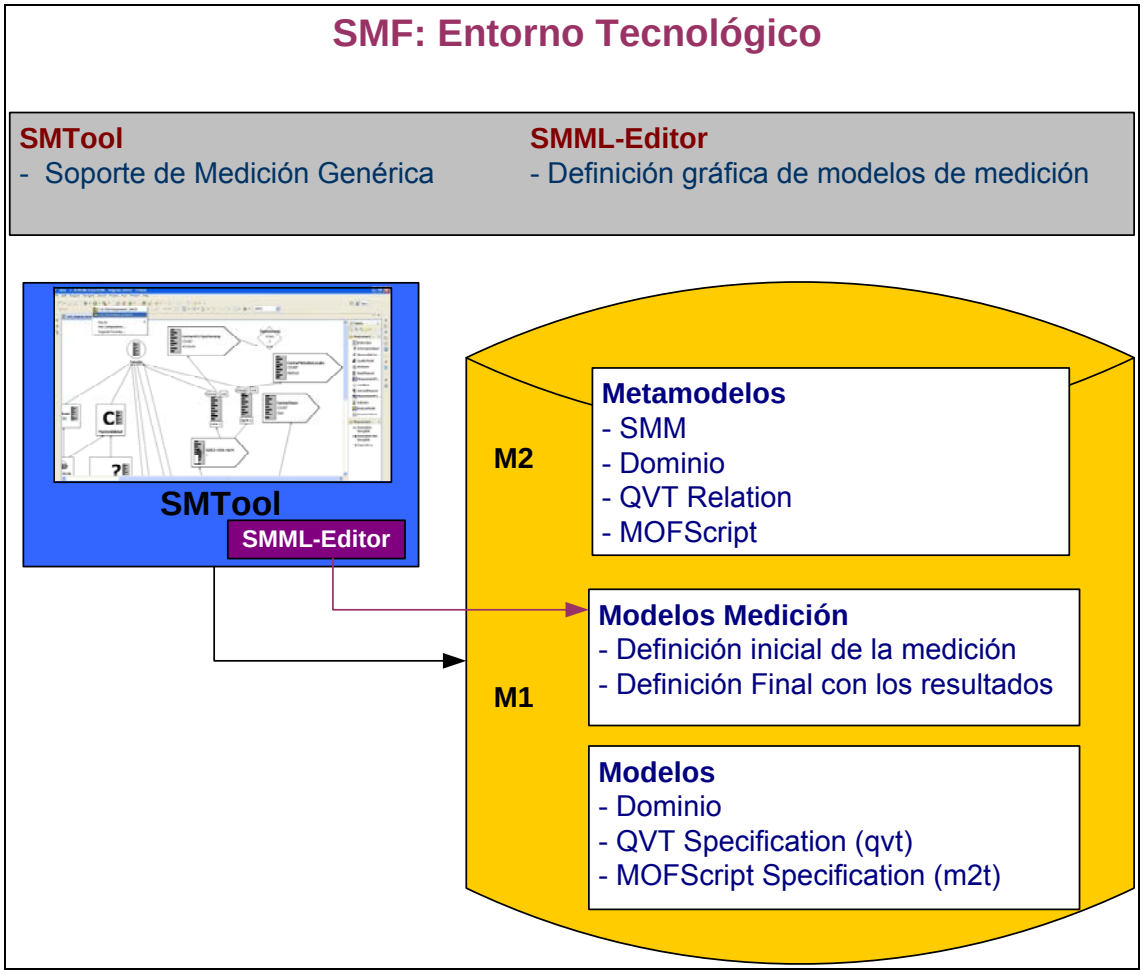


Figura 4-2. Entorno Tecnológico de SMF

En los siguientes apartados se describen con mayor detalle los elementos que constituyen el marco conceptual de SMF.

4.3. Arquitectura Conceptual.

Buscando disponer de un entorno genérico y homogéneo para la medición del software, en trabajos previos (García et al., 2006a; García et al., 2006b; García et al., 2007) se propuso una arquitectura conceptual para la integración de la medición del software en el contexto del entorno FMESP, orientado a la mejora de los procesos de Ingeniería del Software. Aprovechando los beneficios que puede aportar el paradigma MDE en la medición del software, desde el punto de vista de la importancia del uso de modelos y transformaciones en el desarrollo

del software (expuestos en el Capítulo 3), en esta tesis doctoral se ha realizado la adaptación de la arquitectura conceptual de FMESP para que la medición del software se pueda considerar como una serie de transformaciones de modelos. Así, desde este punto de vista, el marco de trabajo SMF puede considerarse una adaptación de FMESP al paradigma MDE.

Con todo ello, la arquitectura propuesta en esta tesis se basa en la arquitectura conceptual en niveles de abstracción MOF propuesta en FMESP (García et al., 2006b) e incluye los elementos necesarios para adaptarla a MDE.

Para adaptar la arquitectura conceptual de FMESP a MDA se han tenido que añadir cinco nuevos tipos de elementos: Metamodelo QVT Relation, Modelo QVT Relation, Metamodelo MOFScript, modelo MOFScript, Metamodelo de Medición del Software, que ha sido adaptado de FMESP para dar soporte al lenguaje de la medición del software (SMML). En la Figura 4-3 se muestra de forma gráfica la arquitectura conceptual de SMF en la que se muestran los modelos y metamodelos relacionados con el modelado de la medición y con el proceso de medición y modelado del dominio.

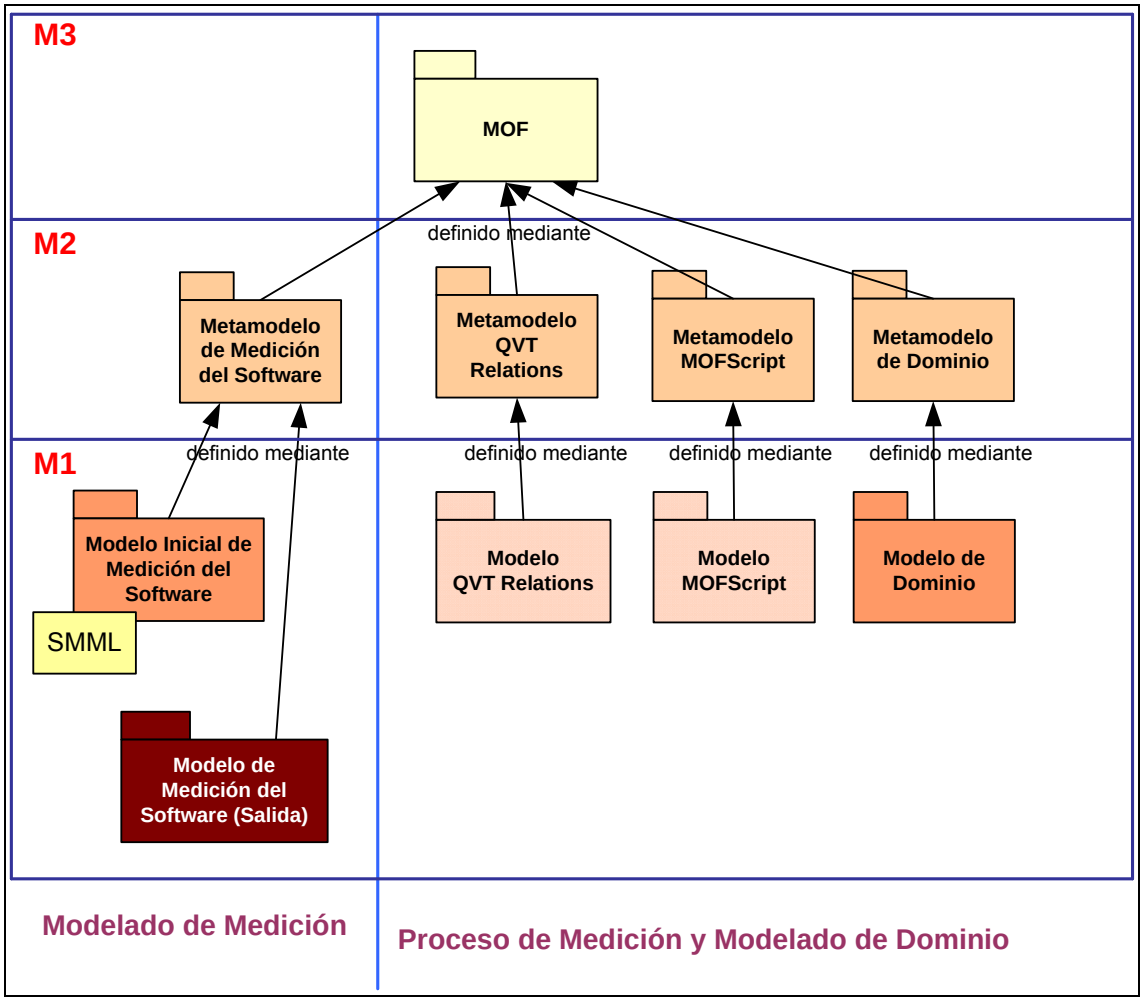


Figura 4-3. Marco de trabajo conceptual para gestionar la medición del Software

Como se observa en la Figura 4-3, los elementos se organizan de la siguiente forma en los niveles de abstracción:

- **Nivel de Meta-Metamodelo (M3).** Refiere al lenguaje abstracto MOF para definir metamodelos. Mediante el uso de este nivel de abstracción es posible integrar y representar diferentes dominios, es decir, trabajar a la vez con varios metamodelos. Por

este motivo y, teniendo en mente nuestro objetivo de medición genérica, se utiliza el lenguaje ECORE para representar los metamodelos necesarios de una forma consistente e integrada. Aunque estrictamente no es MOF, está basado en EMOF, que es parte de la especificación MOF 2.0 para la definición de metamodelos

- **Nivel de Metamodelo (M2).** Incluye los metamodelos necesarios para la definición de los modelos empleados en SMF. Los elementos de los distintos metamodelos se representan con el lenguaje MOF, de forma que todos los conceptos del nivel M2 son instancias de Clase-MOF o Asociación-MOF. Los tipos de metamodelos usados en SMF sones:
  - **De Medición del Software (SMM)**, a partir del cual se definen modelos específicos de medición software.
  - **De dominio.** Por ejemplo, para llevar a cabo la medición de diagramas de clases UML es necesario disponer de un metamodelo, en este caso el metamodelo de UML, que permita representarlos como modelos.
  - **QVT Relations.** Es necesario para definir las especificaciones de las transformaciones entre modelos.
  - **MOFScript.** Es necesario para definir las especificaciones de las transformaciones de modelo a texto.
- **Nivel de Modelo (M1).** En este nivel se incluyen modelos específicos que son instancias MOF-compliant de los metamodelos de M2. Estos modelos pueden ser:
  - **De medición**, definidos mediante SMM. Por ejemplo, modelos específicos para la medición de esquemas de bases de datos relacionales, diagramas de clases UML, diagramas de transición de estados, o código Java.
  - **De dominio**, que representan los dominios que pueden ser objeto de medición, como un diagrama de clases de una aplicación sanitaria, o una base de datos relacional con los pacientes de un hospital.
  - **QVT Relations**, que contiene la especificación de la transformación de modelos (conforme al metamodelo QVT Relations) a partir de la cual se obtienen los resultados de la medición en un dominio concreto. En el apartado 4.5 se encuentra información detallada acerca del proceso de transformación.
  - **MOFScript,** que contiene la especificación de la transformación de modelo a texto (conforme al metamodelo MOFScript) a partir de la cual se obtiene el modelo QVT Relations. Este modelo es único e invariable, es decir, se utiliza para cualquier medición especificada (ver apartado 4.5 para más detalle).
- **Nivel de Datos (M0).** SMF no tiene elementos que se encuentren en este nivel ya que la medición automática se realiza a nivel de modelos y metamodelos, es decir, a partir del nivel M1 hacia arriba.

Con esta arquitectura se permite llevar a cabo la medición genérica en un contexto MDE ya que, como puede observarse, los modelos y las transformaciones son elementos claves del proceso.

4.4. Método de Trabajo para la Medición en SMF.

Para que SMF sea consistente y se utilice de manera correcta ha sido necesario elaborar un método asociado que indique los pasos a seguir para llevar a cabo la medición genérica. En la Figura 4-4 se muestra el método a seguir para cada medición específica (acción de medir).

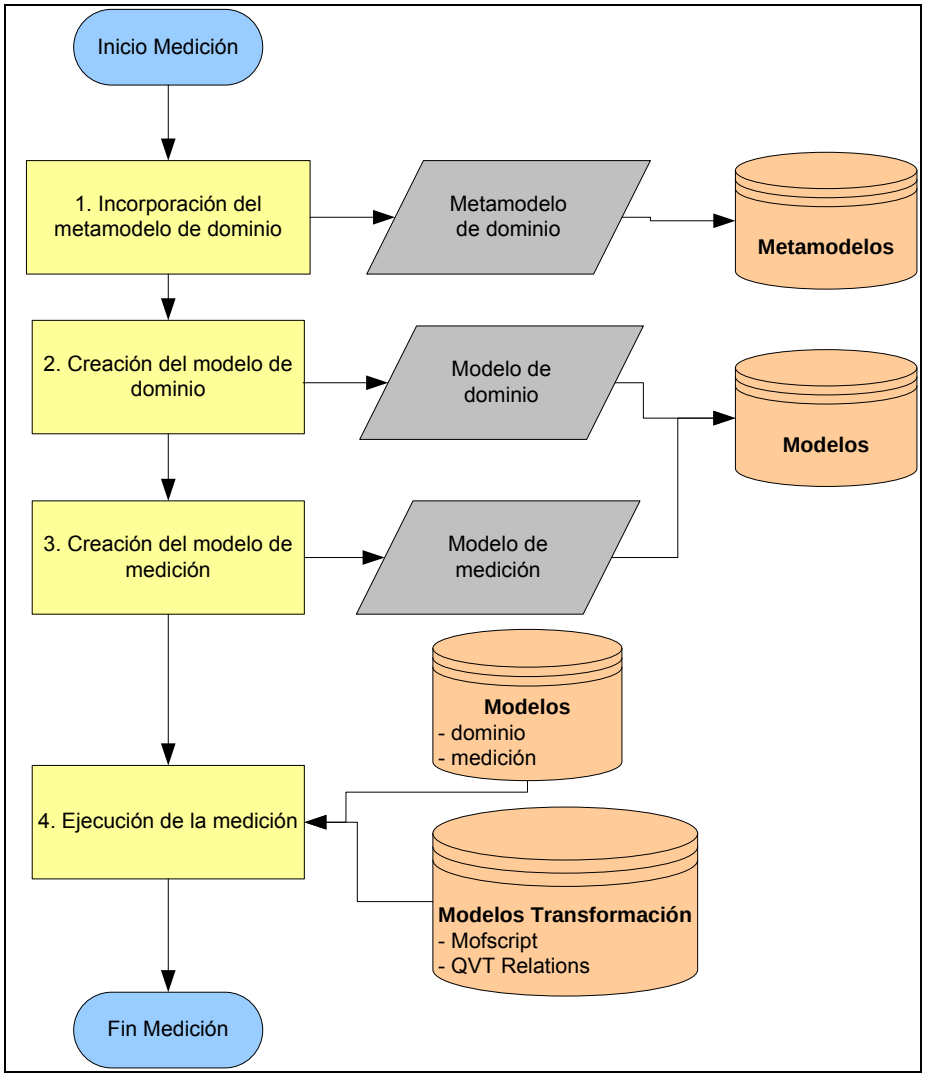


Figura 4-4. Método de SMF

Tal como se indica en la Figura 4-4, las etapas que se deben seguir (representado mediante un diagrama de flujo) para llevar a cabo la medición genérica utilizando SMF son cuatro:

1. **Incorporación del metamodelo de dominio:** la medición se va a realizar sobre un dominio en concreto. Este dominio ha de ser definido mediante su correspondiente metamodelo (situado en el nivel M2 y bien formado respecto al meta-metamodelo ECORE). Ejemplos de dominios son: esquemas de bases de datos relacionales, diagramas de clases UML, diagramas de flujos de datos, código fuente en Java (para poder representar el código fuente Java en forma de modelo), manuales de usuario, etc. A la hora de definir un metamodelo es muy importante determinar cuáles son los conceptos que forman parte del dominio del problema que se aborda y cuáles son sus

relaciones. Una buena forma de realizar esta conceptualización del dominio es mediante la construcción previa de una ontología.

2. **Creación del modelo de dominio:** a partir del metamodelo de dominio (incorporado en la etapa 1) se define el modelo de dominio, que representa "*lo que se va a medir"*, es decir, el modelo sobre el cual se va a aplicar la medición. Por ejemplo, con un metamodelo de UML se podrán definir modelos de diagramas de clases, o para un metamodelo de esquemas relacionales se podrán definir modelos de bases de datos relacionales, etc.
3. **Creación del modelo de medición:** en base al metamodelo de medición integrado en SMF, se crea el correspondiente modelo de medición con la ayuda del lenguaje SMML. Esta primera versión del modelo de medición es un artefacto de entrada a la medición automática y, por ello, no incluye la parte de los resultados (instanciación del paquete *Acción de Medir* de SMM).
4. **Ejecución de la medición:** la ejecución de la medición se realiza mediante dos transformaciones de modelos de forma que, partiendo de los dos modelos de entrada (modelo de medición y modelo de dominio creados en las etapas 2 y 3), se obtiene un modelo de salida que es el modelo de medición ampliado con los resultados de la medición (instanciación del paquete *Acción de Medir*). En el apartado 4.5 se muestra el proceso completo de transformación de modelos.

4.5. Transformaciones de Modelos para realizar la Medición.

Las transformaciones de modelos son uno de los elementos clave en el paradigma MDE (Schmidt, 2006). El proceso de medición de SMF aprovecha el potencial de las transformaciones para llevar a cabo la medición genérica del software. La idea básica es aplicar transformaciones en el modelo inicial de medición para calcular los valores de las medidas definidas en dicho modelo. Esta transformación de modelos se realiza mediante QVT Relations.

Para lograr que la propuesta sea genérica, es decir, sirva para cualquier tipo de dominio software, es necesario que los modelos de transformación QVT, por definición diferentes para cada dominio, sean obtenidos de una manera automática y común. Para ello ha sido necesario idear una transformación previa que genere el modelo de transformación QVT específico para cada acción de medir. Por este motivo, la ejecución de la medición se lleva a cabo mediante dos transformaciones (véase Figura 4-5):

- **Transformación inicial.** Esta transformación (de tipo Modelo a Texto) permite obtener la especificación textual QVT (modelo QVT) a partir del modelo (inicial) de medición de software mediante el modelo MOFScript (ver apartado 4.3).
- **Transformación final o ejecución de la medición.** Esta transformación (de tipo Modelo a Modelo) permite calcular los resultados de la medición, tomando como entradas el modelo (inicial) de medición del software y el modelo de dominio, más la especificación QVT (modelo QVT) obtenida en la transformación anterior.

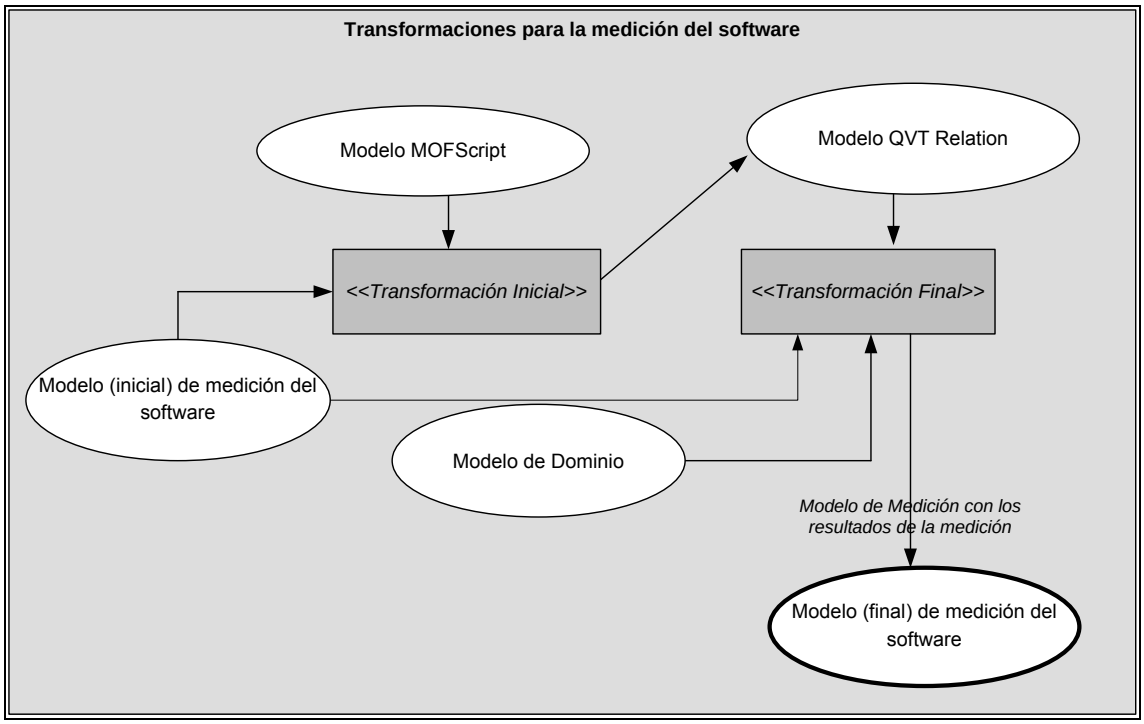


Figura 4-5. Transformaciones del proceso de medición de SMF

Como se observa en la Figura 4-5, gracias a este proceso de transformaciones en dos etapas, para el usuario el proceso de medición en SMF es una caja negra ya que lo único que debe hacer es definir los modelos de medición y de dominio y ejecutar la medición, el usuario no conoce el funcionamiento interno de las tareas que se están llevando a cabo (más detalles del del proceso de transformación se presentan en el Capítulo 6). Un ejemplo en un caso real de este proceso de medición se ilustra en el Capítulo 7.

4.6. Formas de Medir Genéricas.

Una "Forma de Medir" es una secuencia de operaciones cuyo objeto es determinar el valor del resultado de la medición. Existen tres clases de formas de medir según el tipo de medida a la que están asociadas: métodos de medición para las medidas base (véase apartado 4.6.1), funciones de cálculo para las medidas derivadas (véase apartado 4.6.2), y modelos de análisis para indicadores (véase apartado 4.6.3).

Usamos el término 'genéricas' para referir que se trata de trabajar con conceptos más abstractos (formas de medir en lugar de medidas/métricas) para conseguir soluciones más generales, es decir, válidas para muchos más casos. Así, la implementación de una forma de medir puede servir para obtener los resultados de decenas o cientos de medidas/métricas. La clave para conseguirlo está en trabajar a nivel de metamodelo en vez de a nivel de modelo. Usando un símil sencillo, es como si, al desarrollar un programa para cálculos aritméticos, en vez de hacer una implementación para sumar dos números ('sumar', 'dos' y 'número' son conceptos a nivel de modelo-expresión), hacemos una implementación 'genérica' para calcular expresiones aritméticas (trabajando con conceptos como operador, operando, delimitador, etc., que están a nivel del metamodelo de expresiones aritméticas).

En los siguientes tres sub-apartados se presentan, respectivamente, los métodos de medición genéricos, funciones de cálculo genéricas y modelos de análisis genéricos, incorporados hasta ahora en SMF.

4.6.1. Métodos de Medición Genéricos.

Un método de medición es la "forma de medir" una medida base. Es una secuencia lógica de operaciones, descritas de forma genérica, usadas para realizar mediciones de un atributo respecto de una escala específica.

En los siguientes apartados se muestran los métodos de medición genéricos contar entidades, contar referencias salientes y contar referencias entrantes. Con ellos es posible definir múltiples medidas base y derivadas de muchos dominios. En este sentido es importante tener en cuenta que la mayoría de los modelos hacen una representación sistémica (elementos y relaciones entre ellos) en forma de algún tipo de grafo. Con el método de medición "contar entidades" se abarca la parte de los elementos y con los otros la parte de las relaciones entre los elementos.

4.6.1.1. Contar Entidades.

Dado un modelo formado por un número determinado de instancias de tipo A (entidad o elemento del metamodelo) como muestra la Figura 4-6.

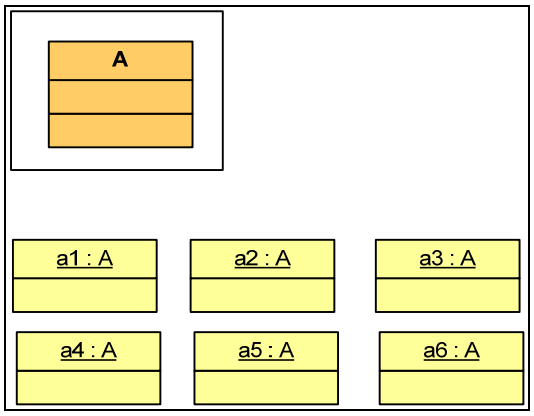


Figura 4-6. Metamodelo y modelo formado por instancias de tipo A

El método de medición **contar entidades de tipo A** se define como "el número de instancias de un cierto tipo A en un dominio concreto". Utilizando el lenguaje OCL puede expresarse de la siguiente manera:

```
X.allInstances()->size()
```

Este método de medición se propuso inicialmente en (García et al., 2006b). Con él se pueden calcular valores de gran cantidad de medidas base: aquellas cuyos resultados se obtienen contando el número de instancias de un cierto tipo X en un artefacto software (véase Tabla 4-1). Además, las medidas base que utilizan este método de medición sirven para definir gran cantidad de medidas derivadas e indicadores. En suma, se puede afirmar que con este método de medición es factible dar una solución global única (genérica) a muchas medidas de diversos tipos.

Para definir el método de medición **contar entidades** es necesario indicar la siguiente propiedad:

- **Tipo de Entidad:** tipo de entidad del metamodelo para el cual quiere calcularse el total de instancias en el modelo.

En la Tabla 4-1 se muestra una selección de medidas base que usan el método de medición contar en diferentes dominios, en la tabla se indica el tipo de entidad que se quiere medir (contar).

Medida	Dominio	Tipo de Entidad
NA Número de atributos en un esquema de bases de datos relacionales	Esquema de bases de datos relacionales	Atributo
NFK Número de claves ajenas en un esquema de bases de datos relacionales	Esquema de bases de datos relacionales	Clave Ajena
NOP Número de paquetes en una aplicación escrita en código java	Código Java	Paquete
NT Número de tablas en un esquema de bases de datos relacionales	Esquema de bases de datos relacionales	Tabla
NE Número de entidades de un esquema de entidad relación	ER	Entidad
NM Número de mensajes UML en un diagrama de secuencias	Diagrama de Secuencias UML	Mensaje
NR Número de roles en un plan de proyecto	Proceso de negocio	Roles
NDP Número de divisiones paralelas de un modelo de procesos de negocio	Procesos de negocio	División Paralela

Tabla 4-1. Selección de medidas base que usan el método de medición contar

Ejemplo

Para ilustrar el método de medición “contar entidades” se utiliza el dominio de bases de datos relacionales. En la Figura 4-7 se muestra un esquema relacional de ejemplo sobre el que se aplica la medición.

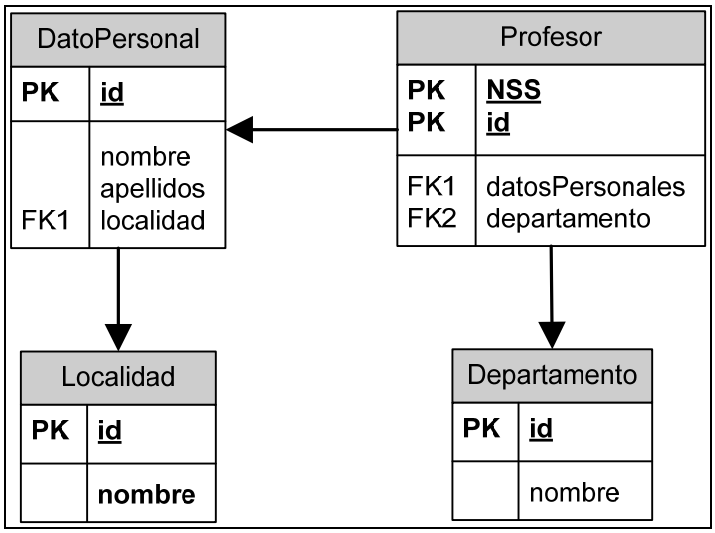


Figura 4-7. Modelo relacional (esquema de base de datos)

Para llevar a cabo la medición genérica con SMF es necesario disponer (o definir) el metamodelo del dominio (véase Figura 4-8) y representar el modelo (el esquema relacional de la Figura 4-7) en base a dicho metamodelo (véase Figura 4-8). Para distinguir los distintos tipos de entidades del modelo, en la Figura 4-9 se han utilizado diferentes colores.

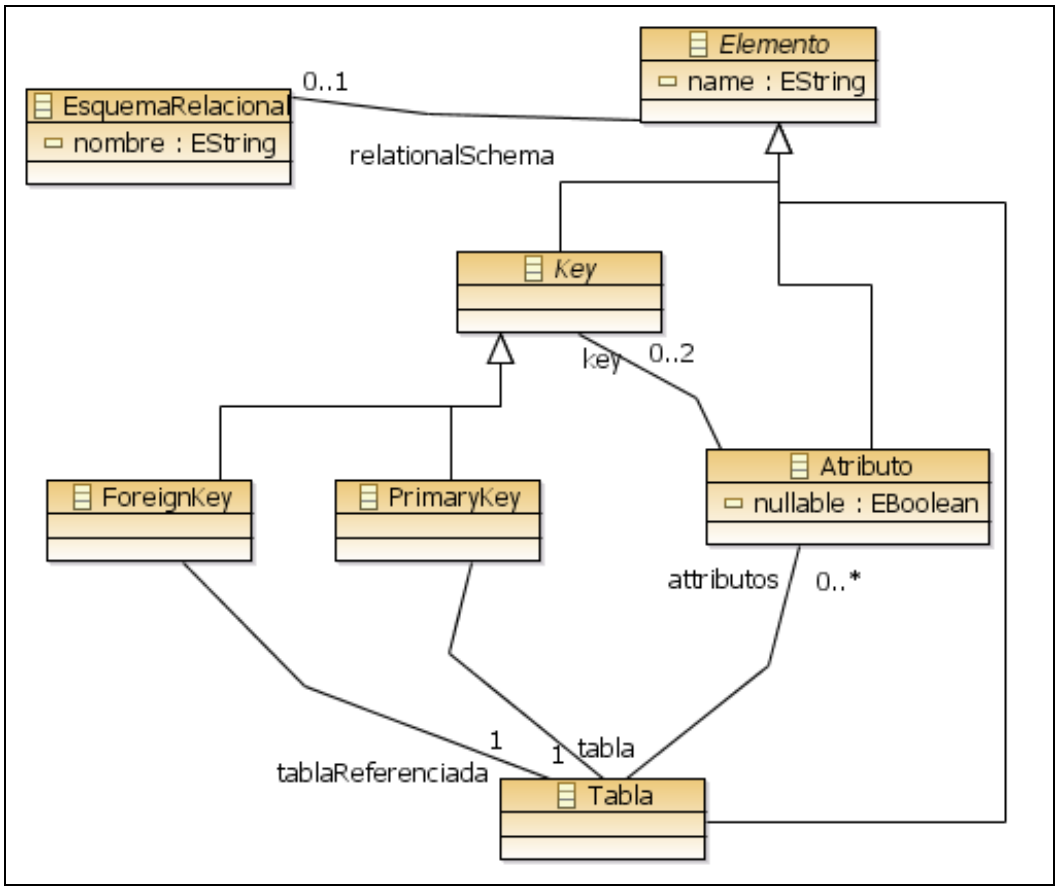


Figura 4-8. Metamodelo de esquemas de bases de datos relacionales

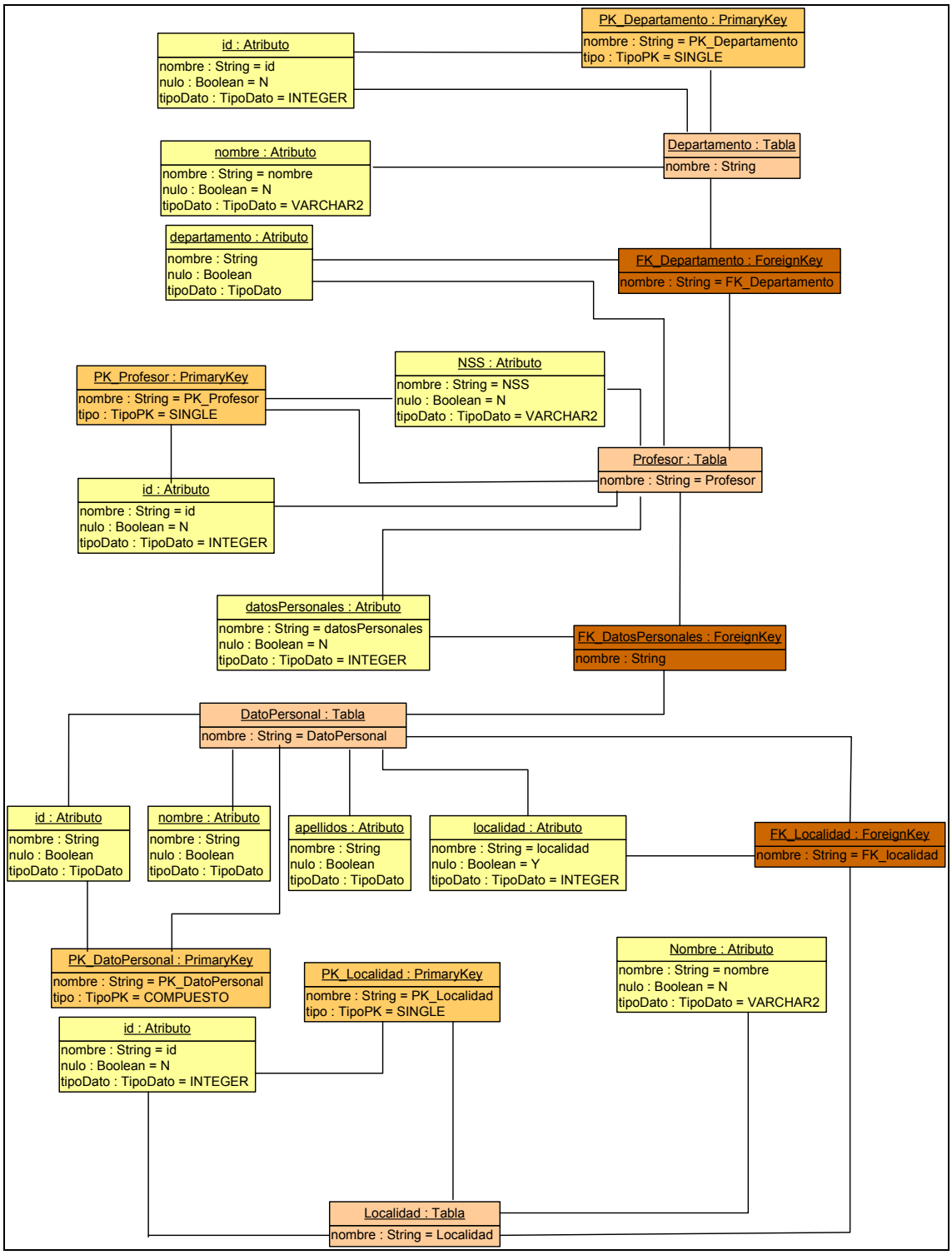


Figura 4-9. Modelo de base de datos relacional

En la Tabla 4-2 se muestran las instanciaciones para aplicar el método “contar entidades” en algunas de las más conocidas medidas base de modelos de bases de datos relacionales. Para cada medida se indica el tipo de entidad a usar al invocar el método de medición “contar entidades” y el resultado en el ejemplo utilizado.

Medida	Tipo de Entidad	Resultado
NA Número de atributos	Atributo	12
NFK Número de claves ajenas	ForeignKey (Clave Ajena)	3
NT Número de tablas	Tabla	4

Tabla 4-2. Ejemplo de medidas base que usan el método de medición contar

4.6.1.2. Contar Referencias Salientes.

Dado un modelo formado por un número determinado de instancias de tipo A y de tipo B relacionadas entre si mediante una asociación X (con referencias x_a y x_b)[13] tal y como muestra la Figura 4-10, el método de medición **contar referencias salientes desde un tipo de entidad A** se define como "el número de referencias de tipo x_b salientes de las entidades de un cierto tipo A (origen) hacia las entidades de otro cierto tipo B (destino)". Para el modelo mostrado en la Figura 4-10 el resultado de contar referencias de tipo Xb salientes de entidades de tipo A es 8.

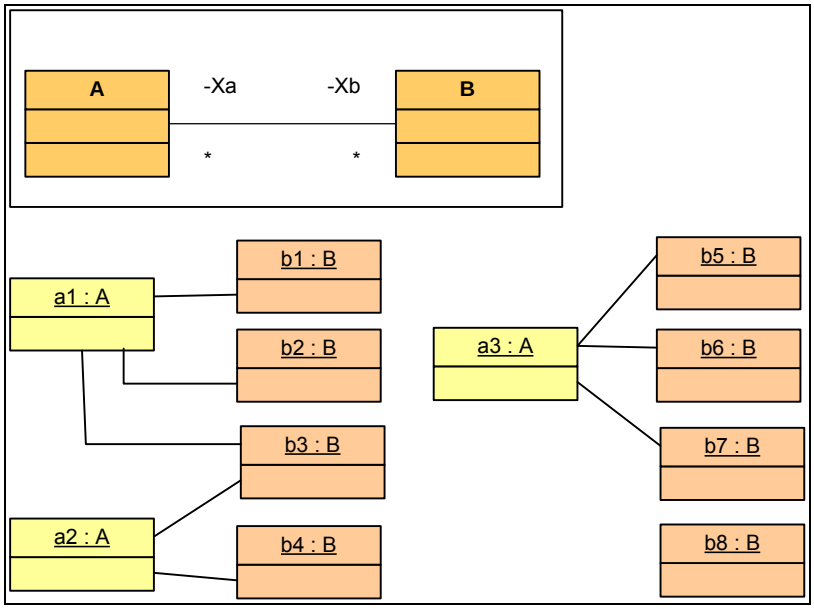


Figura 4-10. Metamodelo y modelo formado por entidades de tipo A relacionadas con entidades de tipo B mediante la referencia de tipo Xb

Utilizando OCL éste método de medición puede expresarse de la siguiente manera:

```
A.allInstances().xb->size().
```

[13] La asociación X definida en un metamodelo MOF estaría formada por las referencias origen y destino Xa y Xb. Por ello cuando se define la forma de medir se indica la referencia y no la asociación.

Para definir el método de medición **contar referencias salientes** es necesario indicar las siguientes propiedades:

- **Tipo de Referencia:** tipo de referencia en el metamodelo (x_b) que asocia el tipo de entidad origen (A) con el tipo de entidad destino (B).
- **Tipo de Entidad Origen:** tipo de entidad origen (A) que tiene relacionadas instancias del tipo de entidad destino (B) mediante referencias de tipo x_b. Es necesario indicar el tipo de entidad origen debido a que pueden existir varios tipos de referencias con el mismo nombre,
- **Tipo de Entidad Destino:** tipo de entidad destino (B) que es referenciada por la entidad origen (A) mediante la referencia de tipo x_b. Este dato no es necesario para el cálculo del método de medición contar referencias salientes, pero se debe añadir en el caso de que se desee añadir condiciones (parametrizaciones) sobre la entidad destino (véase apartado 4.6.4).

En la Tabla 4-3 se muestran algunas medidas base que usan este método de medición.

Medida	Dominio	Tipo de Entidad Origen	Tipo de Entidad Destino	Tipo de Referencia
NA Número de columnas de las tablas	Esquema de bases de datos relacionales	Tabla	Columna	columnasContenidas
NOM Número de métodos locales de las clases	Diagramas de clases UML	Clase	MétodoLocal	Contiene
NCI Número de clases que implementan interfaces	Diagramas de clases UML	Clase	Interfaz	Implementa
NCH Número de clases que heredan propiedades de otras clases	Diagramas de clases UML	Clase	Clase	Extiende

Tabla 4-3. Selección de medidas base que usan el método de medición contar referencias salientes

Ejemplo:

Para ilustrar el uso de este método vamos a seguir con el ejemplo de las bases de datos relacionales presentado en el apartado anterior y cuyo modelo de dominio está definido en la Figura 4-9. Se pretenden calcular las siguientes medidas:

- **NAPK:** número de atributos que forman parte de una clave primaria.
- **NAFK:** número de atributos que forman parte de una clave ajena.

Medida	Tipo de Entidad Origen	Tipo de Referencia	Resultado
NAPK Número de atributos que forman parte de las claves primarias.	PrimaryKey	atributos	5
NAFK Número de atributos que son clave ajena	ForeignKey	atributos	3

Tabla 4-4. Ejemplo de medidas base que usan el método de medición “contar referencias salientes”

Como se observa en la Tabla 4-4 el valor de NAPK=5 (DatoPersonal.id, Localidad.id, Profesor.id, Profesor.NSS y Departamento.id) se obtiene al contar todas instancias de la referencia “atributos” salientes del tipo PrimaryKey. El valor de NAFK = 3 que en este caso coincide con el número de claves ajenas.

4.6.1.3. Contar Referencias Entrantes.

Dado un modelo formado por un número determinado de instancias de tipo B que están relacionadas con un elemento de tipo: A_1 mediante la asociación X_1, A_2 mediante la asociación X_2, …, y con A_n mediante la asociación X_n tal y como muestra la Figura 4-11.

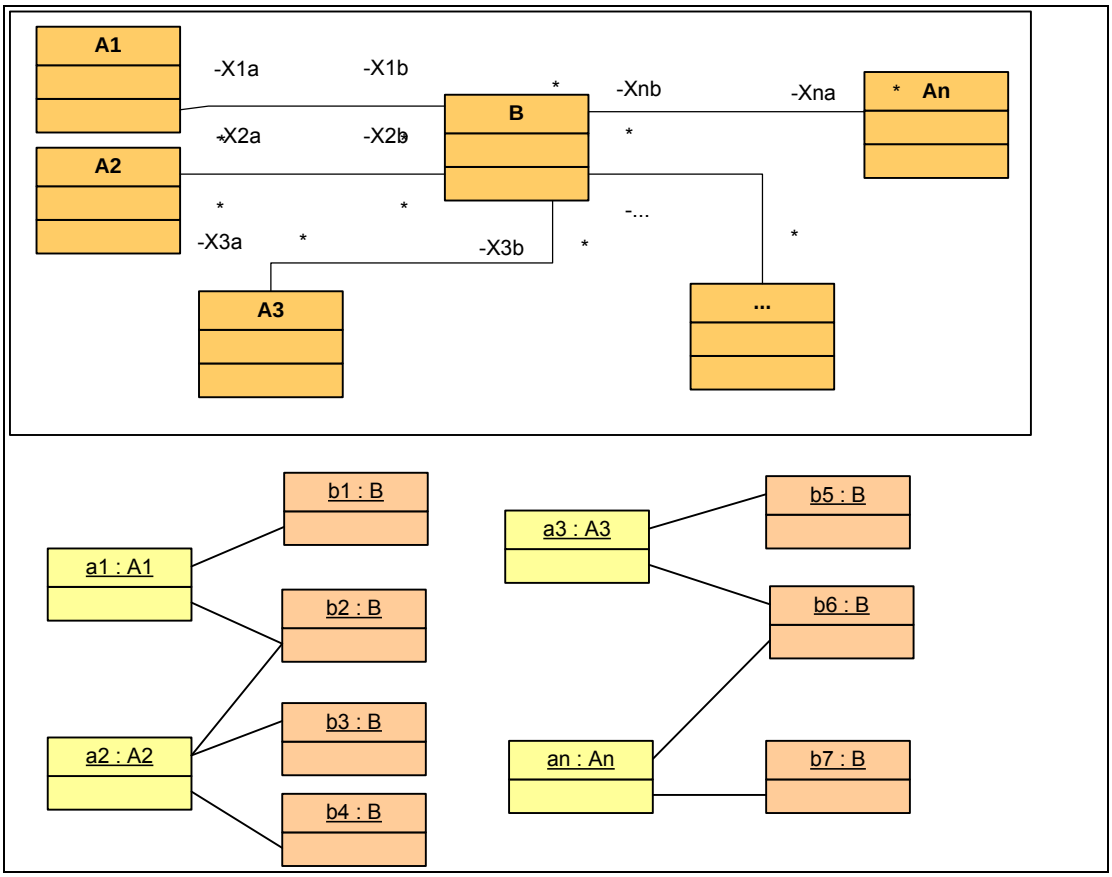


Figura 4-11. Metamodelo y modelo formado por entidades de tipo B relacionadas con entidades de tipo A_1, A_2, A_3 y A_n mediante las asociaciones X_1, X_2, X_3 y X_n

El método de medición **contar referencias entrantes hacia un tipo de entidad B** se define como "el número de referencias que hay hacia el tipo de entidad **B",** es decir el número de instancias referenciadas por los tipos de entidad A_1, A_2, …, A_n con el tipo de entidad **B** mediante las asociaciones X_1, X_2, …, X_n respectivamente (referencias de tipo X_{1b}, X_{2b}, …, X_{nb})

Utilizando el lenguaje OCL el método de medición contar referencias entrantes al tipo de entidad B puede expresarse de la siguiente manera:

```
A1.allInstances().x1b->size() + A2.allInstances().x2b->size() +
+ … + An.allInstances().xnb->size()
```

Para el modelo mostrado en la Figura 4-11 el resultado de contar el número de referencias entrantes al tipo de entidad B es 9.

Este método es parecido al anterior con la diferencia que calcula todas las entidades desde las cuales se puede acceder a un determinado tipo de entidad destino B.

Para definir el método de medición **contar referencias entrantes** es necesario indicar la siguiente propiedad:

- **Tipo de Entidad:** tipo de entidad destino (B) que tiene relacionadas instancias con: el tipo de entidad origen (A_1) mediante la referencia de tipo X_{1b}, el tipo de entidad origen (A_2) mediante la referencia de tipo X_{2b}, …, y el tipo de entidad origen (A_n) mediante la referencia de tipo X_{nb},

En la Tabla 4-5 se muestran medidas base que usan este método de medición.

Medida	Tipo de Entidad
NA_ref Número de referencias a las tablas de un esquema de bases de datos relacional.	Tabla
NOM_ref Número de referencias a las clases de un diagrama de clases UML.	Clase
NCI_ref Número de referencias a las interfaces de un diagrama de clases UML.	Interfaz

Tabla 4-5. Selección de medidas base que usan el método de medición contar referencias entrantes

Ejemplo:

Para ilustrar el uso de este método vamos a considerar el modelo de dominio definido en la Figura 4-7. La medición que se quiere llevar a cabo es el cálculo de las siguientes medidas:

- **NT_ref:** número de elementos que están conectados a las tablas de un esquema relacional de bases de datos.
- **NPK_ref:** número elementos que están conectados a las claves primarias de un esquema relacional de bases de datos.

Medida	Tipo de Entidad	Resultado
NT_ref Número de elementos que están conectados a las tablas de un esquema relacional de bases de datos.	Tabla	21
NPK_ref Número de elementos que están conectados a las claves primarias de un esquema relacional de bases de datos.	Clave Primaria	9

Tabla 4-6. Ejemplo de medidas utilizando el método de medición “contar referencias entrantes”

Como se observa en la Tabla 4-6 el valor de NT_ref=21 ya que el número de elementos conectados para cada tabla del esquema es el siguiente:

- **Tabla Departamento:** 4 elementos (2 atributos, 1 clave ajena y 1 clave primaria)
- **Tabla Profesor:** 6 elementos (3 atributos, 2 claves ajenas y 1 clave primaria)
- **Tabla DatoPersonal:** 7 elementos (4 atributos, 2 claves ajenas y 1 clave primaria)
- **Tabla Localidad:** 4 elementos (2 atributos, 1 clave ajena y 1 clave primaria)

El valor de NPK_ref= 9 ya que el número de elementos conectados por cada clave primaria es:

- **PK_Departamento:** 2 elementos (1 atributo y 1 tabla)
- **PK_Profesor:** 3 elementos (2 atributo y 1 tabla)
- **PK_DatoPersonal:** 2 elementos (1 atributo y 1 tabla)
- **PK_Localidad:** 2 elementos (1 atributo y 1 tabla)

4.6.1.4. Profundidad de una Entidad dada una Asociación entre Entidades del mismo tipo.

El concepto de profundidad de un elemento utilizado en SMF está basado en la teoría de grafos acíclicos dirigidos (Directed Aciclic Graph - DAG)[14] que define los siguientes conceptos:

- La profundidad de un vértice es la longitud del camino más largo que existe desde una fuente[15] a dicho vértice.
- La altura de un vértice es la longitud más larga del camino que existe desde el vértice a un sifón[16].

Aplicando estas ideas a modelos, donde los vértices son las entidades y los arcos las asociaciones, podemos definir el método de medición profundidad de un tipo de elemento.

[14] http://www.allisons.org/ll/AlgDS/Graph/DAG/ y http://en.wikipedia.org/wiki/Directed_acyclic_graph

[15] Una fuente es un vértice sin flujos (relaciones) de entrada

[16] Un sifón es un vértice sin flujos (relaciones) de salida

Dado un modelo formado por elementos de tipo **A** que tienen una asociación **X** entre entidades del mismo tipo (reflexiva) (con referencias x_{ini}, x_{fin}) como se muestra Figura 4-12.

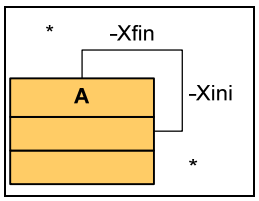


Figura 4-12. Tipo de entidad A con asociación X reflexiva

El método de medición **profundidad de un elemento 'a' de tipo A dada una asociación reflexiva X** (con referencias x_{ini} y x_{fin}) se define como "la longitud del camino más largo desde el elemento 'a' al elemento más alejado de 'a' (también de tipo A) asociados mediante la referencia de tipo x_{fin}

Para definir el método de medición **profundidad de una entidad dada una asociación reflexiva** es necesario definir las siguientes propiedades:

- **Entidad:** entidad 'a' de tipo A sobre la que se calcula la profundidad.
- **Tipo de Referencia:** tipo de referencia en el metamodelo (x_{fin}) que permite navegar desde la entidad 'a' a entidades de tipo A.

Para la Figura 4-13 la profundidad del elemento a_1 (de tipo A) dada la referencia de tipo x_{fin} es 4, que es la distancia de a_1 con el elemento más alejado a_5.

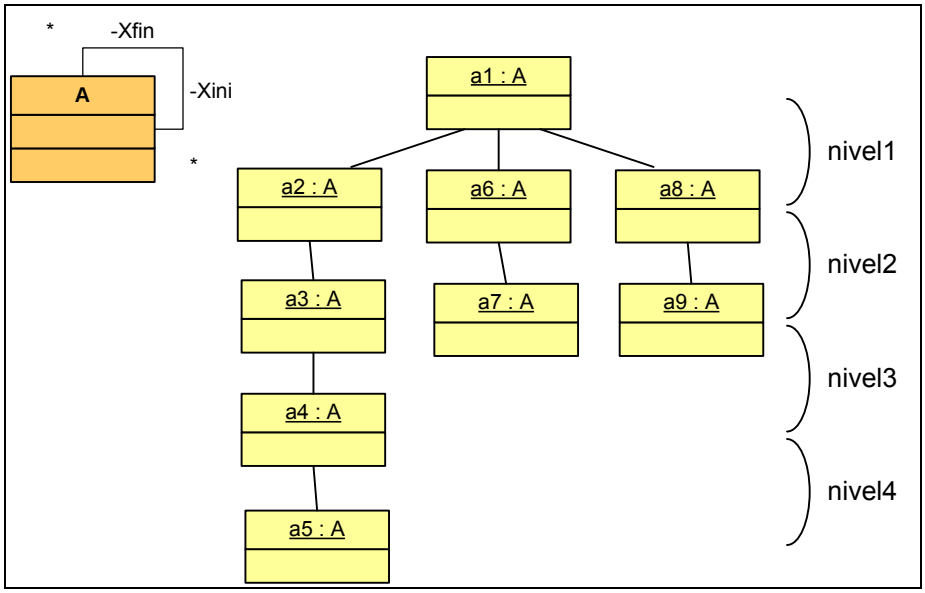


Figura 4-13. Metamodelo y modelo de entidad de tipo A con asociación reflexiva.

Este método de medición es una forma de medir intermedia para obtener la medida derivada calculada con la función de cálculo presentada en el apartado 4.6.2.2.

4.6.1.5. Profundidad de una Entidad dada una Asociación entre Entidades de distinto tipo.

Dado un modelo formado por un número determinado de instancias de tipo A y de tipo B relacionadas entre si mediante una asociación X entre entidades de distinto tipo (binaria) (con referencias x_a y x_b) tal y como muestra la Figura 4-14.

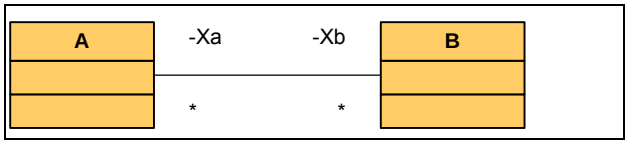


Figura 4-14. Tipo de entidad A con asociación X binaria

El método de medición **profundidad de un elemento 'a' de tipo A dada una asociación binaria X** (con referencias x_a y x_b) se define como "la longitud del camino más largo desde el elemento 'a' al elemento más alejado de 'a' (también de tipo A) asociados mediante la secuencia encadenada de referencias de tipo xb y xa".

Para definir el método de medición **profundidad de una entidad dada una asociación binaria** es necesario definir las siguientes propiedades:

- **Entidad:** entidad 'a' de tipo A sobre la que se calcula la máxima profundidad.
- **Tipo de Referencia destino:** referencia destino de la asociación X (x_b)
- **Tipo de Referencia origen:** referencia origen de la asociación X (x_a)

Para el ejemplo de la Figura 4-15 la profundidad del elemento a_2 sería 2 (profundidad a su último elemento a_{10}). Tal como se puede observar cada nivel de profundidad se considera compuesto por un encadenamiento de referencias de tipo x_b y x_a.

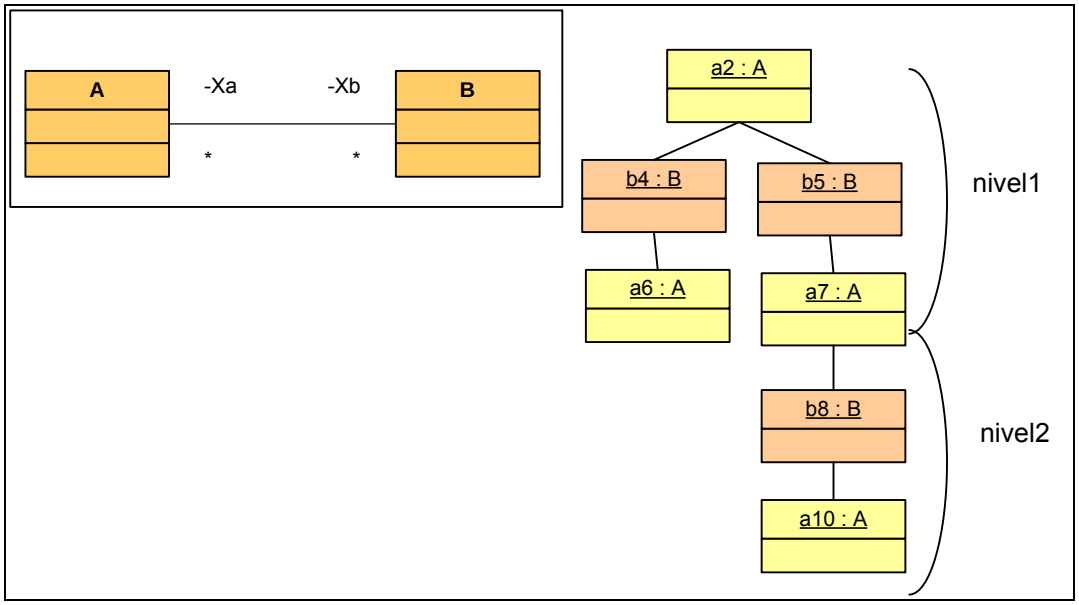


Figura 4-15. Metamodelo y modelo formado por entidades de tipo A relacionadas con entidades de tipo B mediante la referencia de tipo Xb

Este método de medición es una forma de medir intermedia para obtener la medida derivada calculada con la función de cálculo presentada en el apartado 4.6.2.3.

4.6.2. Funciones de Cálculo Genéricas.

Una función de cálculo es la *"forma de medir"* una medida derivada. Consiste en un algoritmo o cálculo realizado para combinar dos o más medidas base y/o derivadas. Además de las funciones de cálculo aritméticas (que utilizan medidas base como operandos), SMF incluye tres funciones de cálculo adicionales

- Máximo: esta función de cálculo calcula el valor máximo de una lista de valores, presentada en el apartado 4.6.2.1
- Máxima profundidad de un tipo de entidad dada una asociación reflexiva, presentada en el apartado 4.6.2.2.
- Máxima profundidad de un tipo de entidad dada una asociación entre dos entidades, presentada en el apartado 4.6.2.3.

4.6.2.1. Máximo.

La función de cálculo **máximo de una lista de medidas** (base, derivada y/o indicador) se define como "el máximo valor de un conjunto de medidas".

Para definir la función de cálculo **máximo** es necesario definir la siguiente propiedad:

- **Medidas:** lista de medidas. La lista debe estar formada por dos o más medidas.

4.6.2.2. Máxima Profundidad de un tipo de Entidad dada una Asociación entre Entidades del mismo tipo.

En este apartado se presenta el cálculo de la máxima profundidad de una entidad con una asociación reflexiva, tal y como se mostraba en la Figura 4-12.

La función de cálculo **máxima profundidad del tipo de entidad A dada una asociación X reflexiva** (con referencias x_{ini} y x_{fin}) se define como "el máximo valor de la profundidad de todos los elementos de tipo A ($a_1, a_2, \ldots, a_n$) enlazados mediante la referencia de tipo x_{fin}". Esta función de cálculo combina dos formas de medir:

- la función de cálculo **máximo,** (definido en el apartado 4.6.2.1).
- y el método de medición **profundidad de una entidad dada una asociación reflexiva** (definido en el apartado 4.6.1.4).

Para definir la función de cálculo **máxima profundidad de un tipo de entidad dada una asociación reflexiva** es necesario definir las siguientes propiedades:

- **Tipo de Entidad:** tipo de entidad (A) sobre la que se calcula la máxima profundidad.
- **Tipo de Referencia:** tipo de referencia en el metamodelo (x_{fin}) que permite navegar desde la entidad origen (A) a la misma entidad.

En la Figura 4-16 **la máxima profundidad del tipo de entidad A dada la referencia de tipo x_{fin} es 5**, que sería la profundidad del elemento a_{10} (distancia a su elemento más alejado a_{15}), y coincide con el máximo valor de la profundidad del elemento $a_1, a_2, \ldots, a_{16}$.

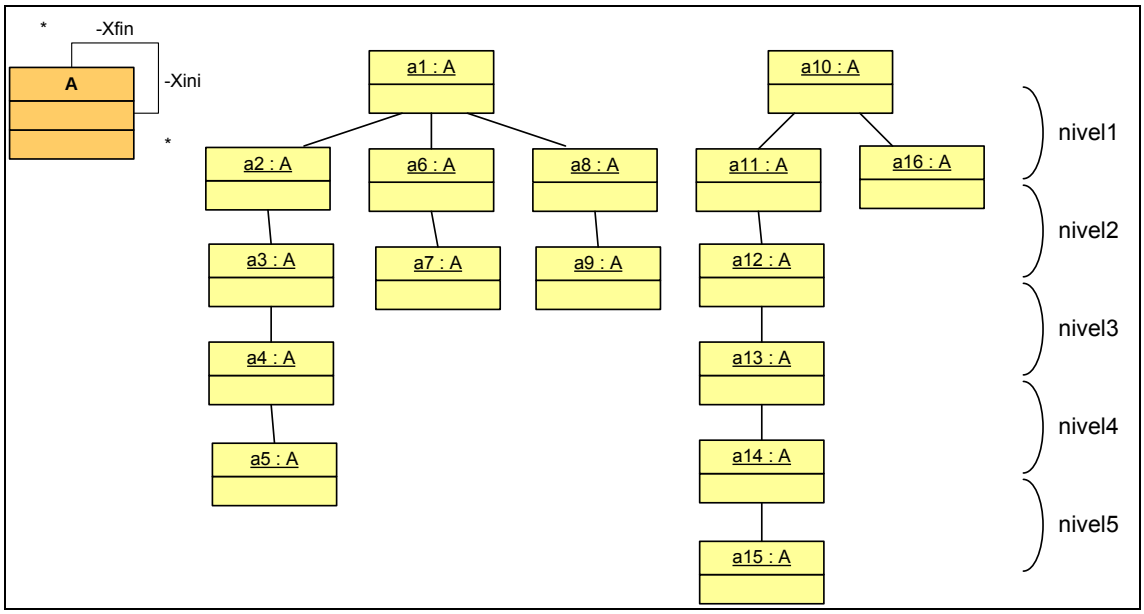


Figura 4-16. Metamodelo y modelo de entidad de tipo A con asociación reflexiva.

En la siguiente tabla se muestra una medida derivada que se calcula mediante esta función de cálculo.

Medida	Tipo de Referencia	Tipo de Entidad
DIT Profundidad de herencia de una clase en un diagrama de clases UML.	Superclass	Class

Tabla 4-7. Medida derivada calculada con la función de cálculo máxima profundidad de una entidad dada una asociación reflexiva

4.6.2.3. Máxima Profundidad de un tipo de Entidad dada una Asociación entre Entidades de distinto tipo.

En este apartado se presenta el cálculo de la máxima profundidad de una entidad con una asociación binaria con otra entidad, tal y como se mostraba en la Figura 4-14. En este caso la función de cálculo máxima profundidad de una entidad se calcula igual que para el caso anterior con la diferencia que en este caso la asociación es binaria, y por tanto se genera una secuencia encadenada de referencias x_b y x_a. Esta función de cálculo combina dos formas de medir:

- la función de cálculo **máximo,**
- y el método de medición **profundidad de una entidad dada una asociación binaria** (definido en el apartado 4.6.1.4).

Para definir la función de cálculo máxima profundidad de una entidad dada una asociación binaria es necesario definir las siguientes propiedades:

- **Tipo de Entidad:** tipo de tipo A sobre la que se calcula la máxima profundidad.

- **Tipo de Referencia destino:** tipo de referencia destino al tipo de entidad B (x_b).
- **Tipo de Referencia origen:** tipo de referencia origen desde el tipo de entidad A(x_a).

Para el ejemplo de la Figura 4-17 la **máxima profundidad del modelo dado el tipo de entidad A y las referencias x_b y x_a sería 3**, que es la profundidad del elemento a_1 (profundidad de a_1 a su último elemento a_{11}), y coincide con el máximo de la profundidad del elemento a_1, a_2, …, a_{11}.

Tal como se puede observar en la Figura 4-17 cada nivel de profundidad se considera compuesto por un encadenamiento de referencias de tipo x_b y x_a.

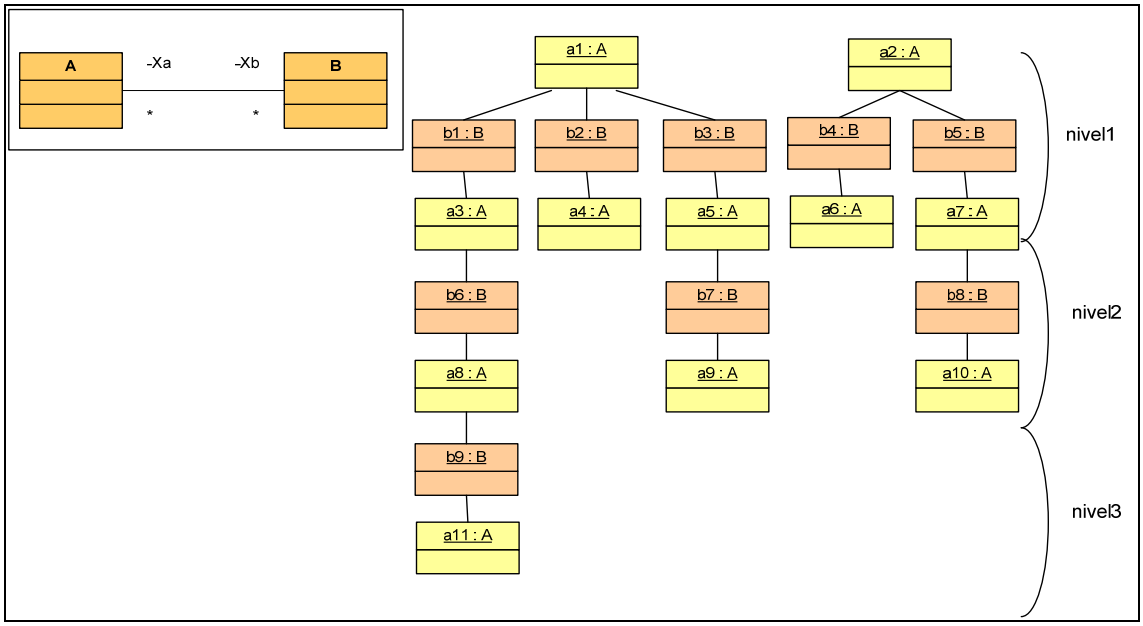


Figura 4-17. Metamodelo y modelo formado por entidades de tipo A relacionadas con entidades de tipo B mediante la referencia de tipo Xb

En la Tabla 4-8 se muestra un ejemplo de medida derivada que se calcula con esta función de cálculo.

Medida	Referencia Origen	Referencia Destino	Tipo de Entidad
DRT Calcular la máxima profundidad posible de Clave Ajena sobre una tabla en un esquema de bases de datos relacional	fks	Tabla_referenciada	Tabla

Tabla 4-8. Medida obtenidas con la función de cálculo máxima profundidad dada una entidad y asociación binaria

Ejemplo:

Para ilustrar el uso de este método vamos a considerar el modelo de dominio definido en la Figura 4-9. La medición que se quiere llevar a cabo es el cálculo de la siguiente medida derivada:

- **DRT:** se define como la longitud del camino máximo referencial de una tabla (las tablas están relacionadas por medio de claves ajenas). Los ciclos sólo se consideran una vez.

Medida	Referencia Origen	Referencia Destino	Tipo de Entidad	Resultado
DRT Calcular la máxima profundidad posible de clave ajena sobre las tablas.	foreingKeys	tablaReferenciada	Tabla	2

Tabla 4-9. Ejemplo de medida derivada calculada con la función de cálculo “máxima distancia”

Como se observa en la Tabla 4-9 el valor de **DRT** es 2, ya que la máxima profundidad del tipo de entidad Tabla dado las dos referencias permite obtener una secuencia de navegación tal como se indica a continuación:

Profesor → FK_DatosPersonales→ DatoPersonal (primer nivel)

DatoPersonal → FK_Localidad → Localidad (segundo nivel)

4.6.3. Modelos de Análisis Genéricos.

Un modelo de análisis es la “forma de medir” un indicador. Consiste en un algoritmo o cálculo realizado para combinar una o más medidas (base, derivadas o indicadores) con criterios de decisión asociados. Los “criterios de decisión” son valores umbral, objetivos, o patrones, usados para determinar la necesidad de una acción o investigación posterior, o para describir el nivel de confianza de un resultado dado. En la Tabla 4-10 se muestra un ejemplo de modelo de análisis para un indicador.

Actualmente SMF da soporte a modelos de análisis que utilizan operadores aritméticos, tal como el modelo presentado en la Tabla 4-10, LCFH/LCFHvm.

	Modelo de Análisis	
Indicador	**Fórmula**	**Criterios de Decisión**
PROD Productividad de los programadores.	Fórmula para obtener la medida PROD LCFH/LCFHvm	LCFH/LCFHvm < 0’70 → PROD=’muy baja’. 0’70 ≤ LCFH/LCFHvm < 0’90 → PROD=’baja’. 0’90 ≤ LCFH/LCFHvm < 1’10 → PROD=’normal’. 1’10 ≤ LCFH/LCFHvm <1’30 → PROD=’alta’. 1’30 ≤ LCFH/LCFHvm→ PROD=’muy alta’.

Tabla 4-10. Ejemplo de un indicador calculado con un modelo de análisis.

4.6.4. Formas de Medir parametrizadas.

La definición de formas de medir parametrizadas aumenta el potencial de definición de medidas añadiendo restricciones o condiciones que la forma de medir debe satisfacer en el nivel de metamodelo del dominio para obtener el resultado de las medidas.

Las condiciones que permiten la parametrización de las formas de medir se aplican sobre los tipos de entidad de un metamodelo. Se deben indicar los siguientes elementos:

- **Atributo:** atributo de la entidad sobre el cual se quiere aplicar la condición. Por ejemplo, el atributo "nombre" de la entidad "Clase".
- **Operador:** operador que se quiere validar. Los operadores de comparación que se soportan son los siguientes.
  - = igual que.
  - < menor que.
  - <= menor o igual que.
  - > mayor que.
  - >= mayor o igual que
  - <> distinto que.
- **Valor:** valor que queremos que satisfaga la condición.
- **Conexión lógica de condiciones:** en el caso que se defina más de una condición, se pueden utilizar los siguientes operadores lógicos:
  - AND
  - OR
  - XOR
  - NOT

Un ejemplo de parametrización o condición para un tipo de entidad "Clase" puede ser "nombre=Cliente", de manera que se pueda contar únicamente los tipo de entidad "Clase" cuyo nombre es "cliente".

En las siguientes tablas se muestran algunos ejemplos de medidas que se pueden obtener a partir de formas de medir parametrizadas. Estas medidas son las medidas presentadas en los apartados anteriores, para distinguirlas se ha añadido la terminación **_p**. En primer lugar, la Tabla 4-11 muestra la parametrización del método de medición contar.

Medida	Tipo de Entidad	Parametrización del Tipo de Entidad		
NA_p Número de atributos de tipo CHAR en un esquema de bases de datos relacionales	Atributo	Atributo	Operador	Valor
		tipoDato	=	CHAR
NFK_p Número de clave primarias que son compuestas en un esquema de bases de datos relacionales	PrimaryKey	Atributo	Operador	Valor
		tipoPK	=	COMPUESTA
NOP_p Número de clases públicas en una aplicación escrita en código java.	Clase	Atributo	Operador	Valor
		Estereotipo	=	Public
NT_p Número de tablas que no son Cliente en un esquema de bases de datos relacionales	Tabla	Atributo	Operador	Valor
		Nombre	<>	Cliente

Tabla 4-11. Selección de medidas base que usan el método de medición contar con parametrización

A continuación, en la Tabla 4-12 se muestran ejemplos de parametrización del método de medición contar referencias salientes (que se puede aplicar a cualquiera de las dos entidades asociadas o sobre ambas).

Medida	Tipo de Entidad Origen	Parametrización del Tipo de Entidad Origen	Tipo de Entidad Destino	Parametrización del Tipo de Entidad Destino	Tipo de Referencia
NA_p Número de columnas de la Tabla Cliente en un esquema de bases de datos relacional	Tabla	**Atributo:** nombre **Operador:** = **Valor:** Cliente	Columna	Sin parametrización	columnasContenidas
NOM_p Número de métodos locales de la Clase C en un diagrama de clases UML.	Clase	**Atributo:** nombre **Operador:** = **Valor:** C	Clase	**Atributo:** tipo **Operador:** = **Valor:** estático	Contiene
NCI_p Número de clases que implementan la interfaz “WebService” en un diagrama de clases UML.	Clase	**Atributo:** tipo **Operador:** = **Valor:** publica	Clase	**Atributo:** nombre **Operador:** = **Valor:** WebService	implementa

Medida	Tipo de Entidad Origen	Parametrización del Tipo de Entidad Origen	Tipo de Entidad Destino	Parametrización del Tipo de Entidad Destino	Tipo de Referencia
NCH_p Número de clases que heredan propiedades de la clases padre "Elemento" en un diagrama de clases UML.	Clase	**Atributo:** nombre **Operador:** = **Valor:** Elemento	Clase	Sin parametrización	extiende

Tabla 4-12. Selección de medidas base que usan el método de medición contar referencias salientes con parametrización

La Tabla 4-13 muestra ejemplos de aplicación de la parametrización del método contar referencias entrantes.

Medida	Tipo de Entidad	Parametrización del Tipo de Entidad		
NA_ref_p Número de referencias a la Tabla Cliente de un esquema de bases de datos relacional.	Tabla	Atributo	Operador	Valor
		Nombre	=	Cliente
NOM_ref_p Número de referencias a las clases públicas de un diagrama de clases UML.	Clase	Atributo	Operador	Valor
		Tipo	=	pública
NCI_ref_p Número de referencias a la interfaz "WebService" de un diagrama de clases UML.	Interfaz	Atributo	Operador	Valor
		Nombre	=	WebService

Tabla 4-13. Selección de medidas base que usan el método de medición contar referencias entrantes

Por último, en la Tabla 4-14 presentan dos casos de parametrización en la función de cálculo máxima profundidad.

Medida	Tipo de Referencia	Tipo de Entidad	Parametrización del Tipo de Entidad		
DRT_p Máxima profundidad posible de Clave Ajena sobre la tabla Banco de un esquema de bases de datos relacional	fks	Tabla	Atributo	Operador	Valor
			Nombre	=	Banco
DIT_p Máxima Profundidad de herencia de la clase Cliente de un diagrama UML	extends	Clase	Atributo	Operador	Valor
			Nombre	=	Cliente

Tabla 4-14. Selección de medidas derivadas que se calculan con la función de cálculo máxima profundidad

Ejemplo:

Para ilustrar el uso de formas de medir parametrizadas se considera el modelo de dominio definido en la Figura 4-7. Se aplican condiciones a las medidas de las tablas Tabla 4-2, Tabla 4-4, Tabla 4-6 y Tabla 4-9. En la Tabla 4-15 se muestra, por cada medida, las propiedades (tipo de entidad origen y destino y tipo de referencia), la parametrización (si aplica), y el resultado. En el caso que exista para un tipo de entidad más de una parametrización se indicará con el símbolo '&'.

Medida	Propiedades de la Forma de Medir y Parametrización	Resultado
NA_p Número de atributos llamados **id** de un esquema de bases de datos relacionales	**Tipo de Entidad:** Atributo **Atributo:** nombre **Operador:** = **Valor:** id	4 (Persona.id, DatoPersonal.id, Localidad.id y Departamento.id)
NFK_p Número de Claves Ajenas llamadas **FK_Departamento** de un esquema de bases de datos relacionales	**Tipo de Entidad:** ForeignKey **Atributo:** nombre **Operador:** = **Valor:** FK_Departamento	1 (FK_Departamento)
NT_p Número de tablas llamadas **DatoPersonal** de un esquema de bases de datos relacionales	**Tipo de Entidad:** Tabla **Atributo:** nombre **Operador:** = **Valor:** DatoPersonal	1 (DatoPersonal)
NAPK_p Número de atributos que forman parte de la clave primaria compuesta PK_DatoPersonal. de un esquema de bases de datos relacionales	**Tipo de Referencia:** atributos **Tipo de Entidad:** PrimaryKey **Atributo:** nombre **Operador:** = **Valor:** PK_DatoPersonal **&** **Atributo:** tipoPK **Operador:** = **Valor:** Compuesto	2 (id y nombre)
NAFK_p Número de atributos que forman parte de la clave ajena FK_DatosPersonales	**Tipo de Referencia:** atributos **Tipo de Entidad:** ForeignKey **Atributo:** nombre **Operador:** = **Valor:** FK_DatosPersonales	1 (datosPersonales)

Medida	Propiedades de la Forma de Medir y Parametrización	Resultado
NT_ref' Número elementos que están conectados a la tabla Localidad de un esquema de bases de datos relacionales	**Tipo de Entidad:** Tabla **Atributo:** nombre **Operador:** = **Valor:** Localidad	4 (2 atributos [id y nombre], la clave ajena FK_Localidad y la clave primaria PK_Localidad)
NPK_ref_p Número elementos que están conectados a la clave primaria PK_Profesor de un esquema de bases de datos relacionales	**Tipo de Entidad:** ClavePrimaria **Atributo:** nombre **Operador:** = **Valor:** PK_Profesor	3 (Tabla Profesor, y atributo id y NSS)
DRT_p Máxima profundidad posible de clave ajena sobre la tabla DatoPersonal de un esquema de bases de datos relacionales	**Tipo de Entidad:** Tabla **Atributo:** nombre **Operador:** = **Valor:** DatoPersonal **Tipo de Referencia Origen:** fks **Tipo de Referencia Destino:** tablaReferenciada	1 nivel Tabla DatoPersonal → FK_Localidad → Tabla Localidad

Tabla 4-15. Ejemplo de medidas aplicando formas de medir parametrizadas.

4.7. Conclusiones.

En este capítulo se ha presentado SMF, un entorno genérico para la definición de modelos de medición basados en un metamodelo común y para la medición de cualquier entidad software a partir de los modelos que las representan. Siguiendo el enfoque de MDA, y partiendo del metamodelo de Medición, es posible llevar a cabo la medición de cualquier dominio mediante transformaciones QVT, proceso que es completamente transparente al usuario.

Además, se han presentado formas de medir genéricas que permiten calcular medidas independientemente del dominio y se ha ilustrado su utilidad mediante ejemplos aplicados a diversos dominios.

En conclusión, con SMF es posible medir cualquier entidad software si se dispone de los correspondientes metamodelos y modelos. La tarea del usuario consiste, únicamente, en la elección del metamodelo de dominio (según lo que se quiere medir en cada momento) y la definición de los modelos de entrada: de dominio y de medición. El metamodelo de medición está integrado en el entorno.

“Si he hecho descubrimientos invaluables ha sido más por tener paciencia que cualquier otro talento” (Isaac Newton)

5. Software Measurement Modeling Language (SMML).

En este capítulo se presenta SMML, el lenguaje de dominio específico desarrollado para definir modelos de medición de una manera gráfica, más fácil e intuitiva. En primer lugar se describe el lenguaje con su sintaxis y semántica. En segundo lugar se presenta la validación empírica llevada a cabo para validar la usabilidad de este nuevo lenguaje frente a una notación textual. En concreto, se describe un experimento para comparar la entendibilidad y modificabilidad en la definición de modelos de medición utilizando SMML frente a una notación textual.

5.1. SMML.

Como se ha presentado en el Capítulo 4, uno de los puntos clave en el proceso de medición genérica de SMF es la definición de modelos de medición basados en el metamodelo SMM. En este contexto, para facilitar al usuario la realización del proceso de medición es útil disponer de una notación apropiada, que haga más intuitiva y fácil la definición de modelos de medición del software. En este sentido, el paradigma MDE ofrece dos alternativas para definir o desarrollar lenguajes: los Lenguajes de Propósito General (GPL) y los Lenguajes de Dominio Específico (DSL). Un GPL es un lenguaje que se puede utilizar para abordar una gran variedad de problemas. Un ejemplo es UML. Por contra, un DSL es un lenguaje que ofrece, mediante notaciones y abstracciones correctas, una potente expresividad y está restringido a un problema de dominio particular (Deursen et al., 2000).

Cuando se diseñan DSLs se debe identificar el origen o fuente del lenguaje, es decir, se necesita saber qué expresa el DSL, e identificar qué notaciones y abstracciones son relevantes en el DSL (Völter, 2009). El origen de un lenguaje puede ser un framework existente, una librería, una arquitectura o un patrón de arquitectura. Además, existen patrones para ayudar en las fases de decisión, análisis, diseño e implementación durante el desarrollo de un DSL (Mernik et al., 2005). En este sentido, los metamodelos de definición de dominio (Domain Definition Metamodel - DDMM), que son una especificación de la conceptualización del dominio (Kurtev et al., 2006), se pueden considerar de interés para obtener la abstracción del lenguaje. Una vez que el conocimiento se ha identificado, la construcción del DSL se basa principalmente en la formalización del conocimiento: definición de la notación (gráfica o textual), formalización del lenguaje y construcción de generadores o editores para implementar el código del lenguaje (Völter, 2009).

Ambas clases de lenguajes, GPL y DSL, se pueden utilizar para aplicar MDE en el contexto de la medición, pero la opción de un DSL se ha considerado más apropiada por las razones siguientes:

- Un DSL tiene una clara identificación con un problema de dominio concreto (como es el caso de la medición de software).
- Gracias a la existencia de la ontología SMO, ya se dispone de una conceptualización y semántica bien definida.
- El metamodelo SMM, derivado de SMO, se puede emplear como DDMM.

Un importante aspecto a considerar en el desarrollo de un DSL es determinar el tipo de notación, gráfica o textual, que es mejor para facilitar el entendimiento y facilidad de mantenimiento de los modelos. La respuesta no siempre está clara ya que depende de la información que se vaya a manejar en cada momento. Para representar información que gestione relaciones entre entidades, que represente tiempos/secuencias de eventos o para cualquier tipo de flujo de señal/datos, se suele considerar que la notación gráfica es mejor (Völter, 2009). Sin embargo, la representación de expresiones gráficas no es una buena solución, por tanto, para analizar expresiones es mejor la notación textual. En este tema es interesante tener en cuenta las propuestas del área de programación visual, que aporta dos ventajas importantes frente a deficiencias habituales en el campo de la Ingeniería del Software (Ahmad, 1999): i) permite a los usuarios controlar la complejidad inherente en una actividad de programación por medio de una notación gráfica (el alto nivel de las abstracciones visuales ayuda a comprender el programa de manera fácil por medio de la visualización de las relaciones semánticas entre entidades de programas); y ii) favorece la productividad de los desarrolladores y usuarios.

Teniendo en cuenta lo anterior, en esta tesis se ha desarrollado el lenguaje SMML, acrónimo de "Software Measurement Modelling Language" (Mora et al., 2008a; Mora et al., 2008e). Es un lenguaje que permite definir de manera gráfica los modelos de medición del software en base a SMM. Como se ha comentado, una notación gráfica puede ser una buena opción para representar entidades y relaciones entre ellos. Esto convierte a SMML en un elemento clave de SMF.

Para crear el lenguaje SMML y con el objetivo de obtener un DSL usable se han seguido las siguientes etapas: definición de la semántica y definición de una sintaxis abstracta y concreta. En los siguientes apartados se explica con detalle.

5.1.1. Semántica.

Uno de los aspectos más importantes en la especificación de un lenguaje es la definición de su semántica (Feilkas, 2006). La semántica de un lenguaje debe proporcionar el significado de cada expresión, de forma que ese significado debe estar en algún dominio bien definido y entendido (Harel y Rumpe, 2004). En este sentido, las ontologías básicamente proporcionan la semántica y tiene un doble papel: pueden describir tanto la semántica de los constructores del lenguaje como la semántica de las instancias de los modelos (Kramler et al., 2006). En otras palabras, la ontología expresa la semántica de los conceptos de modelado cuya sintaxis está definida por el metamodelo.

Teniendo en cuenta lo anterior, podemos afirmar que la ontología SMO expresa la semántica de los conceptos de SMML, cuya sintaxis está definida por SMM.

Por otro lado, el editor de SMML (presentado en el Capítulo 6) y las transformaciones QVT que han sido diseñadas para ejecutar el modelo de medición proporcionan toda la semántica operacional del lenguaje.

En la Tabla 5-1 y en la Tabla 5-2 se presenta la definición de todos los elementos de SMML: entidades y asociaciones. Para más información véase SMO (García et al., 2006a).

Entidad	Definición
Necesidad de Información	Información necesaria para gestionar un proyecto (sus objetivos, hitos, riesgos y problemas).
Concepto medible	Relación abstracta entre *atributos* y *necesidades de información*.
Categoría de Entidad	Una colección de *entidades* caracterizadas por satisfacer un cierto predicado común. Es el dominio
Atributo	Una propiedad mensurable, física o abstracta, que comparten todas las *entidades* de una *categoría de entidad*.
Modelo de calidad	Un conjunto de *conceptos medibles* y relaciones entre ellos que proporciona la base para especificar requisitos de calidad y evaluar la calidad de las *entidades* de una determinada *categoría de entidad*.
Escala	Un conjunto de valores con propiedades definidas.
Unidad de medición	Una cantidad particular, definida y adoptada por convención, con la que se puede comparar otras cantidades de la misma clase para expresar sus magnitudes respecto a esa cantidad particular
Medida base	Una medida de un *atributo* que no depende de ninguna otra medida, y cuya *forma de medir* es un *método de medición*.
Medida derivada	Una medida que es derivada de otra medida base o derivada, utilizando una *función de cálculo* como *forma de medir*.

Entidad	Definición
Indicador	Una medida que es derivada de otras medidas utilizando un *modelo de análisis* como *forma de medir*.
Método de Medición	La *forma de medir* una *medida base*. Secuencia lógica de operaciones, descritas de forma genérica, usadas para realizar mediciones de un atributo respecto de una escala específica.
Función de Cálculo	La *forma de medir* una *medida derivada*. Algoritmo o cálculo realizado para combinar dos o más medidas base y/o derivadas.
Modelo de Análisis	La *forma de medir* un *indicador*. Algoritmo o cálculo realizado para combinar una o más medidas (base, derivadas o indicadores) con *criterios de decisión* asociados.
Criterio de decisión	Valores umbral, objetivos, o patrones, usados para determinar la necesidad de una acción o investigación posterior, o para describir el nivel de confianza de un resultado dado.

Tabla 5-1. Semántica de las entidades de SMML

Asociación	Descripción
Incluye	Una *categoría de entidad* puede **incluir** una o varias *categorías de entidad*, y puede **estar incluida** en una o varias *categorías de entidad*.
Definido para	Un *modelo de calidad* **está definido** para una determinada *categoría de entidad*. Una *categoría de entidad* puede **tener definidos** varios *modelos de calidad*.
Evalúa	Un *modelo de calidad* **evalúa** uno o varios *conceptos medibles*. Un *concepto medible* **es evaluado** por uno o más *modelos de calidad*.
Relaciona	Un *concepto medible* **relaciona** uno o más *atributos*. Un *atributo* **está relacionado** con uno o más *conceptos medibles*.
Está relacionado con	Un *concepto medible* **está relacionado con** una o varias *necesidades de información*. Una *necesidad de información* **se relaciona con** un *concepto medible*.
Incluye	Un *concepto medible* puede **incluir** a varios *conceptos medibles* y puede **estar incluido** en otros *conceptos medibles*.
Tiene	Una *categoría de entidad* **tiene** uno o varios *atributos*. Un *atributo* solo puede pertenecer a una *categoría de entidad*.
Se define para	Una *medida* **está definida para** uno o más *atributos*. Un *atributo* puede tener varias *medidas* asociadas.
Calculada con	Una *medida derivada* es calculada con una *función de cálculo*. Una *función de cálculo* define una o más medidas derivadas.
Calculado con	Un *indicador* es calculado con un *modelo de análisis*. Un *modelo de análisis* puede definir uno o más *indicadores*.
Usa	Una *medida base* usa un *método de medición*. Un *método de medición* define una o más *medidas base*.
Satisface	Un *indicador* satisface *necesidades de información*. Una *necesidad de información* se satisface con uno o más *indicadores*.
Usa	Una *función de cálculo* puede **usar** varias *medidas derivadas* y/o varias *medidas base*. Una *medida derivada/medida base* puede **usarse** en *funciones de cálculo*.
Usa	Un *modelo de análisis* **usa** una o más *medidas*. Una *medida* puede **usarse** en *modelos de análisis*.
Usa	Un *modelo de análisis* usa uno o más *criterios de decisión*. Un *criterio de decisión* puede usarse en uno o más *modelos de análisis*.

Tabla 5-2. Semántica de las asociaciones de SMML

5.1.2. Sintaxis Abstracta.

La sintaxis abstracta describe los conceptos y las relaciones existentes entre ellos. Para el desarrollo del lenguaje se ha tomado como DDMM el Metamodelo de Medición del software (SMM) que permite el modelado de instancias de medición del software y contiene constructores para todos los conceptos y relaciones de SMO agrupadas en cuatro tipos de asociación (dependencias, agregación, asociación navegable y no navegable).

En la Tabla 5-3 se presenta la sintaxis abstracta de las entidades de SMML. Además, por cada entidad se muestra el valor de la etiqueta y los atributos por los que está formado. La etiqueta es el nombre reservado que se utiliza en el lenguaje para identificar a la entidad. La sintaxis abstracta define para cada entidad con qué entidades se puede conectar y qué tipo de asociación utiliza para hacerlo. Entendemos que un determinado elemento A está relacionado con otro elemento B mediante una asociación X siendo A el origen y B el destino de la asociación.

Entidad	Etiqueta	Sintaxis	Atributos
Necesidad de Información	*InformationNeed*	Una *Necesidad de Información* está relacionado con un *Concepto Medible* mediante un tipo de asociación de *dependencia*	**name:** nombre de la entidad
Concepto medible	*MeasurableConcept*	Un *Concepto Medible* está relacionado con uno o varios *atributos* mediante un tipo de asociación *navegable*	**name:** nombre de la entidad
Categoría de Entidad	*EntityClass*	Una *Categoría de Entidad* está relacionado con uno o varios *atributos* mediante un tipo de asociación *no navegable*	**name:** nombre de la entidad **measurementDomain:** nombre del dominio que se quiere medir
Atributo	*Attribute*	Un *atributo* no está relacionado con ningún elemento.	**name:** nombre de la entidad
Modelo de calidad	*QualityModel*	Un *Modelo de Calidad* está relacionado con una *Categoría de Entidad* mediante un tipo de asociación de *dependencia* Un *Modelo de Calidad* está relacionado con uno o varios *Conceptos Medibles* mediante un tipo de asociación de *dependencia*	**name:** nombre de la entidad
Escala	*scale*	La *escala* es un atributo obligatorio de una *Medida*	**name:** nombre de la entidad
Unidad de medición	*unit*	La *unidad* es un atributo obligatorio de una *Medida*	**name:** nombre de la entidad

Entidad	Etiqueta	Sintaxis	Atributos
Medida base	*BaseMeasure*	Una *Medida Base* está relacionado con un *Método de Medición* mediante un tipo de asociación de *navegable*	**name:** nombre de la entidad **scale:** escala de la medida **unit:** unidad de la medida **value:** valor de la medida
Medida derivada	*DerivedMeasure*	Una *Medida Derivada* está relacionado con una *Función de cálculo* mediante un tipo de asociación de *navegable*	**name:** nombre de la entidad **scale:** escala de la medida **unit:** unidad de la medida **value:** valor de la medida
Indicador	*Indicator*	Un *Indicador* está relacionado con un *Modelo de Análisis* mediante un tipo de asociación de *navegable* Un *Indicador* está relacionado con una o varias *Necesidades de Información* mediante un tipo de asociación de *dependencia*	**name:** nombre de la entidad **scale:** escala de la medida **unit:** unidad de la medida **value:** valor de la medida **finalResult:** valor del indicador obtenido a partir del modelo de análisis
Método de Medición	*MeasurementMethod*	Un *Método de Medición* no está relacionado con ningún elemento.	**name:** nombre de la entidad **operation:** operador utilizado para calcular la medida base. **entity:** entidad sobre la que se está operando
Función de Cálculo	*MeasurementFunction*	Una *Función de Cálculo* está relacionado con dos *Medidas Base y/o Derivadas* mediante un tipo de asociación *navegable*	**name:** nombre de la entidad **operation:** operador utilizado para calcular la medida derivada.
Modelo de Análisis	*AnalisisModel*	Un *Modelo de Análisis* está relacionado con uno o dos *Medidas (Base, Derivada o Indicador)* mediante un tipo de asociación *navegable* Un *Modelo de Análisis* está relacionado una *Medida Derivada* mediante un tipo de asociación *navegable* Un *Modelo de Análisis* está relacionado con cuatro *Criterios de Decisiones* mediante un tipo de asociación de *dependencia*	**name:** nombre de la entidad **operation:** operador utilizado para calcular el indicador.

Entidad	Etiqueta	Sintaxis	Atributos
Criterio de decisión	*DecisionCriteria*	Un *Criterio de Decisión* no está relacionado con ningún elemento	**name:** nombre de la entidad **minValue:** umbral inferior **maxValue:** umbral superior **descriptor:** valor del indicador si el valor está dentro del umbral

Tabla 5-3. Sintaxis abstracta de elementos de SMML

En la Tabla 5-4 se muestran la sintaxis abstracta de las asociaciones de SMML, el tipo de asociación y la etiqueta.

Asociación	Tipo Asociación	Etiqueta	Descripción
Definido para	Dependencia	*QualityModel.entityClass*	Tiene como elemento origen un *Modelo de Calidad* y como elemento destino una *Categoría de Entidad*
Evalúa	Dependencia	*QualityModel.measurableConcepts*	Tiene como elemento origen un *Modelo de Calidad* y como elemento destino uno o más *Conceptos Medibles*
Relaciona	Asociación navegable	*MeasurableConcept.attributes*	Tiene como elemento origen un *ConceptoMedible* y como elemento destino uno o más *Atributos*
Está relacionado con	Dependencia	*InformationNeed.measurableConceptAssociation*	Tiene como elemento origen una *Necesidad de Información* y como elemento destino un *Concepto Medible*
Tiene	Asociación no navegable	*EntityClass.attributes*	Tiene como elemento origen una *Categoría de Entidad* y como elemento destino uno o más *Atributos*
Se define para	Dependencia	*Measure.attributes*	Tiene como elemento origen una *Medida* y como elemento destino uno o más *Atributos*

Asociación	Tipo Asociación	Etiqueta	Descripción
Calculada con	Asociación navegable	*DerivedMeasure.measurementFuncion*	Tiene como elemento origen una *Medida Derivada* y como elemento destino una *Función de Cálculo*
Calculado con	Asociación navegable	*Indicator.analysisModel*	Tiene como elemento origen un *Indicador* y como elemento destino un *Modelo de Análisis*
Usa	Asociación navegable	*BaseMeasure.measurementMethod*	Tiene como elemento origen una *Medida Derivada* y como elemento destino un *Método de Medición*
Satisface	Dependencia	*Indicator.informationNeeds*	Tiene como elemento origen un *Indicador* y como elemento destino una o más *Necesidades de Información*
Usa	Asociación navegable	*MeasurementFunction. operableMeasures*	Tiene como elemento origen una *Indicador* y como elemento destino dos *Medidas Base* y/o *Derivadas*
Usa	Asociación navegable	*AnalisysModel.measures*	Tiene como elemento origen una *Modelo de Análisis* y como elemento destino dos *Medidas Base* y/o *Derivadas* y/o *Indicadores*
Usa	Dependencia	*AnalysisModel.DecisionCriteria*	Tiene como elemento origen un *Modelo de Análisis* y como elemento destino cuatro *Criterios de Decisión*

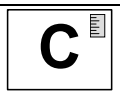
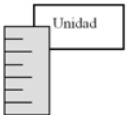
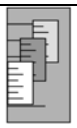
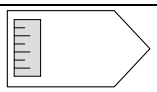
Tabla 5-4. Sintaxis abstracta de las asociaciones de SMML

5.1.3. Sintaxis Concreta.

Como ya se ha comentado, para poder usar un lenguaje es necesario disponer de una sintaxis concreta. Por este motivo, a cada uno de los elementos y relaciones del lenguaje (todos definidos en el paquete básico del metamodelo) se le debe asociar un icono gráfico que represente visualmente el elemento del modelo abstracto. El símbolo asociado con cada constructor se ha elegido con especial cuidado para que resulte lo más familiar e intuitivo posible a los ingenieros del software y así facilitarles su uso. Por ejemplo, el elemento *Categoría de Entidad* es muy similar al de una *Entidad* del diagrama E/R con la diferencia de que este añade la imagen de una regla (como símbolo de medición) en la esquina superior

derecha, y está particularizado para expresar entidades medibles. Por otro lado, para definir las asociaciones, se han estudiado los tipos de relaciones definidas en la ontología y su implementación en el metamodelo. El resultado está resumido en las siguientes tablas, que muestran el icono, definición y fuente de todas las entidades y relaciones del lenguaje SMML.

En la Tabla 5-5 se muestran la sintaxis concreta de las entidades de SMML.

Entidad	Icono	Fuente
Necesidad de Información		Nuevo
Concepto medible		Nuevo
Categoría de Entidad		E/R
Atributo		E/R
Modelo de calidad		TQM
Escala		Nuevo
Unidad de medición		Nuevo
Medida base		Nuevo
Medida derivada		Nuevo
Indicador		Nuevo
Método de Medición		Nuevo

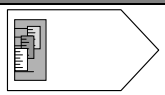
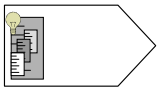
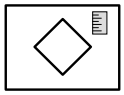
Entidad	Icono	Fuente
Función de Cálculo		Nuevo
Modelo de Análisis		Nuevo
Criterio de decisión		Nuevo

Tabla 5-5. Sintaxis concreta de elementos de SMML

En la Tabla 5-6 se muestran la sintaxis concreta de las asociaciones de SMML.

Asociación	Icono	Fuente
Incluye	◇———	Agregación UML
Definido para	- - - - - - - - - - →	Dependencia UML
Evalúa	- - - - - - - - - - →	Dependencia UML
Relaciona	——————→	Asociación UML
Está relacionado con	- - - - - - - - - - →	Dependencia UML
Incluye	◇———	Agregación UML
Tiene	———————	Asociación sin navegabilidad UML
Se define para	- - - - - - - - - - →	Dependencia UML
Calculada con	——————→	Asociación UML
Calculado con	——————→	Asociación UML
Usa	——————→	Asociación UML
Satisface	- - - - - - - - - - →	Dependencia UML
Usa	- - - - - - - - - - →	Dependencia UML

Tabla 5-6. Sintaxis concreta de las asociaciones de SMML

Como se observa hay 4 formas de representar las asociaciones de SMML: asociación navegable, no navegable, dependencia y agregación. Se puede consultar un ejemplo de diagrama SMML en la Figura 5-2.

5.1.3.1. Restricciones OCL.

Las restricciones OCL definidas sobre SMM para el desarrollo de SMML permiten crear un amplio conjunto de reglas para un modelo. En la Tabla 5-7 se muestran las restricciones OCL definidas para algunas entidades de SMML.

Entidad	Restricción OCL	Descripción
Criterio de Decisión	`self.minValue < self.maxValue`	El umbral inferior tiene que ser menor que el umbral superior
Método de Medición	`self.operation = 'COUNT'`	La operación del método de medición tiene que ser COUNT. Actualmente es la única operación que está implementada.

Función de Cálculo	`self.m1 <> self.m2`
	`and`
	`self.m1 <> null and self.m2 <> null`
		Las medidas usadas por la función de cálculo tienen que ser distintas y no nulas.

Función de Cálculo	`self.operation = '+' or self.operation = '-' or self.operation = '*' or self.operation = '/'`	Las operaciones permitidas por la función de cálculo son las aritméticas: suma, resta, división y multiplicación.

Modelo de Análisis	`self.operation='' and self.m1 <> null`
	`and`
	`self.operation<>'' and self.m1 <> null and self.m2 <> null`
	`and`
	`self.m1 <> self.m2`
		Si no hay operación debe existir una medida que es de la que se coge el valor.
		Si hay operación deben estar definidas las dos medidas y estas deben ser diferentes

Modelo de Análisis	`self.operation = '+' or self.operation = '-' or self.operation = '*' or self.operation = '/' or self.operation = ''`	Las operaciones permitidas por el modelo de análisis son las aritméticas: suma, resta, división y multiplicación.
Medida Derivada	`self <> self.measurementFunction.m1 and self <> self.measurementFunction.m2`	Las medidas derivadas usadas en la función de cálculo no puede ser la misma que calcula la función
Indicador	`self <> self.analysisModel.m1 and self <> self.analysisModel.m2`	Los indicadores usados en el modelo de análisis no puede ser el mismo que calcula el modelo de análisis

Entidad	Restricción OCL	Descripción

Modelo de Análisis	`self.dc1 <> null and self.dc2 <> null and self.dc3 <> null and self.dc4`
	
	`and`
	`(self.dc1 <> null and self.dc2 <> null and dc1.maxValue <=dc2.minValue ) or`
	`(self.dc1 <> null and self.dc2 <> null and self.dc3<>null and dc1.maxValue <=dc2.minValue and dc2.maxValue <=dc3.minValue ) or`
	`(self.dc1 <> null and self.dc2 <> null and self.dc3<>null and self.dc4<>null and dc1.maxValue <=dc2.minValue and dc2.maxValue <=dc3.minValue and dc3.maxValue <=dc4.minValue )`
		Deben existir un mínimo de dos y un máximo de cuatro criterios de decisión, y los valores de los umbrales tienen que ser correlativos.

Tabla 5-7. Restricciones OCL definidas en SMML

5.2. Validación Empírica del Lenguaje.

En este apartado se presenta la validación de la usabilidad y mantenibilidad de SMML por medio de un estudio empírico, centrado en las sub-características de entendibilidad y modificabilidad, según se definen en la norma ISO 9126 (ISO/IEC, 2001). Siguiendo el principio de replicabilidad, básico para cualquier investigación científica empírica, se proporciona información detallada, incluyendo el planteamiento, material, método y análisis, y la interpretación de los resultados (Basili et al., 1999; Shull et al., 2008).

El estudio se ha llevado a cabo gracias a la colaboración de un grupo de expertos en Ingeniería del Software que trabajaron con modelos de medición del software definidos en SMML y en una notación TEXTUAL (datos organizados en tablas con un determinado formato pero con texto libre distribuido en las celdas). El principal objetivo de este estudio empírico es comprobar si los modelos representados con SMML son más fáciles de entender y modificar que los modelos representados con una notación textual. Por tanto, nuestra pregunta de investigación se puede establecer como sigue: *"¿Es SMML más usable, y facilita la construcción de modelos de medición del software más mantenibles?"*

Para llevar a cabo el experimento se siguieron las etapas de un experimento propuestas en (Wohlin et al., 2000) (ver Capítulo 2) y que se detallan en los apartados siguientes.

5.2.1. Definición.

El objetivo general del estudio experimental fue: **"Analizar** modelos de medición representados con y sin SMML **con el propósito de** compararlos, **con respecto a** la usabilidad y mantenibilidad de los modelos, **desde el punto de vista de** los responsables de medición software, en **el contexto** de los expertos en proyectos de desarrollo de software".

5.2.2. Planificación.

El experimento se realizó teniendo en cuenta las plantillas y recomendaciones presentadas en (Wohlin et al., 2000; Juristo y Moreno, 2001; Kitchenham et al., 2002; Jedlitschka y Ciolkowski, 2005). La Figura 5-1 muestra un resumen del plan experimental. Cada elemento del plan experimental se explica con mayor detalle en los sub-apartados siguientes.

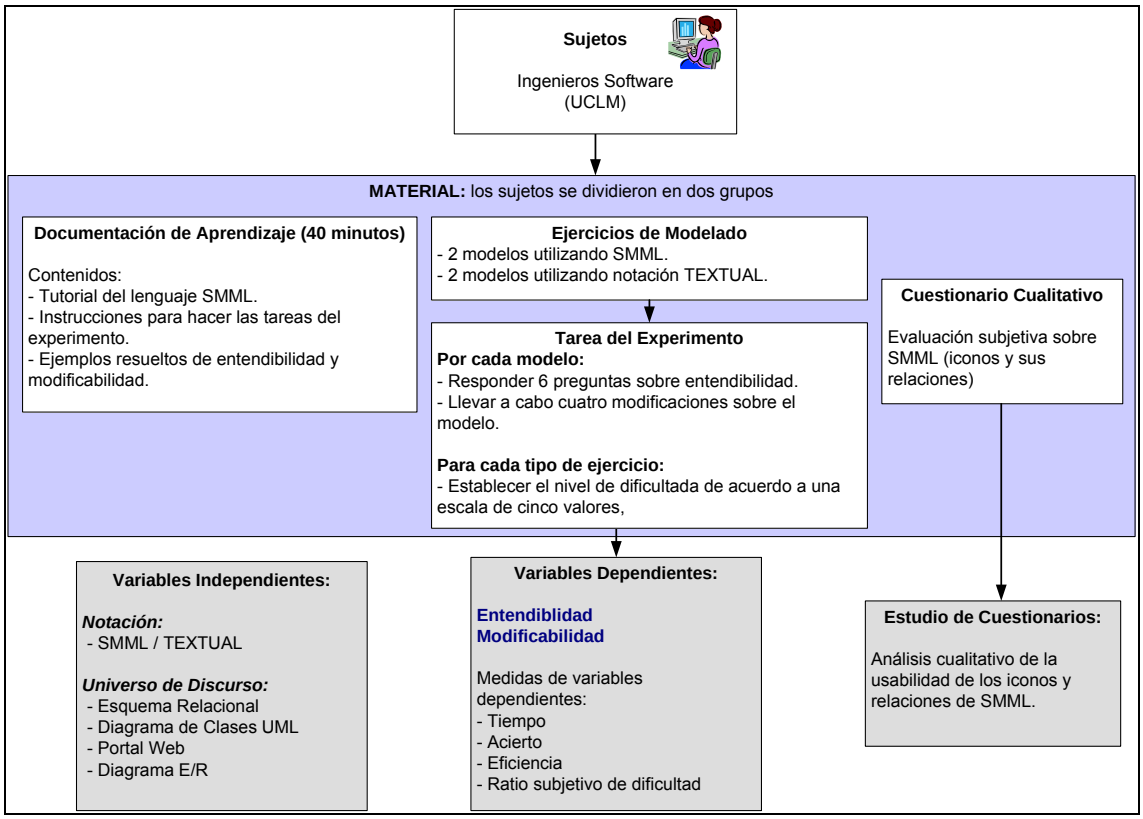


Figura 5-1. Resumen del plan experimental

5.2.2.1. Sujetos.

Los sujetos del experimento fueron 20 miembros del Departamento de Tecnologías y Sistemas de Información de la Universidad de Castilla-La Mancha. El grupo de sujetos elegido debía cumplir los siguientes requisitos:

- Ser Ingenieros/as en Informática.
- Tener experiencia en el modelado.
- Tener conocimiento de medición del software (Ontología y Metamodelo de Medición del Software).
- No tener conocimiento de SMML.

5.2.2.2. Material.

El material proporcionado a los sujetos está formado de tres partes: documentación de aprendizaje, ejercicios de modelado y cuestionario cualitativo. Cada parte se detalla en los siguientes puntos.

a) Documentación de Aprendizaje

Esta parte del material se preparó para proporcionar a los sujetos la información necesaria para aprender SMML y llevar a cabo las tareas del experimento. Está pensado para ser leído en aproximadamente 40 minutos y está compuesto de las siguientes partes:

- **Introducción al lenguaje SMML.** Se explica el objetivo del lenguaje y de la ontología. La razón de incluir esta parte fue para equilibrar el conocimiento necesario de todos los sujetos para llevar a cabo las tareas experimentales, evitando así diferencias de nivel del conocimiento de la ontología.
- **Estructura de SMML.** Proporciona una breve explicación de los 3 paquetes del lenguaje que permiten la creación de modelos de medición, y un ejemplo de un diagrama de cada paquete utilizando el lenguaje SMML.
- **Instrucciones.** Pasos a seguir para realizar el experimento.
- **Un ejemplo resuelto de un ejercicio de entendibilidad.**
- **Un ejemplo resuelto de un ejercicio de modificabilidad.**

b) Ejercicios de Modelado

A cada sujeto se le repartieron dos bloques de ejercicios. El primer bloque estaba formado por cuatro ejercicios para evaluar la entendibilidad del modelo y el segundo bloque estaba formado por cuatro ejercicios para evaluar la modificabilidad del modelo. Cada bloque se realizó sobre cuatro universos de discurso (Universe of Discourse – UoD), uno por cada ejercicio. Se eligieron UoDs que fueran familiares a los ingenieros de software: Esquema relacional de bases de datos, Diagrama de casos UML, Portal Web y Diagrama entidad/interrelación

Por cada UoD se prepararon dos modelos de medición del software semánticamente equivalentes: uno utilizando el lenguaje SMML, y el otro utilizando notación TEXTUAL (texto libre organizado en tablas). Así, cada sujeto trabajó con un total de ocho modelos de medición del software (cuatro para evaluar la entendibilidad y cuatro para evaluar la modificabilidad). La Tabla 5-8 muestra la distribución del material por sujeto.

		UoD			
		Esquema Relacional	**Diagrama de Clases UML**	**Portal Web**	**Diagrama E/R**
Notación	**SMML**	1	2	4	6
	TEXTUAL	3	4	7	8

Tabla 5-8. Distribución del material repartido a cada sujeto

Cada ejercicio de entendibilidad incluía un cuestionario compuesto de seis preguntas (de respuesta si/no) sobre el modelo de medición. Además, al final del ejercicio se incluyó una sección en la que los sujetos tenían que indicar, en opinión subjetiva, el nivel de dificultad del modelo de acuerdo a una escala de cinco valores (desde muy simple a muy complejo). La Figura 5-2 y la Figura 5-3 muestran un ejemplo de ejercicio de entendibilidad para un modelo definido con SMML.

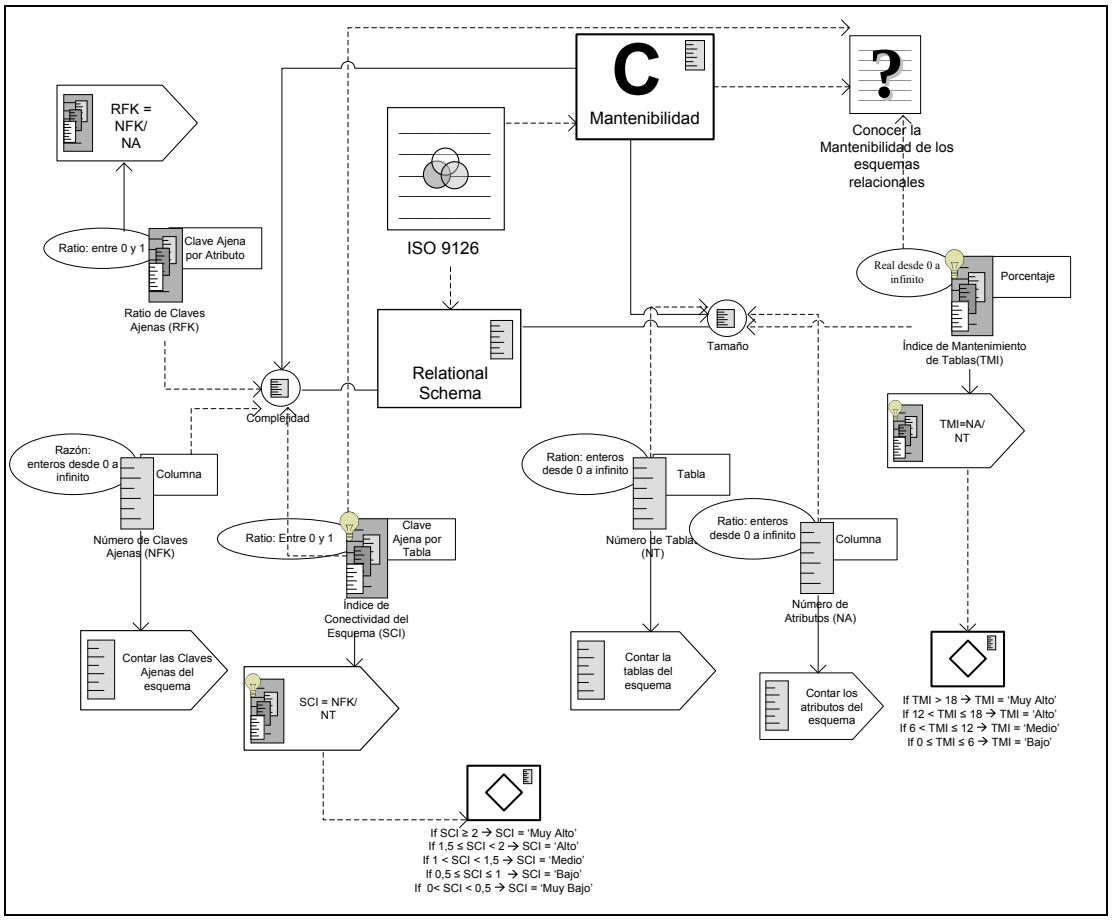


Figura 5-2. Ejemplo de ejercicio de modificabilidad con un Diagrama SMML

EJERCICIO 1: Tareas a realizar:

Anotar la hora de inicio (indique hh:mm:ss): _____

1) Realice las modificaciones necesarias en las tablas (indique el número de apartado en cada modificación), representadas al final del ejercicio, para satisfacer los siguientes requisitos de medición del software:

a) Se desea eliminar *la medida base* NT, elimina si fuera necesario las medidas que usan esta *medida base*

b) Se desea añadir la *medida base* NPK (Número de claves primaria) **definida para** el *atributo* Complejidad y que **usa** el *método de medición* contar el número de claves primarias. Con *escala* Ratio: enteros desde 0 a infinito y *unidad* clave primaria.

c) Se desea modificar el *modelo de calidad* ISO 9126 para que **evalúe** los conceptos medibles Mantenibilidad *y* Portabilidad

d) Se desea modificar la *Necesidad de información* conocer la portabilidad de los esquemas relacionales para que **se relaciona** con el *concepto medible* Portabilidad.

Anotar la hora de fin (indique hh:mm:ss): _____

2) Intente comprender las tablas y valore según su criterio la COMPLEJIDAD del modelo de Medición del Software

Muy simple ☐ Algo simple ☐ Normal ☐ Algo complejo ☐ Muy complejo ☐

Figura 5-3. Ejemplo de ejercicio de modificabilidad con un Diagrama SMML (cont.)

Cada ejercicio de modificabilidad incluía cuatro nuevos requisitos que suponían realizar modificaciones sobre el modelo propuesto. Al final se incluía también una parte de valoración subjetiva del nivel de dificultad del modelo. En la Figura 5-4 y Figura 5-5 se muestra un ejemplo de ejercicio de modificabilidad utilizando una notación Textual.

Elementos del metamodelo de medición (clases MOF)	Modelo de medición (instancia M2)	Relaciones del metamodelo de medición (asociaciones MOF)		Modelo medición (links M2)
Atributo	Tamaño Complejidad	Tiene	Categoría de entidad	Esquema relacional
			Atributo	Tamaño Complejidad
Categoría de entidad	Esquema Relacional	Se define para	Modelo de calidad	ISO 9126
			Categoría de entidad	Esquema Relacional
Modelo de Calidad	ISO 9126	Evalúa	Modelo de Calidad	ISO 9126
			Concepto Medible	Mantenibilidad
Concepto medible	Mantenibilidad	Relaciona	Concepto medible	Mantenibilidad
			Atributo	Tamaño, Complejidad
Necesidad de Información	Conocer la mantenibilidad de los Esquemas Relacionales	Se asocia con	Concepto Medible	Mantenibilidad
			Necesidad de Información	Conocer la mantenibilidad de los Esquemas relacionales
Medida	NFK, RFK, SCI	Se define para	Medida	NFK, RFK, SCI
			Atributo	Complejidad
Medida	NT, NA, TMI	Se define para	Medida	NT, NA, TMI
			Atributo	Tamaño
Indicador	TMI, SCI	Satisface	Indicador	TMI, SCI
			Necesidad de información	Conocer la mantenibilidad de los esquemas relacionales

Medida Base	Descripción	Método de medición	Escala	Unidad
NFK	Número de Claves Ajenas	Contar las claves ajenas del esquema	Ratio: enteros desde 0 a infinito	Columna
NT	Número de Tablas	Contar las tablas de un esquema	Ratio: enteros desde 0 a infinito	Tabla
NA	Número de Atributos	Contar los atributos del esquema	Ratio: enteros desde 0 a infinito	Columna

Medida Derivada	Descripción	Función de medición	Escala	Unidad
RFK	Ratio de claves ajenas	RFK = NFK/NA	Ratio: entre 0 y 1	Clave ajena por atributo

Indicator	Descripción	Modelo de análisis	Criterio de decisión	Escala	Unidad
SCI	Índice de Conectividad del Esquema	SCI = NFK/NT	Si SCI ≥ 2 → SCI = 'Muy alto' Si 1,5 ≤ SCI < 2 → SCI= 'Alta' Si 1 < SCI< 1,5→SCI='Medio' Si 0,5 ≤ SCI ≤ 1→SCI = 'Bajo' Si 0<SCI<0,5→SCI='Muy Bajo'	Ratio: entre 0 y 1	Clave ajena por Tabla
TMI	Índice de Mantenimiento de Tablas	TMI = NA/NT	Si TMI > 18→TMI= 'Muy Alto' Si 12 < TMI ≤ 18→TMI= 'Alto' Si 6<TMI ≤ 12 → TMI='Medio' Si 0 ≤ TMI ≤ 6 → TMI = 'Bajo'	Ratio: entre 0 y 1	Porcentaje

Figura 5-4. Ejemplo de ejercicio de modificabilidad con una notación Textual

EJERCICIO 1. Tareas a realizar:

Anotar la hora de inicio (indique hh:mm:ss): _____

1) **Conteste verdadero o falso las siguientes afirmaciones en base al diagrama SMML que se muestra al final del ejercicio:**

   a) Los *atributos* Tamaño, Complejidad y Longitud **están relacionados** con el *concepto medible* Mantenibilidad

      V ☐ F ☐

   b) La *medida* NA **está definida para** el *atributo* Tamaño

      V ☐ F ☐

   c) El *indicador* TMI **satisface** la *necesidad de información* Conocer la mantenibilidad de los esquemas relacionales

      V ☐ F ☐

   d) La medida SCI **tiene** la escala Ratio: entre 0 y 1 y la unidad Clave Ajena por Tabla

      V ☐ F ☐

   e) El *modelo de calidad* ISO9126 **evalúa** el *concepto medible* Usabilidad

      V ☐ F ☐

   f) La *medida derivada* RFK **se calcula con** la *función de medición* Contar las claves ajenas del esquema

      V ☐ F ☐

Anotar la hora de fin (indique hh:mm:ss): _____

2) **Intente comprender el diagrama SMML y valore según su criterio la COMPLEJIDAD del modelo de Medición del Software**

   Muy simple ☐ Algo simple ☐ Normal ☐ Algo complejo ☐ Muy complejo ☐

Figura 5-5. Ejemplo de ejercicio de modificabilidad con una notación Textual (cont.)

c) Cuestionario Cualitativo

Otro aspecto que se evaluó fue la percepción de los sujetos respecto a la idoneidad de cada símbolo gráfico del lenguaje. Para ello, se incluyó al final del material un cuestionario (véase Figura 5-6) en el que los sujetos seleccionaban en una escala del 1 (muy adecuado) al 5 (muy inadecuado) acorde a su percepción. Podían añadir además comentarios, que se tuvieron en cuenta para mejorar el lenguaje.

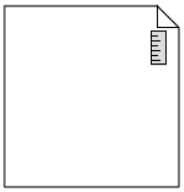
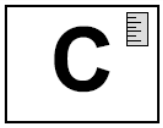
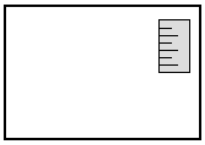
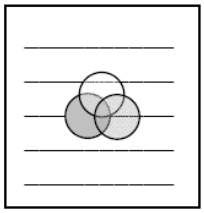
Elemento	Icono	Definición	Valoración	Comentarios
Description *(Descripción)*		Descripción del elemento de medición en concreto		
Information need *(Información de necesidad)*		Información necesaria para gestionar un proyecto (sus objetivos, hitos, riesgos y problemas).		
Measurable Concept *(Concepto medible)*		Relación abstracta entre *Attributes* y *Information Need*.		
Entity Class *(Categoría de entidad)*		Una colección de *entities* caracterizadas por satisfacer un cierto predicado común. Es el dominio.		
Attribute *(Atributo)*		Una propiedad mensurable, física o abstracta, que comparten todas las *Entity Class*.		
Quality Model *(Modelo de calidad)*		Un conjunto de *Measurable Concept* y relaciones entre ellos que proporciona la base para especificar requisitos de calidad y evaluar la calidad de *entity class*		

Figura 5-6. Extracto del cuestionario para la adecuada correspondencia de los elementos con los iconos

5.2.2.3. Variables.

De acuerdo al estándar ISO/IEC 9126, la usabilidad es una característica del software que se define como "la capacidad del producto software de ser entendido, aprendido, utilizado y ser atractivo al usuario, cuando se utiliza en las condiciones especificadas". La usabilidad se estructura en las siguientes sub-características: entendibilidad, facilidad de aprendizaje, operabilidad, atracción y conformidad. Siguiendo esta norma, otra característica software es la mantenibilidad, que se define como "la capacidad del producto software para ser modificado". Las modificaciones pueden incluir correcciones, mejoras o adaptación del software a cambios en el entorno, en los requisitos o en las especificaciones funcionales". Las sub-características de

la mantenibilidad son: analizabilidad, cambiabilidad (también llamada modificabilidad), estabilidad, facilidad de prueba y conformidad.

Las **variables dependientes** de nuestro estudio empírico corresponden a dos sub-características, una de la usabilidad y otra de la mantenibilidad:

- **Entendibilidad**, que se midió por medio de:
  - **Tiempo:** tiempo empleado por los sujetos en responder las seis preguntas.
  - **Acierto:** número de respuestas correctas.
  - **Eficiencia:** relación del número de respuestas correctas por el tiempo empleado en responderla (tiempo / acierto).
  - **Valoración:** valoración subjetiva del nivel de dificultad de los modelos.

- **Modificabilidad,** que se midió por medio de:
  - **Tiempo:** tiempo empleado por los sujetos en realizar las 4 modificaciones de los nuevos requisitos.
  - **Acierto:** número de modificaciones correctas.
  - **Eficiencia:** relación del número de modificaciones correctas por el tiempo empleado en realizarlo (tiempo / acierto).
  - **Valoración:** valoración subjetiva del nivel de dificultad de los modelos.

La principal **variable independiente** fue el uso o no de SMML (**notación**) para representar los modelos de medición del software. Otra variable independiente fue el **UoD**.

5.2.2.4. Hipótesis.

El experimento se planificó con el propósito de probar las hipótesis presentadas en la Tabla 5-9 en la cual, para cada conjunto de hipótesis, se detallan las variables dependientes e independientes, las medidas de las variables dependientes y la formulación de hipótesis nula y alternativa.

ID	Variable Dependiente	Medidas de Variables Dependientes	Variable Independiente	Hipótesis Nula - H_0	Hipótesis Alternativa - H_1
SU	Entendibilidad	Tiempo	Notación	H_{0Su}	H_{1Su}
		Acierto		El uso de SMML no mejora la Entendibilidad de los modelos de medición del software	El uso de SMML mejora la Entendibilidad de los modelos de medición del software
		Eficiencia			
		Valoración			
SM	Modificabilidad	Tiempo	Notación	H_{0Sm}	H_{1Sm}
		Acierto		El uso de SMML no mejora la Modificabilidad de los modelos de medición del software	El uso de SMML mejora la Modificabilidad de los modelos de medición del software
		Eficiencia			
		Valoración			
UoD	Entendibilidad	Tiempo	UoD	H_{0UoD}	H_{1UoD}
	Modificabilidad	Acierto		No hay diferencias en Entendibilidad o en Modificabilidad debido al Universo de Discurso del modelo	Hay diferencias en Entendibilidad o en Modificabilidad debido al Universo de Discurso del modelo
		Eficiencia			
		Valoración			
SxUoD	Entendibilidad	Tiempo	Notación UoD	H_{0SxUoD}	H_{1SxUoD}
	Modificabilidad	Acierto		No hay diferencias en Entendibilidad o en Modificabilidad como resultado de combinar el uso de SMML y el Universo de Discurso del modelo	Hay diferencias en Entendibilidad o en Modificabilidad como resultado de combinar el uso de SMML y el Universo de Discurso del modelo
		Eficiencia			
		Valoración			

Tabla 5-9. Conjunto de hipótesis

5.2.2.5. Diseño, Tareas Experimentales y Tratamiento.

De acuerdo con la naturaleza del experimento se aplicó el diseño experimental que muestra la Tabla 5-10. Los sujetos se dividieron aleatoriamente en dos grupos: Grupo A, cuyos miembros recibieron el material A, y el Grupo B, a quienes se repartió el material B. Como se puede observar en dicha tabla, el propósito era que los sujetos trabajaran con un UoD en un ejercicio de entendibilidad utilizando SMML, y el mismo UoD se aplicara en un ejercicio de modificabilidad con notación TEXTUAL. Por cada sujeto se requería que contestara los cuestionarios, llevara a cabo las modificaciones de cada modelo y valorase la dificultad de cada modelo según su opinión. Por último, cada sujeto tenía que responder a un cuestionario cualitativo relacionado con la usabilidad de SMML. La Tabla 5-11 muestra los modelos específicos que recibió cada sujeto según el grupo al que pertenecía y el tipo de ejercicio.

		Universo de Discurso			
Tipo de ejercicio	**Notación**	**Esquema Relacional**	**Diagrama de Clases UML**	**Portal Web**	**Diagram E/R**
Entendibilidad	**SMML**	Grupo A	Grupo A	Grupo B	Grupo B
	TEXTUAL	Grupo B	Grupo B	Grupo A	Grupo A
Modificabilidad	**SMML**	Grupo B	Grupo B	Grupo A	Grupo A
	TEXTUAL	Grupo A	Grupo A	Grupo B	Grupo B

Tabla 5-10. Diseño del Experimento

Para desarrollar los cuestionarios de entendibilidad y los requisitos de modificabilidad, se siguió la misma plantilla de ejercicio, tanto para los modelos representados con SMML como para los representados con notación textual (Tabla 5-11). Como consecuencia, la complejidad de las asignaciones estaba compensada tanto en las representaciones gráficas como textuales.

Grupo	Tipo de ejercicio	Modelos
Grupo A	Entendibilidad (6 preguntas)	1 2 3 4
Grupo A	Modificabilidad (4 requisitos)	5 6 7 8
Grupo B	Entendibilidad (6 preguntas)	5 6 7 8
Grupo B	Modificabilidad (4 requisitos)	1 2 3 4

Tabla 5-11. Material específico para cada sujeto clasificado en grupos y tipos de ejercicio

5.2.3. Operación.

Se comprobó varias veces el material para evitar que tuviera errores. Además, un ingeniero de software con conocimientos similares a los sujetos participantes llevó a cabo una experiencia piloto con 10 modelos. Esto permitió comprobar que el tiempo previsto era apropiado y que el material de aprendizaje, los modelos experimentales y las tareas a realizar se entendían correctamente. Probablemente gracias a ello, no fue necesario hacer ningún retoque significativo al diseño una vez obtenidos los resultados de la prueba piloto.

La ejecución del experimento consistió en repartir el material a los sujetos, a los cuales se les puso una fecha límite para devolver el material. Por tanto, el experimento no estaba directamente supervisado por los experimentadores. No se consideró necesaria la supervisión ya que los sujetos tenían experiencia en este tipo de experimento y se comprometieron a colaborar con la investigación siguiendo estrictamente las instrucciones que recibieron.

5.2.4. Análisis e Interpretación.

En este apartado se describe la recopilación de datos y un posterior análisis e interpretación de los resultados.

5.2.4.1. Estadística Descriptiva.

Una vez recogidos y revisados todos los cuestionarios rellenos por los sujetos, se consideró que todos ellos eran válidos. Cuando se obtuvieron los resultados, en primer lugar se realizó un análisis descriptivo de los datos. La Tabla 5-12 y Tabla 5-13 muestran la principal estadística descriptiva para las medidas de entendibilidad y modificabilidad, respectivamente. Puesto que la medida de valoración pertenece a una escala ordinal, se calculó la mediana para esta medida. En la Figura 5-7 y Figura 5-8 se muestra la representación gráfica en forma de diagrama de cajas (box plot) de los datos descriptivos para la eficiencia y tiempo respectivamente (véase Anexo B para ver el resto de diagramas de cajas para las medidas de las variables dependientes).

Medidas de Entendibilidad (Variables Dependientes)	SMML		TEXTUAL	
	Media	Desviación	Media	Desviación
Tiempo (m, s)	3m 4s	1m 35s	3m 16s	1m 23 s
Acierto (%)	93,33	9,091	88,33	16,537
Eficiencia	0,639	0,32	0,535	0,24
Valoración (Mediana)	Mediana=2	0,868	Mediana=3	0,992

Nota: Valoración: 1=Muy simple, 2=algo simple, 3=normal, 4=algo complejo, 5 muy complejo

Tabla 5-12. Resultados de entendibilidad: estadística descriptiva

Medidas de Modificabilidad (Variables Dependientes)	SMML		TEXTUAL	
	Media	Desviación	Media	Desviación
Tiempo (m, s)	4m 5 s	1m 36s	5m 35s	2m 44s
Acierto (%)	0,482	0,230	0,286	0,200
Eficiencia	99,37	3,953	67,50	22,072
Valoración (Mediana)	Mediana=2	0,944	Mediana= 4	0,987

Nota: Valoración: 1=Muy simple, 2=algo simple, 3=normal, 4=algo complejo, 5 muy complejo

Tabla 5-13. Resultados de modificabilidad: estadística descriptiva

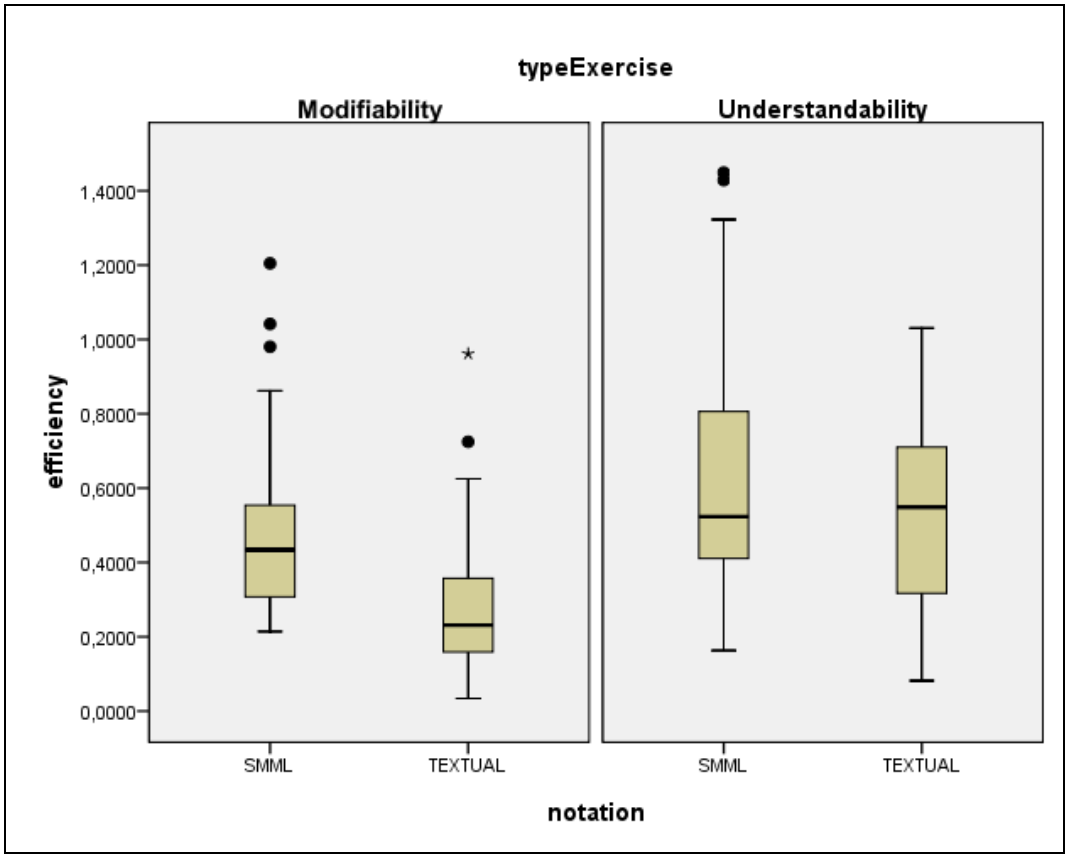


Figura 5-7. Diagrama de cajas de la eficiencia de la modificabilidad y entendibilidad.

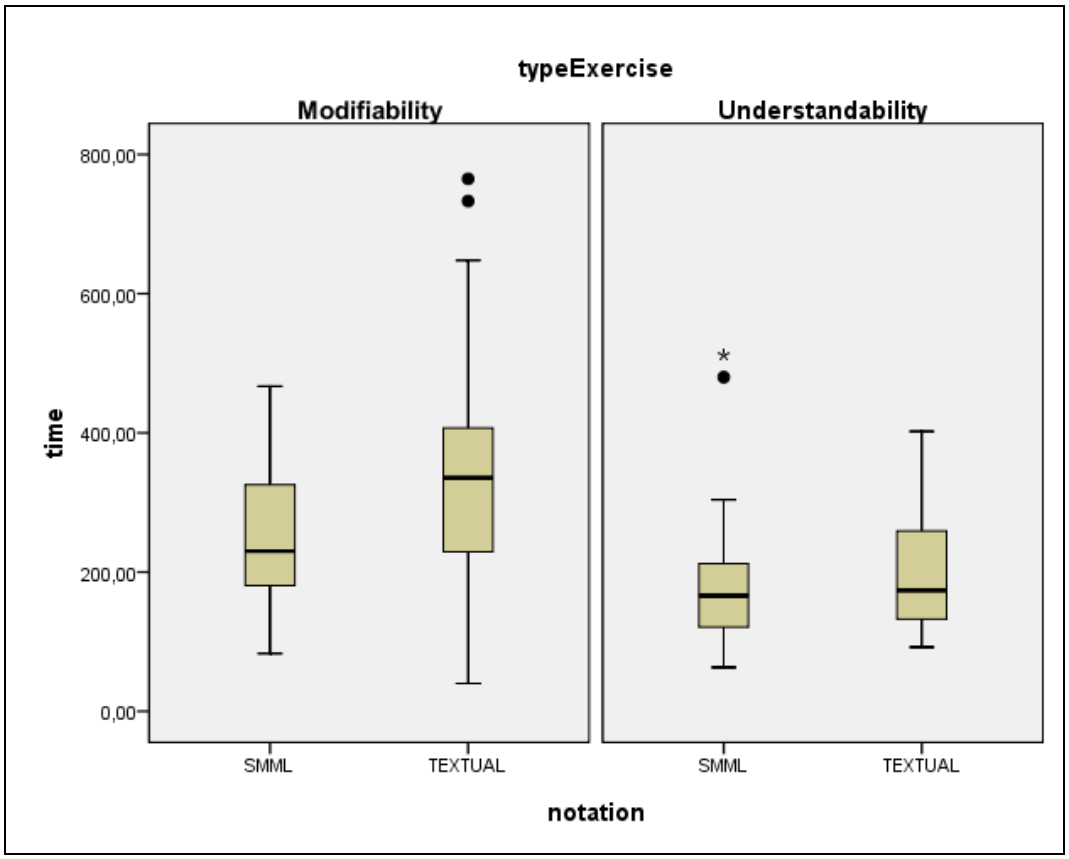


Figura 5-8. Diagrama de cajas del tiempo de la modificabilidad y entendibilidad.

Los resultados mostraron que el **tiempo** que los sujetos emplearon en entender y modificar las cuestiones de los modelos que utilizaban SMML fue menor que los resultados para los modelos con notación textual. La media de trabajo que los sujetos emplearon en los ejercicios de entendibilidad fue de 12,525 segundos menos (6,7% mejor) en los ejercicios que utilizaban SMML frente a los que no lo utilizaban, con respecto a los ejercicios de modificabilidad emplearon 89,77 segundos (1m 29s) menos en los ejercicios que utilizaban SMML frente a los que utilizaban la notación textual (36,5 % mejor). Además, el **acierto** de las respuestas de entendibilidad fueron de un 5,66% mejor cuando se utilizó SMML, y la acierto de las respuestas en modificabilidad fueron de un 47,21% mejor si se utilizaba SMML en los modelos. Por consiguiente, con estos resultados se puede afirmar que el uso de SMML mejoró la **eficiencia**, la diferencia de eficiencia en entendibilidad fue de una media de 0,104 mejor (19,4%) a favor de SMML, y en la modificabilidad fue de 0,196 (y una mejora del 68,5%). Por último, la **valoración subjetiva** de la dificultad de los diagramas que se definieron con SMML fue más entendible que las valoraciones de los modelos en los que no se utilizó SMML.

5.2.4.2. Contraste de Hipótesis e Interpretación Gráfica.

Una vez que se analizaron los datos descriptivos, el siguiente paso fue saber si las diferencias eran estadísticamente significativas o no. Para ello se realizó un test estadístico ANOVA con nivel de significancia $\alpha = 0,05$. Los resultados estadísticos para validar las hipótesis en relación a la medida de entendibilidad y modificabilidad de las variables dependientes se resumen en la Tabla 5-14.

Variable Dependiente	Tipo de Ejercicio	Notación	UoD	notación*UoD	Grupo
Time	Entendibilidad	0,539	0,366	0,830	0,254
	Modificabilidad	**0,002**	**0,000**	0,423	0,212
Eficiencia	Entendibilidad	0,110	0,100	0,987	0,338
	Modificabilidad	**0,000**	**0,000**	0,747	0,324
Acierto	Entendibilidad	0,093	0,161	0,390	0,158
	Modificabilidad	**0,000**	0,26	0,051	0,202
Valoración	Entendibilidad	**0,01**	0,235	0,348	0,236
	Modificabilidad	**0,02**	**0,000**	0,096	0,110

Tabla 5-14. Resultados ANOVA (niveles de significación)

Como muestra la Tabla 5-14, no se obtuvieron resultados significativos para explicar las diferencias en los valores de **entendibilidad**, excepto en la variable subjetiva de valoración ($p=0,01 < \alpha$). Por tanto se puede rechazar la hipótesis nula H_{0SU} y se puede afirmar que los sujetos, en entendibilidad, consideraron el uso de SMML más fácil que la notación textual.

Con respecto a la **modificabilidad** se obtuvieron las siguientes evidencias:

- Los modelos que no utilizaban SMML fueron más difíciles de modificar que las versiones con SMML. Se encontraron diferencias significativas con respecto al tiempo ($p=0,02 < \alpha$), aciertos ($p=0,00 < \alpha$), eficiencia ($p = 0,000 < \alpha$) y valoración subjetiva ($p=0,02 < \alpha$). Por tanto, la hipótesis H_0S_m pudo rechazarse.
- El UoD tuvo también una influencia en el tiempo, eficiencia y valoración subjetiva de la modificabilidad, es decir, los sujetos obtuvieron resultados y opiniones diferentes cuando trabajaban con diferentes UoDs ($p = 0,000 < \alpha$). Por tanto H_{0UoD} se pudo rechazar.
- El uso combinado de la notación y el UoD no tuvo influencia en los resultados obtenidos.

Por otro lado, como se observa en la columna "Grupo" de la Tabla 5-14, no hubo diferencias significativas en ninguna medida de las variables dependientes entre los sujetos de los ambos grupos. Esto significa que el grupo al que pertenecían los sujetos no se vio afectado en la facilidad de entender o modificar los modelos de medición del software. Por tanto, se puede concluir que el nivel de dificultad era equitativo para los dos grupos.

Además del análisis numérico, se realizó un análisis gráfico por medio de un gráfico de perfil (*profile plots*), mostrando las interacciones entre las variables del estudio. Concretamente, se obtuvieron diagramas para comprobar si había interacción entre el UoD (cuerpo del gráfico) y el uso o no de SMML (eje X) con respecto al tiempo/eficiencia/valoración/acierto de la entendibilidad y modificabilidad (eje Y) respectivamente.

La Figura 5-7, la Figura 5-8 y la Figura 5-9 muestran que el tiempo y la eficiencia empleada en llevar a cabo los ejercicios de **modificabilidad** son mejores cuando se utiliza el lenguaje SMML, independientemente del UoD que se utilice. Estos gráficos demuestran, otra vez, que el uso de SMML produce mejores resultados, independientemente de cual sea el UoD.

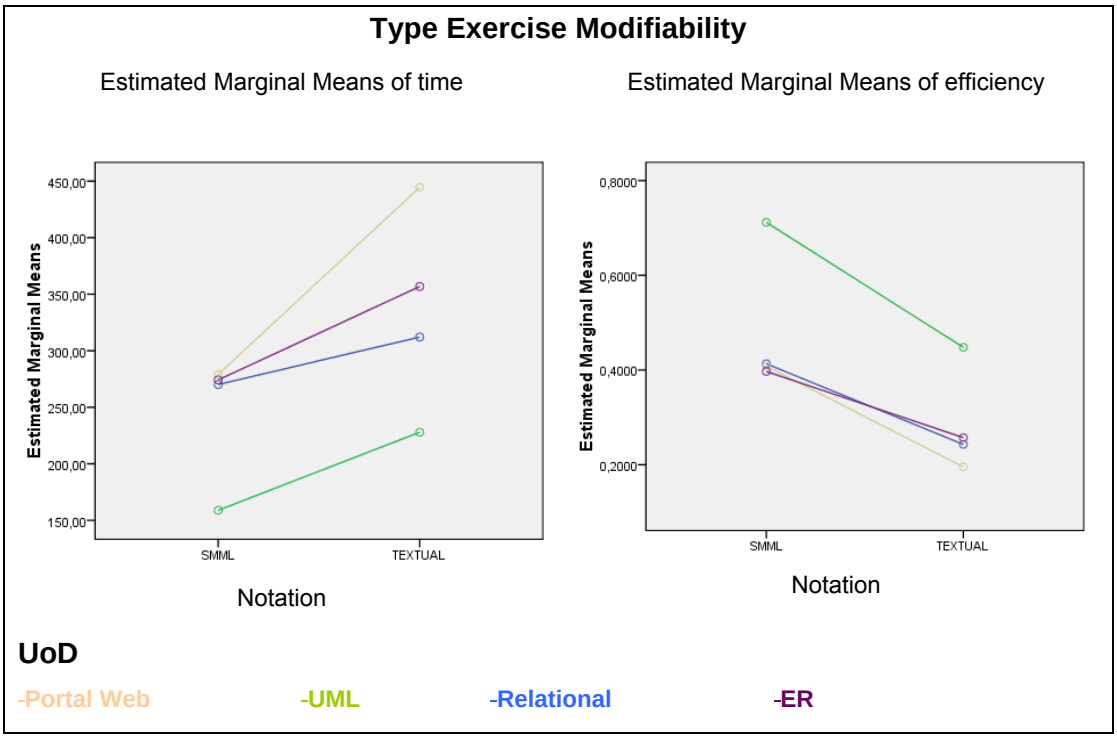


Figura 5-9. Diagrama de líneas de la interacción del UoD x uso de SMML en el experimento para el tiempo y eficiencia de la modificabilidad

De los gráficos también se puede deducir que los resultados de la eficiencia eran diferentes para cada UoD (líneas no paralelas), esto se debe a que el dominio de aplicación pudo afectar a la facilidad de modificación.

Como se ha comentado antes, con respecto a la **entendibilidad** no se encontraron diferencias estadísticamente significativas. A pesar de ello, la Figura 5-10 muestra que la eficiencia en entendibilidad es mejor cuando se utiliza el lenguaje SMML, independientemente del UoD que se aplique. Este gráfico muestra que no hay interacción entre el uso o no de SMML y el UoD.

Las líneas paralelas de la Figura 5-10 indican que no hubo interacción entre el uso o no de SMML y el UoD, es decir el UoD no afectó a la entendibilidad de los modelos. Por otro lado, aunque sólo hubo una evidencia estadística significativa en la entendibilidad, los gráficos nos demuestran que el uso de SMML produce mejores resultados en la modificación independientemente de cuál sea el UoD.

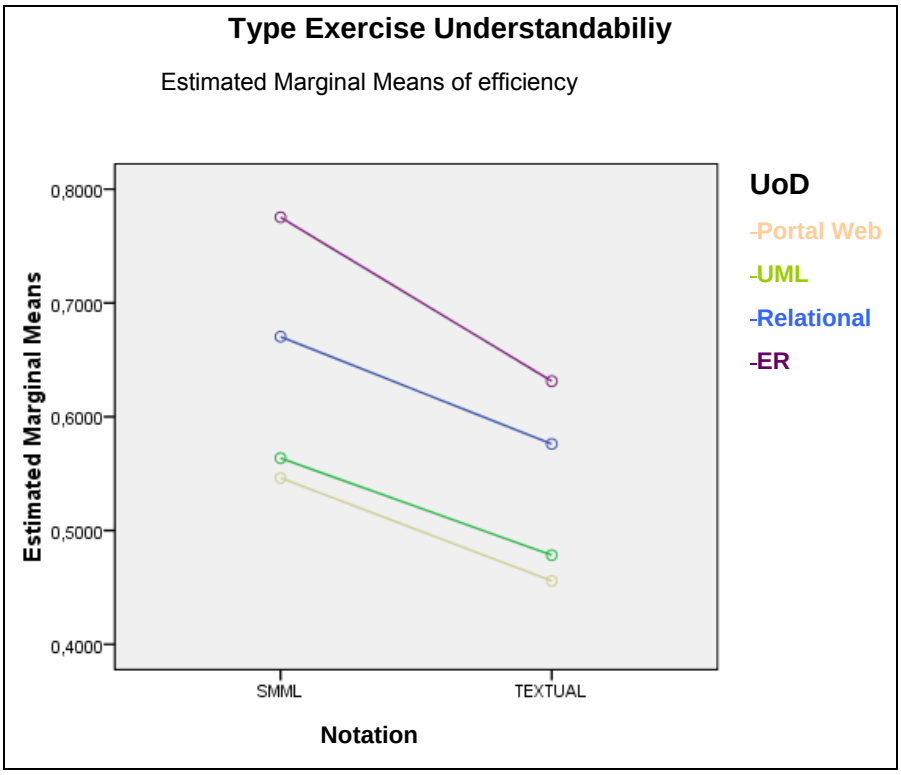


Figura 5-10. Gráfico de perfil de la interacción del UoD x uso de SMML en el experimento para la eficiencia de la entendibilidad

5.2.4.3. Valoración de los iconos.

Por último, se realizó un análisis descriptivo de los datos obtenidos en los cuestionarios cualitativos con el objetivo de detectar alguna mejora en la colección de iconos gráficos de SMML. La Tabla 5-15 y la Tabla 5-16 muestran, en orden ascendente, los resultados (mediana) asignados por los sujetos a los iconos de cada entidad y asociación de SMML.

Elemento	Mediana	Elemento	Mediana	Elemento	Mediana
Necesidad de Información	1	Descripción	2	Función de Cálculo	2
Criterio de Decisión	1	Unidad de Medición	2	Método de medición	2
Medida base	1	Indicador	2	Modelo de Calidad	2
Atributo	2	Modelo de Análisis	2	Escala	2
Medida Derivada	2	Concepto Medible	2	Entidad	3

Nota: Mediana (1=muy adecuado, 2=algo adecuado, 3=normal, 4=algo inadecuado, 5=muy inadecuado)

Tabla 5-15. Valoración de las entidades del lenguaje

Asociación	Mediana	Asociación	Mediana
Concepto Medible **(Relaciona)** Atributo	1	Indicador(**Satisface**) Necesidad de Información	2
Concepto Medible (**Incluye**) Concepto Medible	1	Medida Base (**Utiliza**) Método de Medición	2
Medida Derivada (**Calculada con**) Función de Cálculo	1	Concepto Medible (**se asocia con**) Necesidad de Información	2
Entidad (**Incluye**) Entidad	1	Modelo de Calidad (**Evalúa**) Concepto Medible	2
Indicador (**Calculada con**) Modelo de Análisis	1	Medida (**Definido para**) Atributo	2
Entidad (**Has**) Atributo	2	Modelo de Calidad (**Definido para**) Entidad	2
Modelo de Análisis (**Utiliza**) Criterio de Decisión	2		

Nota: Mediana (1=muy adecuado, 2=algo adecuado, 3=normal, 4=algo inadecuado, 5=muy inadecuado)

Tabla 5-16. Valoración de las asociaciones del lenguaje

Como muestra la Tabla 5-15, los iconos se consideraron algo adecuados para once entidades, muy adecuados para tres, y normal para una. La Tabla 5-16 muestra que la interpretación gráfica para las asociaciones se consideró algo adecuada en 8 casos, y muy adecuada en 5. En general, estos resultados fueron positivos y se interpretó que no había que hacer ningún cambio significativo en los símbolos gráficos.

5.2.5. Evaluación de la Validez.

Varios aspectos podían amenazar al proceso de experimentación y, por ello, tuvieron que analizarse durante la planificación de experimento:

- **Validez de la conclusión.** Algo que pudo afectar a la validez de la conclusión de este estudio es el tamaño de los datos del estudio (160 valores, 8 modelos por sujeto x 20 sujetos). Somos conscientes de este hecho y, por eso, se considera conveniente realizar una réplica con una muestra mayor.
- **Validez de constructo[17].** Con respecto a este aspecto, las medidas de las variables dependientes (tiempo, acierto, eficiencia) se utilizan normalmente en estudios en los que se llevan a cabo tareas cognitivas (Eysenck y Keane, 2005) que son de la misma naturaleza que en el estudio presentado en este capítulo. Normalmente, estas variables dependientes son las que se evalúan en un estudio experimental (Genero et al., 2005a; Patig, 2008) en Ingeniería del Software empírica. Puesto que estas medidas se obtuvieron a partir de la recolección datos de unos cuestionarios rellenados por los sujetos, las mediciones podían haber carecido de precisión, es decir, podía pasar que los sujetos no indicaran correctamente el tiempo en el que se terminó la tarea. Para solucionar esto, los sujetos recibieron instrucciones detalladas sobre cómo utilizar y rellenar los cuestionarios.
- **Validez Interna.** Los aspectos que podían haber amenazado a la validad interna se abordaron de la siguiente manera.

[17] Constructo: construcción teórica para resolver un problema científico determinado

- **Efectos de persistencia.** El experimento se llevó a cabo con sujetos que nunca antes habían hecho un experimento similar, evitando así efectos de persistencia.
- **Conocimiento del Universo de Discurso:** el conocimiento del dominio no afectó a la validez, ya que los modelos de medición se aplicaron en diferentes UoDs (ER, relacional, UML y Web), pero que eran familiares a los sujetos (ingenieros de software) para evitar efectos secundarios debidos a la curva de aprendizaje.
- **Efectos de Fatiga.** Para evitarlos la media de duración del experimento fue de una hora y media, incluyendo la lectura de la documentación de aprendizaje.
- **Motivación de los Sujetos.** Los sujetos se comprometieron a hacer este estudio y se les explicó los beneficios posibles de los resultados.
- **Plagio e Influencia entre los sujetos.** Esto no se controló, pero creemos que este aspecto no se dio debido al perfil del sujeto y su compromiso con el estudio.

- **Validez externa.** El experimento y las tareas se diseñaron teniendo en cuenta las restricciones de tiempo y el hecho de que los sujetos pertenecían al ámbito académico y el área de investigación. Se considerará una réplica futura de este experimento con modelos representativos de proyectos y con sujetos que formen parte de una empresa.

5.2.6. Presentación y Empaquetamiento.

Una vez que se realizó el experimento se presentaron los resultados y conclusiones en un artículo de revista (Mora et al., 2010a). El estudio completo se puede obtener en http://alarcos.esi.uclm.es/smf/smml.

5.3. Conclusiones.

En este capítulo se ha presentado un lenguaje para la definición de modelos de medición software. El conjunto de iconos que forman parte del lenguaje se seleccionaron para que resultaran 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.

SMML es un lenguaje con una definición sintáctica y semántica clara y una base ontológica sólida. Además, el lenguaje se ha desarrollado con el objetivo de cumplir con los requisitos de un DSL:

- Usabilidad: más usable que una notación textual.
- Conforme al metamodelo de medición del software.
- Ortogonal, ya que cada constructor del lenguaje se utiliza para representar exactamente un concepto distinto en el dominio.
- Con el soporte de SMML Editor. Para dar soporte al uso del lenguaje se ha desarrollado un editor gráfico basado en GMF y que se presenta en el Capítulo 6.
- Simple, SMML se ha desarrollado con el objetivo de ser lo más simple posible para expresar los conceptos del dominio y dar soporte a sus usuarios.

Este lenguaje juega un papel fundamental en SMF, ya que permite definir los modelos de medición del software que son entrada en el proceso de medición, 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.

También se ha estudiado la usabilidad y mantenibilidad (basada en ISO/IEC 9126) del lenguaje SMML mediante un estudio empírico. El estudio se llevó a cabo gracias a la colaboración de expertos ingenieros de Software cuyo trabajo consistió en comprender y modificar modelos de medición del software con dos notaciones, gráfica en SMML y textual, en forma de tablas. Las principales conclusiones obtenidas del experimento fueron las siguientes:

- El uso de SMML mejora la modificabilidad de los modelos de medición del software. Con respecto a la entendibilidad se obtuvieron mejores resultados con el uso de SMML, aunque no con una diferencia estadísticamente significativa.
- El uso de SMML se considera más fácil (valoración subjetiva) que el uso de la notación textual, tanto en entendibilidad como en modificabilidad.
- Los iconos de las entidades y sus relaciones se consideraron adecuados.

"El genio se compone del dos por ciento de talento y del noventa y ocho por ciento de perseverante aplicación" (Ludwig van Beethoven)

6. Entorno Tecnológico.

En este capítulo se describe el entorno tecnológico del marco de trabajo SMF. Dicho entorno está constituido por SMTool, una herramienta desarrollada para evaluar de manera pragmática la propuesta del marco conceptual SMF (véase Capítulo 4) con el fin de facilitar la medición de cualquier entidad software.

En primer lugar se describen las características generales y arquitectura software de la herramienta, para después pasar a presentar cada uno de sus componentes principales.

6.1. Características Generales.

Para poder aprovechar el marco conceptual SMF en la práctica es necesario disponer de herramientas adecuadas. Teniendo en cuenta los objetivos de SMF, los requisitos que se deberían satisfacer son los siguientes:

- Soporte al **Modelado de la Medición**. Facilitar la definición de modelos de medición software, basados en SMM y expresados mediante el lenguaje SMML presentado en el Capítulo 5.
- Soporte a la **Medición Genérica**. Facilitar la medición genérica del software, es decir, que implemente métodos de medición genéricos (un concepto mucho más abstracto y, por tanto, más genérico, que el de medida/métrica). Para ello la mejor opción es aplicar el paradigma MDE, trabajando a nivel de metamodelo en vez de a nivel de modelo.
- Debe ser un **Entorno Flexible**. Dado que la medición del software es una tarea compleja y dinámica, el entorno tecnológico debe ser abierto y extensible para que fácilmente se puedan incorporar nuevas herramientas de soporte.

SMTool (Software Measurement Tool)[18] ha sido desarrollada teniendo en cuenta estos requisitos, con el objetivo inicial de ayudar a los usuarios a llevar a cabo mediciones de cualquier tipo de artefacto software. Para conseguir este objetivo, SMTool maneja y representa el conocimiento relacionado con la medición de cualquier tipo de entidad[19] de una manera integrada y consistente. SMTool es una herramienta que se instala como un plug-in de la plataforma Eclipse.

6.2. Componentes y Arquitectura de SMTool.

Para cumplir los requisitos expuestos en el apartado anterior, SMTool está compuesto por los siguientes elementos (véase Figura 6-1):

[18] Disponible en http://alarcos.inf-cr.uclm.es/smf/smtool

[19] A nivel de implementación un tipo de entidad se corresponde a nivel de implementación con una meta-clase de ECORE; y una entidad se corresponde con una instancia de una meta-clase de ECORE.

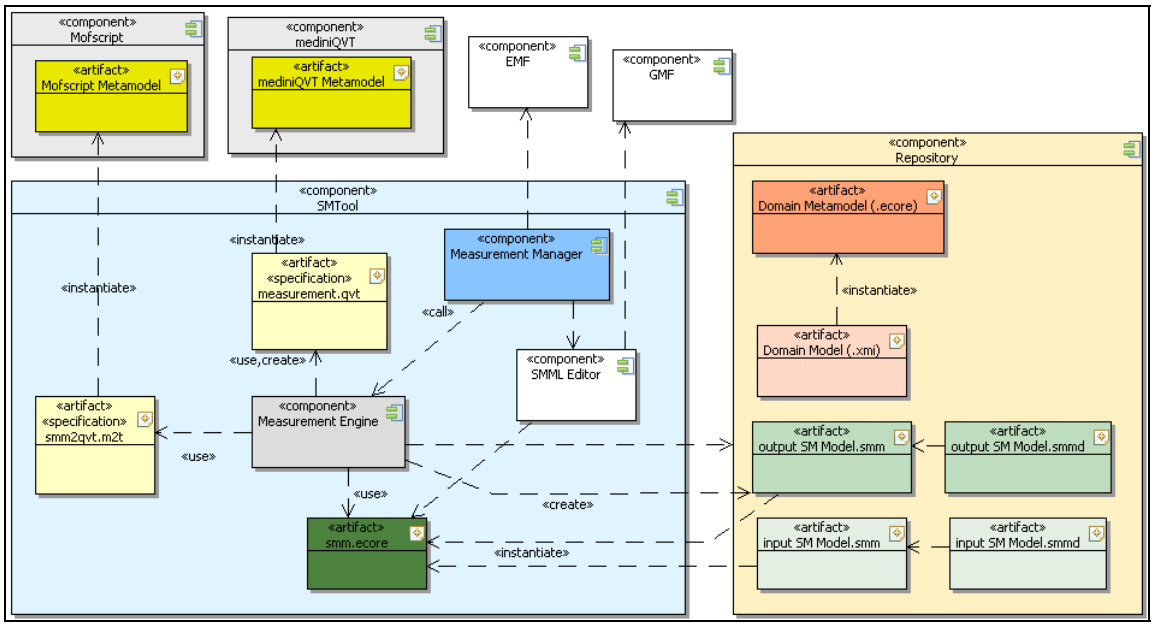


Figura 6-1. Diagrama de componentes UML de SMTool

- **smm.ecore.** Es el Metamodelo de Medición de Software adaptado y definido en ECORE que se utiliza para definir modelos de medición. Este modelo está integrado en SMTool y por tanto es transparente al usuario, que lo utiliza de forma indirecta por medio del editor gráfico SMML-Editor. Este elemento se presenta en mayor detalle en el apartado 6.4.3.

- **MOFScript.** Es una herramienta, basada en MOF, de transformación de modelo a texto para dar soporte, por ejemplo, a la generación de la implementación de código o documentación a partir de modelos. Proporciona un metamodelo independiente del lenguaje que permite utilizar cualquier tipo de metamodelo y sus instancias para generar representaciones textuales.

- **MediniQVT**[20]**.** Es una herramienta para llevar a cabo transformaciones de modelo a modelo por medio del estándar QVT Relations (OMG, 2002a), de la OMG.

- **smm2qvt.m2t.** Un fichero que contiene la especificación MOFScript necesaria para generar/obtener la especificación qvt (componente measurement.qvt) que ejecuta cada medición genérica del software; es decir, es el modelo necesario para ejecutar la *Transformación Inicial* de SMF (véase el apartado 6.5). No es necesario para el usuario saber de la existencia de este fichero.

- **measurement.qvt.** Este fichero contiene la especificación QVT para ejecutar la medición genérica de forma automática por medio de una transformación QVT; es decir, es el modelo necesario para ejecutar la *Transformación Final* o ejecución de la Medición de SMF (más información en el apartado 6.5). Este fichero se genera con la

[20] Disponible en http://projects.ikv.de/qvt/

transformación previa automática de MOFScript, y una vez generado el usuario puede consultarlo.

- **Measurement Engine.** Este componente ejecutable (descrito en mayor detalle en el apartado 6.5) gestiona e invoca la ejecución, en el orden y forma correcta, la transformación inicial y la transformación final llevadas a cabo en SMF.

- **SMML-Editor.** Editor gráfico desarrollado para facilitar la definición de los modelos de medición (véase apartado 6.4).

- **Repositorio.** El repositorio es una colección de ficheros XMI que incluye:
  - **Metamodelos de dominio.** Cada uno de los elementos necesarios para definir una medición en un dominio concreto. Estos ficheros están definidos a partir del meta-metamodelo ECORE.
  - **Modelos de dominio.** Son instancias de los metamodelos de dominio anteriores y contienen las instancias de los elementos a medir.

- **Modelos de Medición del Software.** Son ficheros smm definidos conforme a SMM. Se crean y editan con SMML-Editor y representan toda la información de medición de un proyecto. A cada modelo de medición le corresponde un diagrama SMML (fichero smmd que incluye los aspectos gráficos).

- **Measurement Manager.** Componente ejecutable que integra todas las opciones para guiar al usuario durante la realización de mediciones siguiendo el método indicado en SMF. Las acciones a las que da soporte son:

  a) Creación de un proyecto SMTool. Facilita la creación de un proyecto de SMTool generando la estructura de directorios apropiada.

  b) Permitir la definición del metamodelo y modelo de dominio. Definir o cargar un modelo de dominio y su correspondiente metamodelo. Para implementar esta funcionalidad se ha utilizado EMF[21].

  c) Llamada al componente Measurement Engine. Este componente se invoca con el propósito de realizar una medición automática. Antes de invocarlo es necesario seleccionar los elementos que son entrada a la ejecución de la medición, que son: el metamodelo de dominio, el modelo de dominio y el modelo de medición. El resto de elementos de entrada (*smm.ecore* y *smm2qvt.m2t*) se asignan automáticamente. Como resultado, se obtiene un nuevo modelo de medición (fichero en formato smm) extendido con los resultados de la medición. Para centralizar la selección de elementos de entrada y ejecución de la medición, se utiliza el componente *Run configurations* de Eclipse.

  Para una mayor usabilidad, existe un menú específico de medición que permite invocar las acciones del componente *MeasurementManager*.

[21] Disponible en http://www.eclipse.org/modeling/emf/

6.3. Arquitectura Conceptual del Repositorio.

El repositorio de SMTool es una colección de ficheros XMI. Conceptualmente, dichos ficheros están organizados según la arquitectura conceptual de SMF (véase Figura 6-2), definida en base a los niveles de abstracción del estándar MOF:

- **Nivel de Meta-Metamodelo (M3):** el meta-metamodelo que pertenece a este nivel en SMTool es ECORE.
- **Nivel de Metamodelo (M2):** con los siguientes metamodelos:
  - Smm.ecore
  - Metamodelos de dominio (.ecore), definidos a partir de ECORE.
  - Metamodelos de transformación: QVT Relations y MOFScript.
- **Nivel de Modelo (M1):** con modelos de varios tipos:
  - Modelos de medición iniciales y finales (extendidos con resultados) (.smm).
  - Modelos de dominio (.xmi) definidos en base al metamodelo correspondiente.
  - Modelos de Transformación: *measurement.qvt* y *smm2qvt.m2t*.

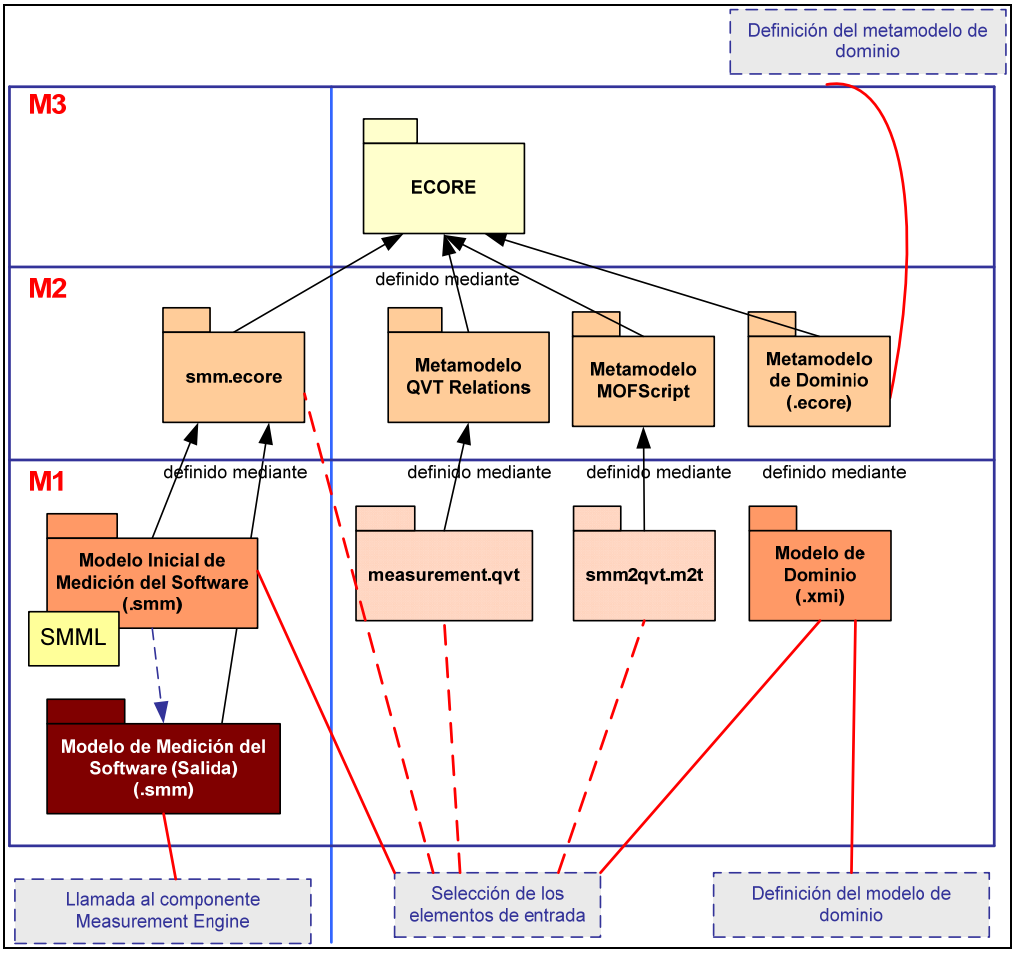


Figura 6-2. Arquitectura Conceptual del Repositorio de SMTool

La Figura 6-2 muestra cada uno de los tipos de elementos del repositorio de SMTool clasificados verticalmente en los niveles de abstracción de la arquitectura conceptual MOF.

Además, se indican en un recuadro azul las acciones invocadas por el *MeasurementManager* que gestionan dichos elementos, comentadas en el apartado anterior.

6.4. SMML-Editor.

Como se ha justificado en los capítulos 4 y 5, el lenguaje SMML tiene un rol importante en SMF ya que facilita definir modelos de medición que son entrada de la medición automática. La representación gráfica de los modelos de medición permite que SMF sea un marco de trabajo más usable e intuitivo para el usuario. En otras palabras, hace que el proceso de medición pueda ser más fácil.

SMML-Editor es el editor gráfico de modelos de medición, expresados en SMML, integrado en SMTool. Ha sido creado en base a SMM utilizando GMF de Eclipse. SMML Editor trabaja con dos tipos de ficheros, smm y smmd, que se describen a continuación.

6.4.1. Ficheros smm.

Estos ficheros implementan los modelos de medición de software conforme al metamodelo SMM, es decir, contienen instancias dinámicas de SMM expresadas en formato XMI. Para editarlos se puede utilizar el editor estándar *Sample Reflective Ecore Model Editor*, incluido en EMF (véase Figura 6-3 izquierda). Este editor con vista en árbol lista todos los elementos de medición (Entity Class, Attribute, Base Measure, etc. [22]). Además, se puede mostrar una vista de propiedades por cada elemento de medición (véase Figura 6-3 derecha), que visualiza y permite editar los valores (propiedades) de los atributos para cada elemento. Por ejemplo, para un elemento de tipo *Base Measure* muestra las siguientes propiedades: *Attributes* (lista de atributos), *Measurement Method*, *Name*, *Scale*, *Status*, *Unit* y *Value*. Todos estas propiedades del modelo de medición vienen establecidos en el metamodelo SMM.

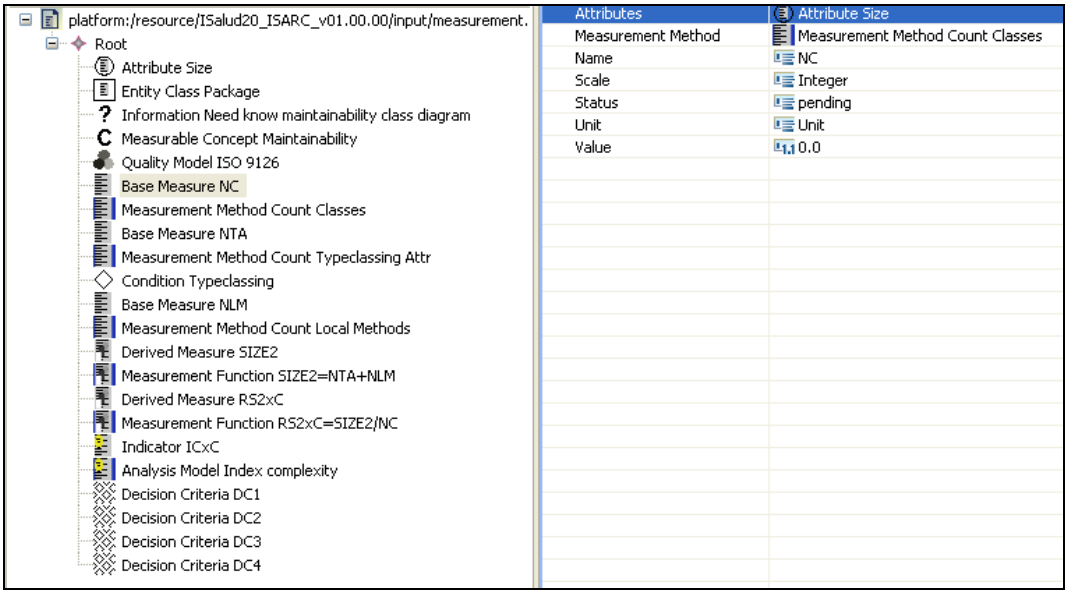


Figura 6-3. Fichero smm abierto con el clásico editor de árbol EMF (izq) y la vista de propiedades (der)

[22] Aunque en el Capítulo 4 SMML se presenta con los elementos en español, en este capítulo, al estar la herramienta implementada en inglés se han respetado los nombres de los elementos y atributos en inglés.

6.4.2. Ficheros smmd.

Los ficheros smmd contienen diagramas en notación SMML de los modelos de medición. Estos ficheros se crean con el SMML-Editor a partir de un fichero smm (un fichero smmd no puede existir sin su correspondiente fichero smm) y contienen la información gráfica adicional a los modelos de medición, es decir, la posición y los símbolos gráficos. Los elementos y asociaciones de medición se pueden introducir en el diagrama por medio de las opciones incluidas en la paleta de herramientas, que contiene todas las opciones necesarias para definir diagramas de medición de una manera gráfica.

Existen dos maneras de introducir los valores de los atributos de cada elemento de medición:

- Con la vista de propiedades, igual que la mostrada con los ficheros smm.
- Escribiendo directamente en el diagrama (véase Figura 6-4).

Con SMML-Editor el usuario puede ver el modelo de medición completo (véase Figura 6-3 y Figura 6-4), con todos los elementos y sus relaciones.

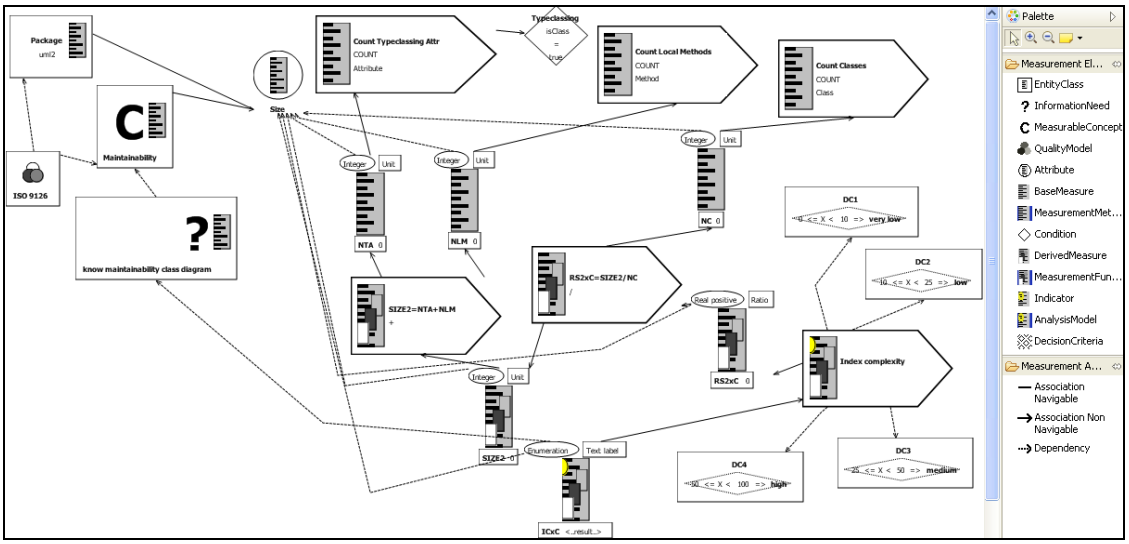


Figura 6-4. Modelo de medición abierto con el SMML-Editor

Una característica importante de SMML-Editor es la actualización sincronizada de los ficheros smmd y smm, de modo que un cambio en el fichero smm se refleja automáticamente en el diagrama smmd y viceversa. Otra característica importante es la posibilidad de validar los modelos con respecto a smm.ecore. De esta manera los usuarios pueden asegurar que los modelos de medición son *SMM-compliant*.

6.4.3. Adaptación de SMM.

El metamodelo smm.ecore es el metamodelo base de SMML-Editor. En este sentido, debe tenerse en cuenta que, para que los modelos definidos con SMML-Editor sean operables por el resto de SMTool, tienen que cumplir los siguientes requisitos:

- Estar basados en SMM.

- Permitir al componente ***Measurement Engine*** que opere sobre ellos para llevar a cabo las mediciones (transformaciones).
- Que el metamodelo que los define sea un modelo de definición de dominio válido para generar un editor mediante GMF.

Para cumplir estos requisitos ha sido necesario adaptar el metamodelo inicial SMM para añadirle los siguientes nuevos elementos (véase Figura 6-5):

- **Operable Measure.** Es una entidad abstracta que representa las medidas que son operables, es decir aquellas que utilizan las formas de medir (*Measurement Method*, *Measurement Function* y *Analysis Model*) para generar medidas derivadas. Las medidas operables son *Base Measure* y *Derived Measure*.
- **Condition.** Este elemento permite llevar a cabo formas de medir parametrizadas, es decir, ejecutar una forma de medir con una condición añadida. Actualmente este elemento sólo está implementado para los métodos de medición.
- **Root.** Para poder generar un editor gráfico GMF a partir de un metamodelo, es necesario tener un elemento raíz (o base) que contenga a todos los elementos del modelo. El elemento Root contiene todos los elementos de medición.
- **Atributo *status* de Measure.** Este nuevo atributo define el estado de una medida:
  - **Pending.** Cuando no ha sido calculada.
  - **Processed.** Cuando ha sido calculada y por tanto tiene un valor asociado.

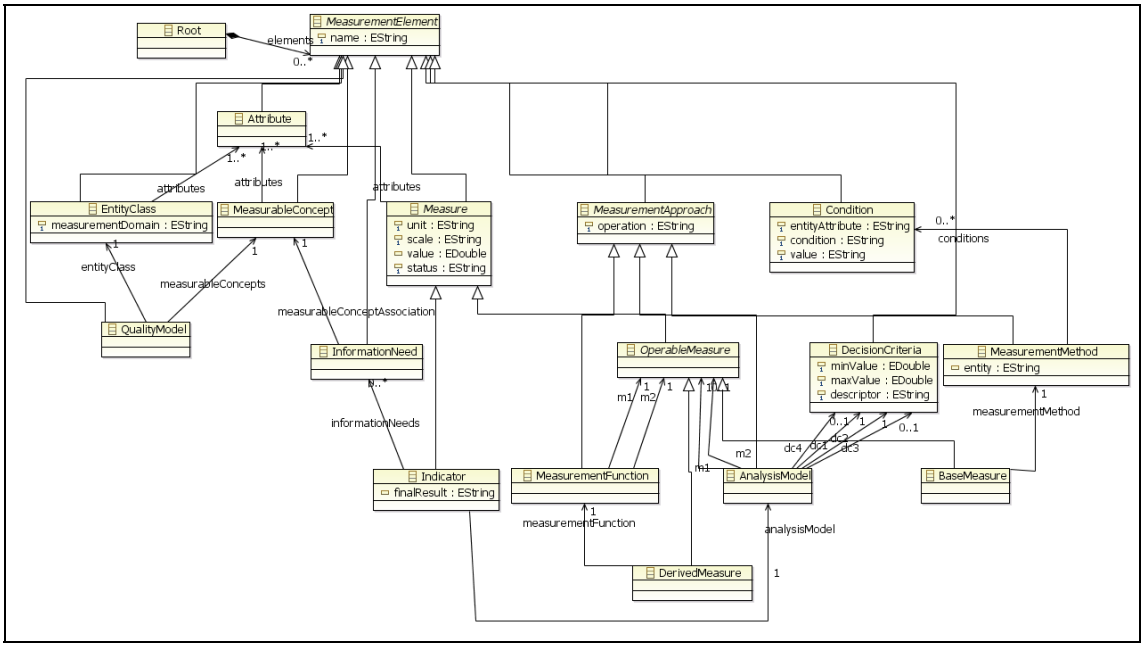


Figura 6-5. Metamodelo smm.ecore

Como se observa en la Figura 6-5, el metamodelo smm.ecore cumple con todos los requisitos expuestos.

Además de los requisitos mencionados, otro muy importante es que el editor gráfico generado implemente los elementos gráficos del lenguaje, en este caso, SMML. Para ello se ha seguido el proceso de desarrollo de un DSL de GMF presentado en la Tabla 6-1.

N	Etapa	Ficheros generados
1	**Definición del modelo de dominio**. Metamodelo que define el lenguaje del editor gráfico.	smm.ecore
2	**Definición gráfica.** Definición de las figuras de cada uno de los elementos del editor: entidades y relaciones.	smm.gmfgraph
3	**Definición de la Paleta.** Definición de las opciones de la paleta de herramientas para dibujar las entidades y las relaciones.	smm.gmftool
4	**Correspondencia con los elementos del diagrama.** Correspondencia entre los elementos del dominio y las figuras gráficas, así como la correspondencia con las opciones de la paleta que permiten dibujar dichos elementos. El fichero .gmfmap se obtiene a partir de los tres ficheros anteriores. Mediante .gmfmap se obtiene el fichero .gmfgen.	smm.gmfmap
5	**Generación del código.** A partir del .gmfgen se genera código java de forma automática.	smm.gmfgen

Tabla 6-1. Proceso de desarrollo de un editor gráfico utilizando GMF

6.5. Measurement Engine.

Como se ha introducido en el apartado 6.3, el componente Measurement Engine permite lleva a cabo automáticamente el proceso de medición a través de una transformación QVT. Como se ha explicado en el Capítulo 4, el proceso de medición llevado a cabo por el componente Measurement Engine está formado por las siguientes dos transformaciones:

- **Transformación Inicial.** Mediante el componente MOFScript y a partir de la especificación *smm2qvt.m2t* se obtiene el modelo QVT (*measurement.qvt*). Puesto que el modelo de medición está definido a partir de *smm.ecore*, para llevar a cabo la transformación de modelo a texto es necesario también el conocimiento del metamodelo que lo define (véase Figura 6-6).

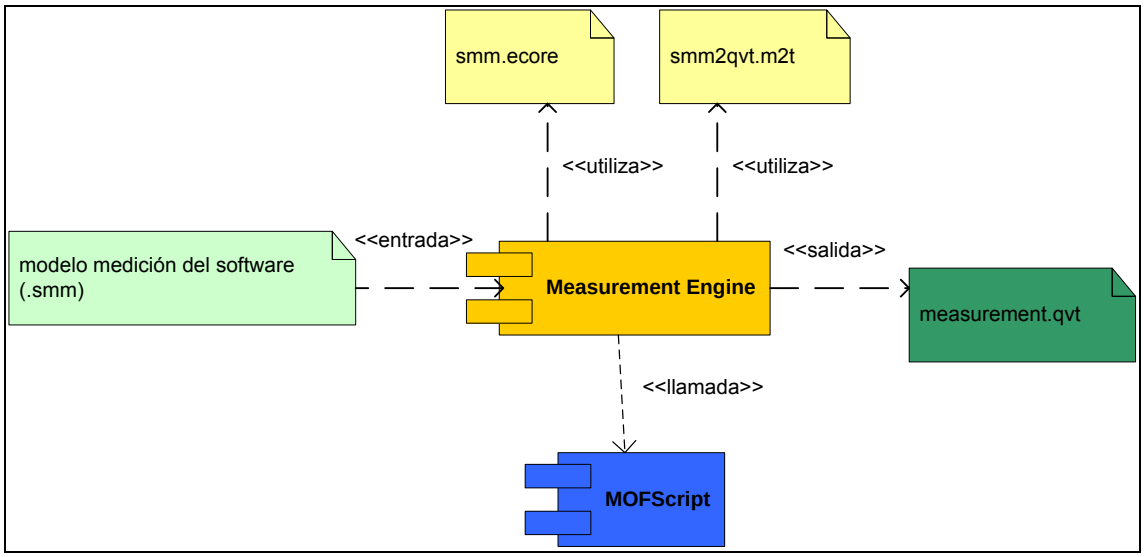


Figura 6-6. Transformación Inicial

- **Transformación Final o Ejecución de la Medición.** Una vez que se ha obtenido la especificación QVT (*measurement.qvt*), se puede ejecutar la transformación final a partir de los modelos de entrada (el modelo de dominio y el modelo de medición) y mediante la especificación de la transformación el modelo (véase Figura 6-7).

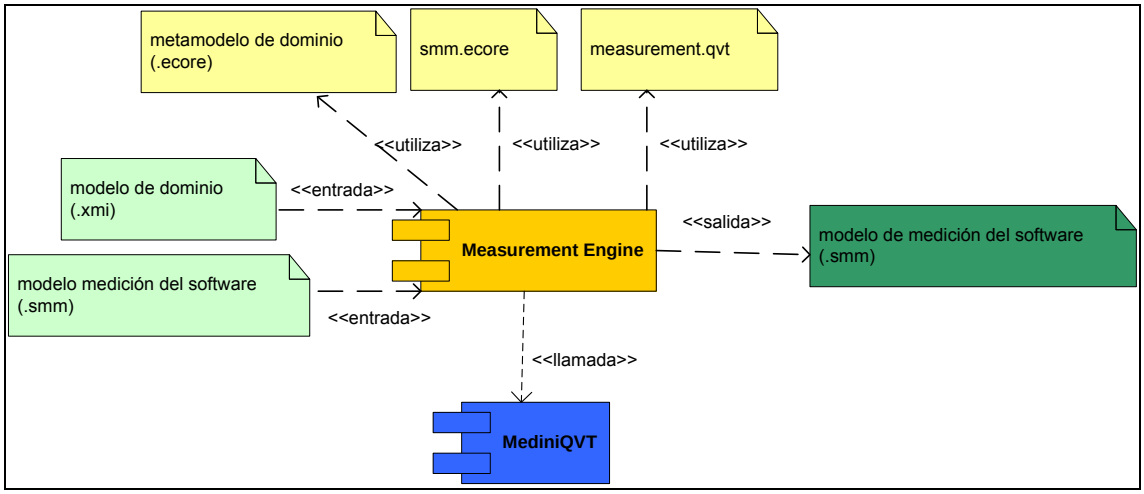


Figura 6-7. Transformación Final o Ejecución de la Medición

Todo este proceso de transformaciones aplica el principio de caja negra, es decir, el usuario no conoce el funcionamiento interno de las tareas que se están llevando a cabo. En la Figura 6-8 se resume el proceso completo de transformaciones llevado a cabo por el componente Measurement Engine.

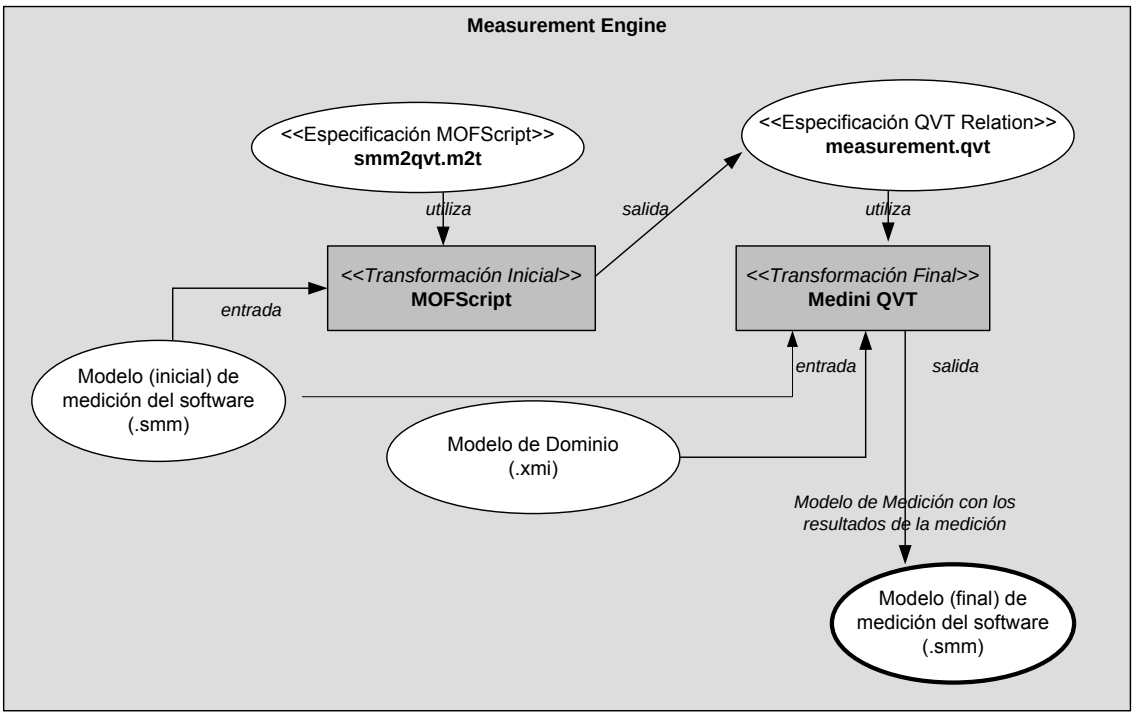


Figura 6-8. Measurement Engine

6.5.1. Especificación de los Modelos de Transformación.

Para ilustrar cómo son estos modelos de transformación presentamos a continuación un ejemplo sencillo de una medición genérica aplicada en el dominio de bases de datos relacionales. Las medidas base que se quieren obtener son las siguientes:

- NOA: "número de Atributos cuyo nombre sea id".
- NOT: "número de Tablas".

Las dos medidas utilizan el método de medición contar, y en el caso de la medida NOA se aplica una condición adicional, es decir, un método de medición parametrizado.

6.5.1.1. Modelo para la Transformación Inicial Model-to-Text.

En primer lugar, está el modelo MOFScript "smm2qvt.m2t" (véase Figura 6-9), que sirve para generar el modelo de transformación QVT "measurement.qvt".

```
texttransformation Smm2QVT (in model:"smm" ) {

      model.Root::main () {
            file("measurement.qvt");
            /* Imprime comentarios en la especificación qvt */
            printInitComments();

            /* Imprime en la especificación qvt
            el dominio a medir */
            self.elements->forEach(e:model.EntityClass)
                  e.printModels()

            /* Especificación QVT invariable */
            printBody();

            var domain:String

            /* Indica el dominio de entrada para
            calcular las medidas base */
            self.elements->forEach(e:model.EntityClass){
                  domain = e.name
                  e.printBaseMeasure()
            }
            /* Define la especificación para calcular el
             * método de medicion */
            model.printMeasurementMethod(domain);
            /* Imprime las consultas para calcular
            las medidas derivadas e indicadores */
            printQueries();
            /* Imprime comentarios finales*/
            printEndComments();

    }// MAIN
```

Figura 6-9. Extracto de la especificación de smm2qvt.m2t

Como se observa en la Figura 6-9 este modelo tiene como única entrada un modelo de medición del software. Incluye los siguientes métodos:

- **printInitComments.** Imprime comentarios en el modelo qvt.
- **printModels.** Define la cabecera del modelo QVT, indicando los modelos de entrada (modelo de dominio *domainModel* y modelo de medición *measurementDomainSrc*) y el modelo de salida (modelo de medición *measurementDomainDst*). Como se observa en la Figura 6-10, únicamente se debe indicar el nombre del dominio a medir (definido en el elemento *EntityClass*), ya que el resto de modelos son iguales para todas las mediciones. En este ejemplo el nombre del modelo de dominio sería “relational”.

```
//Imprime la especificación de los modelos de entrada.
       model.EntityClass::printModels(){
          println ("transformation Measurement(");
          println ("\tdomainModel:"+self.name+",");
          println ("\tmeasurementDomainSrc:smm,");
          println ("\tmeasurementDomainDst:smm)");
          println ("{");
       }
```

Figura 6-10. Método printModels

- **printBody.** Define el cuerpo general de la transformación (conjunto de reglas de transformación) para los elementos que permanecen invariantes en el modelo de medición de salida, por ejemplo: *InformationNeed*, *QualityModel*, *Attribute* o *EntityClass*. Este cuerpo es fijo para cualquier medición ya que estos elementos son independientes al dominio. En la Figura 6-11 se muestra una parte de este método.

```
printBody(){
        println("
      top relation Root {
            checkonly domain measurementDomainSrc source:smm::Root{};
            enforce domain measurementDomainDst target:smm::Root{};

      }
       top relation Attribute{
            checkonly domain measurementDomainSrc source:smm::Root{
                  elements = e:smm::Attribute{}
            };
            enforce domain measurementDomainDst target:smm::Root{
                  elements = e:smm::Attribute{}
            };
            when{
            Root(source,target);
            }
      }
```

Figura 6-11. Método printBody

- **printBaseMeasure.** Define la regla necesaria para calcular las medidas base utilizando la definición del dominio que se está midiendo. Como se observa en la Figura 6-12 el modelo de dominio forma parte del *checkonly* que indica una de las entradas de la medición. El otro modelo de entrada es el modelo de medición *measurementDomainSrc*. Como salida, definida en el *enforce*, está el modelo de medición *measurementDomainDst*.

```
model.EntityClass::printBaseMeasure(){
         println("
         top relation BaseMeasure{
            n : String;
            checkonly domain domainModel
d:"+self.name+"::"+self.measurementDomain+"{};
            checkonly domain measurementDomainSrc source:smm::Root{
                  elements = e:smm::BaseMeasure{
                        name = n,
                        status = 'pending'} };
            enforce domain measurementDomainDst target:smm::Root{
                  elements = e:smm::BaseMeasure{
                        name = n,
                        value = measurementMethodExecution (n),
                        status = 'processed'} };
            when{
                  MeasurementMethod(source,target);
                  Attribute(source, target);}
      }  ");
```

Figura 6-12. Método printBaseMeasure

- **printMeasurementMethod.** Define la especificación de la consulta (*query*) necesaria para obtener el valor de cada una de las medidas base del modelo de medición a partir del cálculo del método de medición. Dependiendo de la información definida en cada método de medición de cada medida se formará la especificación necesaria para calcular la medida base. Como se observa en la Figura 6-13, por cada entidad del dominio que se quiere medir (definida en el método de medición) se crea una consulta *EntidadDominio.allInstances()->size()*. Este método llama al método printConditions (véase Figura 6-14) que añade las condiciones (si las hubiera) definidas en los métodos de medición (métodos de medición parametrizados). Como se observa en la Figura 6-14, si existe más de una condición se concatenan con el operador lógico *and*.

```
model.Root:: printMeasurementMethod(domain:String)
       {
         println("
      query measurementMethodExecution(name:String) : Real
            {
        ");
           /* Meter todas las entidades a contar */
           self.elements->forEach(b:model.BaseMeasure){
          println("
          if (name = '"+b.name+"') then"
          );
          if(b.measurementMethod.conditions.size() = 0){
            /* Whithout Conditions */
                  println("\t\t\t"

      +domain+"::"+b.measurementMethod.entity+".allInstances()-
>size()+0.0"+
                  "\n\t\t"+"else");
          }//if
          else{ /* With conditions */
                print("\t\t\t"

+domain+"::"+b.measurementMethod.entity+".allInstances()"+"\n\t\t\t"+
                  "-
>select(m:"+domain+"::"+b.measurementMethod.entity+"|");
                  /* Print conditions */
                  b.measurementMethod.printConditions();

                  println("\n\t\t\t)->size()+0.0"+
                  "\n\t\t"+"else");
            }//else
            }
            println("\t\t-9999");
            self.elements->forEach(b:model.BaseMeasure)
            println("\t\tendif");
            println("}");
      }//print Measurement Method
```

Figura 6-13. Método printMeasurementMethod

```
model.MeasurementMethod::printConditions()
      {
        var count : integer;
        count = 0;
        self.conditions->forEach(c:model.Condition){
          if (count > 0) print(" and ");
            print("m."
                  +c.entityAttribute
                  +c.condition
                  +""+c.value+"");
            count = count + 1;
        }
      }
```

Figura 6-14. Método printConditions

- **printQueries.** Define las dos consultas necesarias para calcular las medidas derivadas e indicadores, la consulta *executeOperation* para calcular el valor obtenido de una función de cálculo o modelo de análisis, y la consulta *executeInterpretation* que evalúa los criterios de decisión para definir el valor final del indicador. Estas consultas son siempre fijas e independientes del dominio de medición, ya que son datos derivados y la lógica operacional no depende del dominio que se esté midiendo. En la Figura 6-15 y Figura 6-16 se muestra un extracto del método.

```
      printQueries(){
        println("

      /* Query Domain general */
      query executeOperation(m1:Real, m2:Real, operation:String) :
Real
      {
            if(operation = '+') then
                  m1 + m2
            else
                  if(operation = '-') then
                        m1 - m2
                  else
                        if(operation = '*') then
                              m1*m2
                        else
                              if(operation = '/') then
                                    m1/m2
                              else
                                    -1.0
                              endif
                        endif
                  endif
            endif
      }
```

Figura 6-15. Método printQueries

```
query executeInterpretation(
      vmin1:Real, vmax1:Real, des1:String,
      vmin2:Real, vmax2:Real, des2:String,
      vmin3:Real, vmax3:Real, des3:String,
      vmin4:Real, vmax4:Real, des4:String,
      v1:Real, v2:Real, op:String):String
{
      if(op='+')then /* Procesar Suma*/
            if((v1+v2) >= vmin1 and (v1+v2) < vmax1)then
             des1 /* Valor del Criterio 1*/
            else
             if((v1+v2) >= vmin2 and (v1+v2) < vmax2)then
              des2 /* Valor del Criterio 2*/
             else
              if((v1+v2) >= vmin3 and (v1+v2) < vmax3)then
               des3 /* Valor del Criterio 3*/
              else
                if((v1+v2) >= vmin4 and (v1+v2) <= vmax4)then
                  des4 /* Valor del Criterio 4*/
                 else
                  'ERROR IN DEFINITION'
                  endif
                endif
              endif
            endif
      else
           …
           …


}
```

Figura 6-16. Método printQueries (cont.)

- **printEndComments.** Imprime comentarios al final del modelo qvt.

6.5.1.2. Modelo para la Transformación Final Model-to-Model.

En segundo lugar está el modelo QVT "measurement.qvt", que contiene la medición específica del dominio a medir y se crea mediante la transformación anterior. A continuación se muestran las partes del modelo QVT que contienen reglas específicas para el cálculo de las medidas NOA y NOT de una base de datos relacional (en negrita se marcan las palabras que dependen del dominio concreto, y que son creadas automáticamente en la transformación anterior. El resto de código es común para cualquier dominio):

- **Cabecera de la transformación.** En ella se definen los modelos de entrada y salida (véase Figura 6-17). En este caso los modelos de entrada son el modelo del dominio relacional y el modelo de medición; y el modelo de salida es el modelo de medición extendido.

```
transformation Measurement (
      domainModel:relational,
      measurementDomainSrc:smm,
      measurementDomainDst:smm)
{
```

Figura 6-17. Cabecera de la transformación

- **Regla para el cálculo de la medida base**. La regla de transformación que se muestra en la Figura 6-18 permite calcular el valor de la medida base en el dominio de las bases de datos relacionales. Para calcular el valor de la medida es necesario ejecutar la consulta measurementMethodExecution. Esta consulta es específica de la medición y por tanto variará para cada dominio y cálculo de medida base.

```
top relation BaseMeasure{
      n : String;
      checkonly domain domainModel d:relational::RelationalSchema{};
      checkonly domain measurementDomainSrc source:smm::Root{
            elements = e:smm::BaseMeasure{
                  name = n,
                  status = 'pending'
            }
      };
      enforce domain measurementDomainDst target:smm::Root{
            elements = e:smm::BaseMeasure{
                  name = n,
                  value = measurementMethodExecution (n),
                  status = 'processed'
            }
      };
      when{
            MeasurementMethod(source,target);
            Attribute(source, target);
      }
}
```

Figura 6-18. Regla de transformación para calcular la medida base

- **Query measurementMethodExecution.** Esta consulta contiene el código necesario para calcular las medidas base que intervienen en la medición, en este caso las medidas base son NOT y NOA. Como se observa en la Figura 6-19 cada medida contiene la especificación necesaria para calcular el valor de la medida y dependen de la especificación de la medición. Nótese que para la medida NOA se ha añadido una condición ya que aplica un método de medición parametrizado.

```
query measurementMethodExecution(name:String) : Real
      {
      if (name = 'NOA') then
            relational::Attribute.allInstances()
            ->select(m:relational::Attribute|m.name='id')->size()+0.0
      else
            if (name = 'NOT') then
                  relational::Table.allInstances()->size()+0.0
            else
                  -9999
            endif
      endif
}
```

Figura 6-19. Función measurementMethodExecution

El resto de código del modelo QVT es idéntico para cualquier otro dominio ya que no depende del dominio sino de los elementos del modelo de medición.

6.6. Medición con SMTool.

Utilizando SMTool el usuario puede llevar a cabo una medición (o acción de medir) de acuerdo al método de SMF del siguiente modo:

1. **Creación de un proyecto SMTool.** Para empezar a trabajar con SMTool tiene que crearse un proyecto SMTool. Con esta acción se configura y prepara el entorno en el que se ejecutará la medición. Esta acción consiste principalmente en la creación de los directorios mínimos recomendados para gestionar una medición:
   - **Input:** directorio utilizado, por defecto, para incorporar los elementos de entrada definidos por el usuario (metamodelos de dominio, modelos de dominio y modelos de medición).
   - **Output:** directorio donde se crean los resultados de la medición.
   - **Transformation:** directorio donde se almacena la especificación QVT.
2. **Incorporación del metamodelo de dominio.** Debido al hecho de que cada instancia del proceso de medición corresponde a un dominio específico, es necesario que se incluya el correspondiente metamodelo de dominio. Por ejemplo, si queremos medir atributos de los modelos UML es necesario añadir el metamodelo de UML y si queremos medir propiedades de un código Java es necesario añadir el metamodelo que permite representar dicho código en forma de modelos. El metamodelo de dominio debe ser un fichero en formato ECORE creado utilizando la opción *Ecore Model* de EMF y puede ser editado por cualquier editor incluido en EMF.
3. **Creación del modelo de medición.** El modelo de medición se crea acorde a SMM (pre-incluido como parte de SMTool) y se edita por medio de SMML-Editor. Este modelo es la entrada a la medición y contiene toda la información necesaria sobre el proceso de medición, producto y proyecto. Como se ha dicho en el apartado 6.4, existen dos opciones disponibles para la definición de los modelos de dominio. Por un lado, el uso del clásico editor *Sample Reflective Ecore Model Editor* incluido en SMF para crear el fichero smm (véase Figura 6-3). Por otro lado, el uso de SMML-Editor para definir los diagramas de medición expresados en SMML de una forma gráfica (véase Figura 6-4). Un diagrama SMML (almacenado como un fichero smmd) sólo se puede interpretar correctamente si su correspondiente fichero smm (modelo de medición sin aspectos gráficos) se ha creado previamente. Sin embargo, el usuario puede dibujar directamente en el diagrama SMML porque SMML-Editor refleja los cambios entre el fichero smm y el diagrama SMML de una manera bidireccional. Esta opción permite que los usuarios trabajen únicamente con diagramas SMML, "olvidándose" de los ficheros smm.
4. **Ejecución de la Medición.** Una vez que los elementos anteriores se han incorporado al proyecto de medición, se puede llevar a cabo una medición automática. Para ello el usuario tiene que ejecutar el "run configurations" (véase Figura 6-20) de Eclipse y crear una nueva instancia de SMTool llamada "Measurement Specification". En esta instancia se deben definir los siguientes elementos:
   - **Metamodelo y modelos de entrada:** rutas donde están almacenados los ficheros del metamodelo de dominio, el modelo de dominio y el modelo de medición.
   - **Ruta de Transformación:** ruta del directorio donde se almacenará el fichero de especificación qvt.
   - **Modelo de Salida de Medición:** ruta y nombre del modelo de medición con los resultados incluidos. Este modelo es la extensión del modelo de medición de entrada.

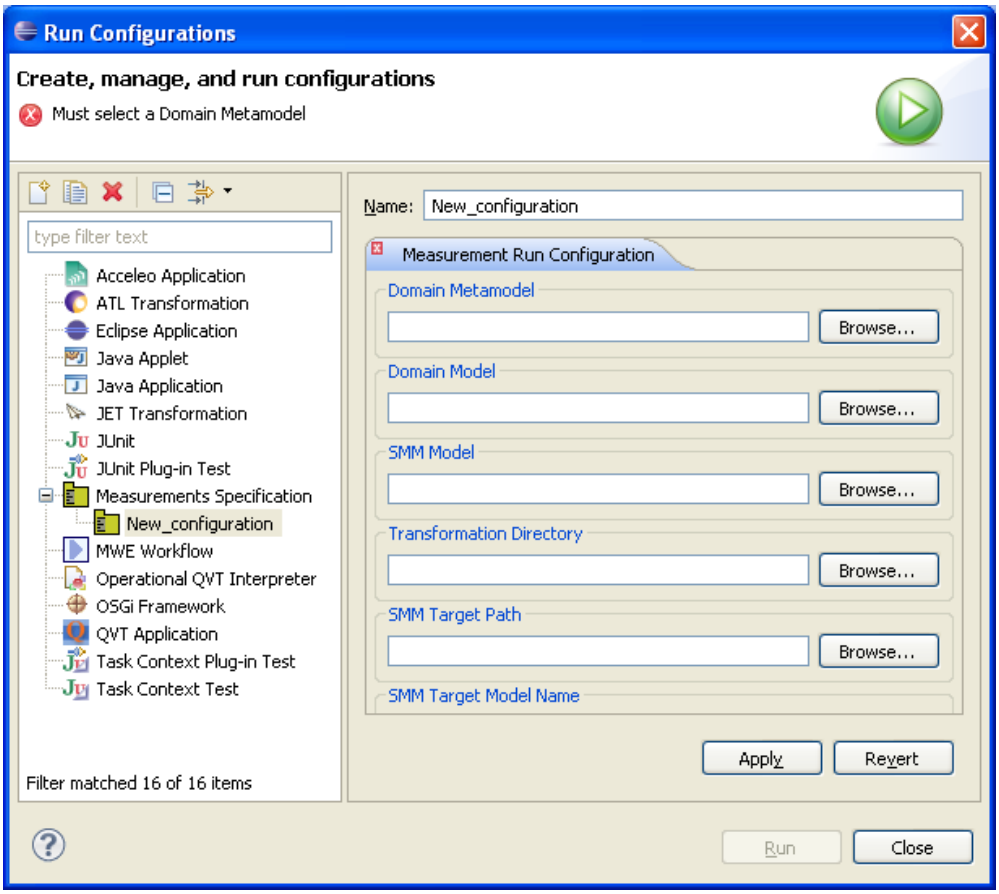


Figura 6-20. Formulario de “Measurement Specification” de SMTool

Con el objetivo de facilitar al usuario el proceso de medición, SMTool incorpora un menú específico que incluye las 4 tareas de medición que se deben llevar a cabo (véase Figura 6-21). El menú incluye los siguientes comandos.

- Domain Metamodel Definition.
- Domain Model Definition.
- Software Measurement Model Definition.
- Mesurement Execution.

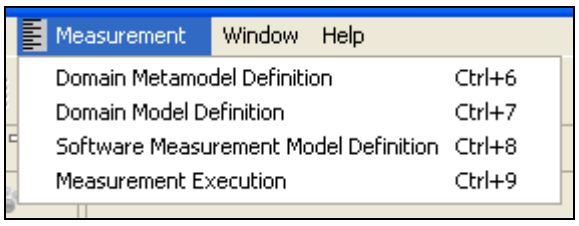


Figura 6-21. Menú específico de medición.

6.7. Conclusiones.

SMTool es una herramienta de software libre, pública e integrada en SMF para llevar a cabo mediciones genéricas de cualquier dominio software. Para conseguir este objetivo SMTool está basada en el paradigma MDE, aplicando la tecnología MDA.

Las mediciones se pueden modelar de una manera similar a los clásicos artefactos software, tales como diagramas UML, bases de datos, código, etc. SMTool da soporte a esta nueva funcionalidad de una manera potente, gracias al metamodelo de medición y al lenguaje gráfico. Todo esto soportado por los componentes "SMML-Editor" y "Measurement Engine" de SMTool.

Los modelos de medición contienen toda la información necesaria para dar soporte a las actividades de medición. Gracias a ello, SMTool puede calcular automáticamente los resultados de las medidas definidas. Es importante dejar claro que se pueden modelar todas las medidas que se pueden definir en términos de propiedades intrínsecas de artefactos software (por ejemplo, número de tablas de una base de datos) y obtener sus resultados automáticamente. Por contra, las medidas que necesitan intervención humana para encontrar sus valores (por ejemplo, coste de una actividad) se pueden representar por un modelo de medición, pero obviamente el resultado no se calcula ni obtiene automáticamente con SMTool. En estas situaciones consideramos que su "forma de medir" es manual.

“Largo es el camino de la enseñanza por medio de teorías; breve y eficaz por medio de ejemplos” (Lucio Anneo Séneca)

7. Caso de Estudio.

En este capítulo se describe con detalle un caso de estudio en el que se ha aplicado el marco conceptual de medición genérica SMF mediante el uso de la herramienta SMTool. En primer lugar se presentan las características generales de la empresa de software en la que se ha realizado y el proyecto que se ha abordado. En segundo lugar se describen los pasos seguidos para llevar a cabo el caso de estudio de acuerdo al método de trabajo (véase Capítulo 2). En el tercer apartado se explica cómo se aplicó SMF y se muestran los resultados obtenidos. Por último, se presentan las conclusiones, lecciones aprendidas y las consideraciones para el futuro.

7.1. Introducción.

Como se ha descrito en el Capítulo 2, el grupo de referencia y la empresa con la que colaboró el grupo de investigación para conseguir su objetivo fue Indra Software Labs.

Con el objetivo de proporcionar el soporte necesario para satisfacer las necesidades de Indra con respecto a la calidad del software, se aplicó SMF siguiendo un método y mediante la definición y ejecución de modelos de medición con SMTool para los dominios seleccionados por la organización.

Por nuestra parte, el principal motivo de este caso de estudio fue comprobar la utilidad real de SMF a la hora de facilitar a una empresa las mediciones software en cualquier dominio y la posibilidad de reutilizar la medición dentro de un mismo dominio. Para ello, en primer lugar se seleccionó un proyecto representativo de aplicación para, a continuación, identificar las entidades relevantes a medir de distintos dominios y aplicar el marco SMF para la definición de los modelos de medición y la ejecución de las mediciones correspondientes.

A continuación se describe el contexto de la empresa y el proyecto seleccionado para llevar a cabo el caso de estudio.

7.2. Contexto.

Indra Software Labs es una compañía de desarrollo software que trabaja en los sectores de Transporte y Tráfico, Energía e Industria, Administración Pública y Sanidad, Servicios Financieros, Seguridad y Defensa y Telecom y Media. Ha recibido el reconocimiento por el SEI (Software Engineering Institute) del Nivel de Madurez 3 del modelo CMMI, esto implica, entre otras cosas que los proyectos de Indra se lleven a cabo en un contexto de garantía de calidad en la que la medición del software es un área fundamental. El departamento de calidad de la empresa ha trabajado para que todos los proyectos de la compañía sigan un riguroso control de calidad, formado principalmente por auditorias de calidad anuales y reporte de indicadores cuatrimestrales.

Actualmente, la compañía tiene métodos con los que llevar a cabo las mediciones del software específicos para los dominios involucrados (código java, requisitos del sistema, etc). Además, debido a las necesidades de las mejoras tecnológicas, son varias las herramientas utilizadas y están en continuo cambio. Existe por tanto un interés por parte de la compañía para homogeneizar los procesos de medición del software y utilizar un entorno que permita la definición de modelos de medición y el cálculo de las medidas en cualquier dominio.

El proyecto "*IndraSalud 2.0*" pertenece al departamento de Sanidad de Indra y su objetivo es la implantación de una aplicación software (con tecnología J2EE) que gestione toda la actividad sanitaria de un hospital. Se trata de un proyecto internacional de gran tamaño que cuento con clientes de varios países como Brasil y Portugal.

El proyecto está dividido en módulos sanitarios (o sub-proyectos). De esta manera se consigue centralizar cada actividad de un área sanitaria en un módulo. Aún así existen dependencias entre los módulos, como por ejemplo entre urgencias y quirófano, hospitalización y farmacia, etc.

Debido al gran tamaño de la aplicación el proyecto está siendo desarrollado y gestionado por varias sedes nacionales: Ciudad Real, Madrid, Sevilla, Valencia, Gijón y Málaga. En la Tabla 7-1 se muestra un resumen de los módulos sanitarios y la localidad en la que se está desarrollando cada uno.

Módulo	Área Sanitaria	Localidad		Módulo	Área Sanitaria	Localidad
ISHOS	Hospitalización	Ciudad Real		ISTAR	Gestión de tareas	Sevilla
ISURG	Urgencias	Ciudad Real		ISPRE	Prescripciones	Sevilla
ISQUI	Quirófano	Ciudad Real		ISCIT	Citación	Valencia
ISFAR	Farmacia	Ciudad Real		ISEC	Estación Clínica	Valencia
ISHDI	Hospital de Día	Ciudad Real		ISFAC	Facturación	Gijón
ISDOC	Gestión de Informes	Ciudad Real		ISPOB	Población	Málaga
ISCAT	Catálogo Sanitario	Ciudad Real		ISDIE	Dietas	Madrid
ISGEP	Peticiones Médicas	Ciudad Real				
ISODO	Odontología	Ciudad Real				
ISGEN	Genograma	Ciudad Real				

Tabla 7-1. División del proyecto Indrasalud 2.0 por módulos y localidades

Como se observa en la tabla anterior, más de la mitad de los módulos se llevan a cabo en la factoría de Ciudad Real, que también ha sido la sede elegida para la aplicación del marco de medición genérico.

Por estos motivos, hemos considerado que la aplicación del marco de medición en este proyecto puede ser beneficiosa para demostrar su potencial en un proyecto real de Ingeniería del Software de gran envergadura.

7.3. Descripción del Caso de estudio.

En este apartado se presenta el caso de estudio de acuerdo al método de trabajo descrito en el Capítulo 2, en el que se han seguido las etapas de Yin (Yin, 2002) y se ha utilizado la plantilla de (Brereton et al., 2008).

7.3.1. Antecedentes.

- **Identificación del tópico de investigación.** Se realizó una búsqueda en la bibliografía existente acerca de la medición genérica del software, particularmente en el contexto de modelos (véase Capítulo 3).
- **Definición de la principal pregunta de investigación.** ¿Es SMF un marco de medición útil para llevar a cabo mediciones de software?
- **Identificación de preguntas adicionales que derivaron de la principal.**
  - ¿SMF mejora la medición frente a la manera tradicional de medición llevada a cabo con herramientas específicas de cada dominio?

7.3.2. Diseño.

- **Caso de estudio múltiple o único.** El caso de estudio fue único, es decir se llevó a cabo en una sola empresa y en un proyecto de la misma. El estudio se aplicó en varios módulos y dominios dentro del mismo proyecto.
- **Objeto de estudio.** Las unidades de análisis fueron tres. La primera unidad de análisis fue el módulo de desarrollo ISHOS (véase apartado 7.4.1) en el que se midieron los requisitos. La segunda unidad de análisis fue el módulo de desarrollo ISURG (véase apartado 7.4.2) donde se aplicó una medición de los diagramas de clases. La última unidad de análisis fue el módulo de desarrollo ISFAR (véase apartado 7.4.3) sobre el que se midió el esquema de bases de datos relacional.
- Las **Sub-preguntas** que se formularon fueron las siguientes, aplicadas a los tres dominios de interés del proyecto (requisitos, diagramas estructurales de diseño en UML y esquemas de bases de datos relacionales):
    - ¿Es adecuado el proceso de medición llevado a cabo para satisfacer las necesidades de medición de los requisitos, diagramas de clases UML y esquemas de bases de datos relacionales?
    - ¿Es entendible el modelo de medición definido para el dominio de los requisitos, diagramas de clases UML y esquemas de bases de datos relacionales?
    - ¿Se identifican claramente los resultados de la medición en el modelo?
    - ¿Es práctica y usable la herramienta SMTool para definir y ejecutar la medición?
    - ¿Es reutilizable el modelo de medición para medir los requisitos, diagramas de clases UML y esquemas de bases de datos relacionales en otros módulos de desarrollo?

Los datos recopilados fueron datos cualitativos y se obtuvieron a partir de las respuestas subjetivas del cuestionario proporcionado a los participantes: el analista de la medición y los responsables de los módulos ISHOS, ISURG e ISFAR. El cuestionario puede consultarse en el Anexo D.

7.3.3. Selección del caso.

Tal como se ha descrito en el apartado anterior, los módulos en los que se aplicó el marco SMF fueron ISHOS, ISURG e ISFAR, que fueron elegidos por las siguientes razones:

- Grado de avance. Los tres módulos se encontraban en un estado de desarrollo avanzado (fase de pruebas) y por tanto se disponía de toda la documentación necesaria para ejecutar las mediciones en los dominios seleccionados.
- Participación directa. La investigadora había participado en las tareas de análisis y diseño de los módulos por lo que tenía un conocimiento amplio de los mismos para facilitar la aplicación del caso.

El **contexto** en el que se aplicó SMF fue el siguiente:

- Necesidad de medir los proyectos de desarrollo y registrar los resultados en el informe cuatrimestral de indicadores y gestión de calidad.

- Existencia de herramientas para el modelado de los dominios involucrados en la medición.
- Inexistencia de herramientas que permitieran definir y calcular nuevas medidas e indicadores.

7.3.4. Procedimientos y roles.

- Los **roles** fueron: analista de la medición y responsables de los módulos de desarrollo.
- **Procedimiento.** En primer lugar se repartió a los participantes los manuales de la herramienta SMTool que se iba a utilizar. A continuación se aplicó el método definido en el marco de medición genérico SMF de la siguiente manera:
  - **Definición de metamodelos de dominio.** El analista de la medición recopiló los metamodelos de los dominios de la siguiente manera:
    - **Metamodelo de requisitos.** El metamodelo se creó nuevo a partir de la estructura de los requisitos definidos para el proyecto.
    - **Metamodelo de diagrama de clases UML.** El metamodelo se descargó de una página pública (http://www.emn.fr/z-info/atlanmod/index.php/Ecore), y se dejó únicamente la parte correspondiente a los diagramas de clases con el objetivo de dejar el metamodelo lo más simple posible pero proporcionando toda la funcionalidad requerida. El motivo de ello fue que los responsables de los módulos no tenían gran conocimiento del uso de metamodelos.
    - **Metamodelo de bases de datos relacional.** Este metamodelo se adoptó del definido en (García et al., 2006b).
  - **Definición de modelos de dominio.** El analista de la medición definió los modelos de dominio conforme a los correspondientes metamodelos y a partir de los diagramas proporcionados por los responsables de los módulos.
  - **Definición de los modelos de medición.** A partir de una notación textual definida por el analista de la medición los responsables de los módulos definieron los modelos de medición para cada dominio utilizando SMML Editor.

    Estas tres tareas se describen en detalle en los apartados 7.4.1, 7.4.2 y 7.4.3.
  - **Ejecución de la medición.** Los responsables de los módulos ejecutaron la medición sobre cada dominio mediante la herramienta SMTool. La ejecución se presenta en detalle en el apartado 7.4.4.

Una vez ejecutada la medición el analista de la medición repartió unos cuestionarios que rellenaron los encargados de los módulos para evaluar el uso de SMF.

7.3.5. Colección de datos.

La colección de datos se ha realizado a partir de las respuestas a las preguntas de investigación proporcionadas a los participantes, que se detallan a continuación:

- ¿Es adecuado el proceso de medición llevado a cabo para satisfacer las necesidades de medición de los requisitos, diagramas de clases UML y esquemas de bases de datos relacionales?

  - ¿Son suficientes las etapas definidas en el método de SMF para llevar a cabo las mediciones?
- ¿Se entiende el modelo de medición definido para el dominio de medición?
  - ¿Está de acuerdo que el modelo de medición gráfico facilita la definición del modelo de medición?
  - ¿Está de acuerdo que el modelo de medición gráfico permite ver toda la información de la medición que se va a ejecutar?
- ¿Se identifican claramente los resultados de la medición en el modelo?
  - ¿Está de acuerdo que el modelo de medición muestra los resultados de la medición de una manera entendible?
- ¿Es práctica y usable la herramienta SMTool para definir y ejecutar la medición?
  - ¿Está de acuerdo que la interfaz de SMTool resulta intuitiva para realizar mediciones?
  - ¿Está de acuerdo que la herramienta SMTool cubre las necesidades de medición de las herramientas existentes (Caliber, Together y Power Designer)?
  - ¿Está de acuerdo que SMTool facilita la definición y ejecución de mediciones de software)
- ¿Es aplicable el modelo de medición en otros módulos de desarrollo?
  - ¿Se puede importar un modelo de medición para ejecutar una medición en el mismo dominio pero sobre un módulo de desarrollo distinto?
  - ¿Está de acuerdo que el marco de medición es genérico desde el punto de vista que se puede medir cualquier tipo de entidad (Requisitos, Diagramas UML, etc.)?

7.3.6. Plan de Validez.

Con el fin de resolver posibles amenazas a la validez del caso de estudio se tuvieron en cuenta los siguientes aspectos:

- **Validez del constructo.** Las preguntas incluidas en el cuestionario se basaron en suposiciones ciertas que permiten indicar cuándo un marco de medición genérico cumple su propósito y es útil:
  - Un buen marco de medición genérico es aquel que permite definir modelos de medición utilizando medidas genéricas.
  - Un buen marco de medición genérico es aquel que permite ejecutar modelos de medición sobre cualquier dominio.
  - Un buen marco de medición genérico es aquel que tiene un método definido para la ejecución de mediciones.
  - Un buen marco de medición genérico es aquel que tiene soporte para ejecutar las mediciones.

  Estos supuestos eran bastante básicos y aceptables para un buen marco de medición genérico.
- **Validez Externa.** Tal como se indicó en la introducción, la empresa en la que se ha aplicado el caso de estudio tiene un nivel de madurez CMMI 3, y por tanto es una empresa representativa para poder aplicar el marco de medición genérico ante la

necesidad a partir de dicho nivel de medir de forma integrada y homogénea diversos tipos de entidades software (procesos, proyectos, productos). Los resultados de este caso de estudio se pueden por tanto generalizar a las empresas de desarrollo de software en condiciones similares a INDRA.

- **Fiabilidad.** Este aspecto se refiere a en qué medidas los datos y el análisis son fiables y no depende de los investigadores. Este caso de estudio los resultados pueden verse afectados por el punto de vista del investigador, ya que formaba parte del mismo. Sin embargo, la colección de datos fueron tomados de otras tres fuentes (encargados de los módulos de desarrollo) con el objetivo de obtener un análisis de los datos más completo y objetivo en la mayor medida de lo posible. Además, el investigador que trabajaba como responsable de calidad en el proyecto trató de hacer un buen trabajo desde el punto de vista de un analista de medición y las conclusiones obtenidas no se vieron afectadas por ninguna motivación especial de la investigación.

7.3.7. Resultado del análisis del caso de estudio.

En este apartado se muestra el resultado del análisis del caso de estudio para cada uno de los dominios.

- **¿Es adecuado el proceso de medición llevado a cabo para satisfacer las necesidades de medición?**
  - **Requisitos.** Sí es adecuado, ya que permite rellenar la plantilla de indicadores proporcionada por el departamento de calidad.
  - **Diagramas de clases UML.** Sí es adecuado y útil para el analista programador ya que permite conocer la mantenibilidad de los diagramas de clases.
  - **Esquema de bases de datos relacionales.** Sí es adecuado y útil para el analista programador ya que permite conocer la mantenibilidad de los diagramas que son puntos críticos en el desarrollo de software.
- **¿Se entiende el modelo de medición definido para el dominio elegido?**
  - En todos los puntos coinciden que necesitaron tiempo para familiarizarse con el lenguaje. Una vez que se habían familiarizado el modelado les resultó fácil.
    - En esta pregunta surgió la necesidad de importar el modelo de medición a partir de plantillas proporcionadas por el departamento de calidad.
    - Otra propuesta fue que el departamento de calidad de la empresa se encargara de la definición de los modelos de medición y metamodelos de dominio necesarios para realizar las mediciones requeridas por el departamento de calidad. De esta manera la tarea de medición se centraba únicamente en la definición de los modelos de dominio y por tanto no estaba restringida la tarea de medición a los analistas de medición.
- **¿Se identifican claramente los resultados de la medición en el modelo?**
  - Relacionada con la pregunta anterior. Una vez que se había familiarizado con el lenguaje les resultó fácil identificar los resultados de la medición.
  - En este punto surgió la necesidad de generar a partir de los modelos de medición documentos en un formato estándar abierto como cvs (comma-separated values).
- **¿Es práctica y usable la herramienta SMTool para definir y ejecutar la medición?**

  - En este punto coincidieron en que la ejecución era fácil una vez que tenían los modelos y metamodelos de medición. Pero destacaron su falta de experiencia en la definición de modelos y metamodelos.

- **¿Es aplicable el modelo de medición en otros módulos de desarrollo?**
  - La respuesta de todos los participantes fue positiva. En este punto se propuso la mejora de tener vinculado a la herramienta un repositorio de mediciones de manera que las mediciones se subieran automáticamente a un repositorio común. De esta manera el departamento de calidad podía tener el control de todas las mediciones ejecutadas por cada proyecto. Esta propuesta está cubierta gracias a la posibilidad que ofrece de crear proyectos SVN[23] (subversion) o CVS[24] (Concurrent Versions System). Los proyectos SVN o CVS permiten crear proyectos a partir de repositorios, de esta forma los usuarios de SMTool pueden crear nuevas mediciones; y modificar y consultar las existentes.

7.4. Aplicación de SMF en el Caso de Estudio.

En este apartado se presenta la definición de las mediciones en distintos dominios que se han llevado a cabo en el proyecto IndraSalud 2.0 mediante la aplicación del marco SMF y siguiendo su método de trabajo.

El estudio se ha centrado en dominios de la fase del proyecto de desarrollo de software "análisis y diseño del sistema". Por cada dominio de medición se han realizado las siguientes tareas:

- **Definición del dominio:** consiste en primer lugar en la identificación e incorporación del metamodelo del dominio sobre el cual se va a realizar la medición del software; y en segundo lugar en la definición del modelo de dominio que va a medirse.
- **Definición del modelo de medición:** consiste en la definición de las medidas necesarias para llevar a cabo las mediciones de software para cada uno de los dominios mencionados. Para ello se presentan los modelos de medición del software con sus correspondientes constructores. Para cada dominio se presenta en primer lugar el modelo de medición realizado por el analista de la medición, representado en una notación textual (datos organizados en tablas). Y en segundo lugar, se presenta su correspondiente modelo representado con SMML que definió cada responsable de los módulos.

7.4.1. Medición de Requisitos.

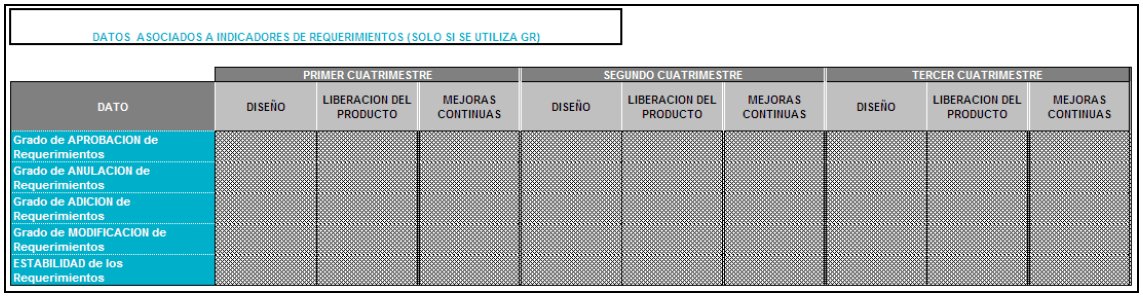
El primer dominio que se evaluó fue el de "Requisitos" del módulo ISHOS del proyecto IndraSalud dentro de la actividad "análisis de requisitos". Dentro de la actividad de análisis es fundamental que a través de una colección de requisitos funcionales y no funcionales, el desarrollador o desarrolladores del software comprendan completamente la aplicación a desarrollar. El análisis de requisitos es una actividad crítica en la creación del producto software, y por esta razón es fundamental aplicar mediciones para asegurar la calidad de esta actividad.

[23] Disponible en http://subversion.apache.org/

[24] Disponible en http://www.nongnu.org/cvs/

Para la definición de esta medición fue muy útil la plantilla de indicadores de Indra. Esta plantilla de indicadores recoge los indicadores de calidad de varios dominios que exige el departamento de calidad de Indra para cada proyecto de desarrollo y que debe reportarse cuatrimestralmente.

El objetivo de la medición que se presenta en este apartado es determinar la estabilidad de requisitos comparando una versión con respecto a la anterior por cada fase de desarrollo del proyecto. La Figura 7-1 muestra la entrada de datos necesaria para medir la estabilidad de requisitos definida en la plantilla de indicadores de calidad, y en la Figura 7-2 el valor de los indicadores calculados a partir de los datos y mediante los umbrales de los indicadores definidos en Calidad.



DATOS ASOCIADOS A INDICADORES DE REQUERIMIENTOS (SOLO SI SE UTILIZA GR)

DATO	PRIMER CUATRIMESTRE			SEGUNDO CUATRIMESTRE			TERCER CUATRIMESTRE		
	DISEÑO	LIBERACION DEL PRODUCTO	MEJORAS CONTINUAS	DISEÑO	LIBERACION DEL PRODUCTO	MEJORAS CONTINUAS	DISEÑO	LIBERACION DEL PRODUCTO	MEJORAS CONTINUAS
Grado de APROBACION de Requerimientos									
Grado de ANULACION de Requerimientos									
Grado de ADICION de Requerimientos									
Grado de MODIFICACION de Requerimientos									
ESTABILIDAD de los Requerimientos									

Figura 7-1. Entrada de datos para la medición de la estabilidad de requisitos

DATO	FASE DISEÑO			FASE LIBERACION DEL PRODUCTO			MEJORAS CONTINUAS		
	Valor Indicador (%)	Valor Objetivo (%)	Valor Acumulado (%)	Valor Indicador (%)	Valor Objetivo (%)	Valor Acumulado (%)	Valor Indicador (%)	Valor Objetivo (%)	Valor Acumulado (%)
PRIMER CUATRIMESTRE									
Grado de APROBACION de Requerimientos	N/A	<=15%	2,22	N/A	<=10%	6,42	N/A	N/A	N/A
Grado de ANULACION de Requerimientos	N/A	>=90%	69,36	N/A	>=90%	69,56	N/A	N/A	N/A
Grado de ADICION de Requerimientos	N/A	<30%	4,49	N/A	<=10%	4,15	N/A	N/A	N/A
Grado de MODIFICACION de Requerimientos	N/A	<=25%	7,32	N/A	<=5%	12,08	N/A	N/A	N/A
ESTABILIDAD de los Requerimientos	N/A	>=50%	73,7	N/A	>=75%	54,46	N/A	N/A	N/A
ANÁLISIS DE DATOS de indicadores asociados a: Requerimientos									

Figura 7-2. Análisis de datos de la estabilidad de requisitos

Como se muestra en las figuras anteriores para medir la estabilidad de los requisitos es necesario calcular el grado de APROBACIÓN, ANULACIÓN, ADICIÓN y MODIFICACIÓN de los requisitos.

En los siguientes subapartados se muestra en detalle las etapas llevadas a cabo para aplicar SMF en el dominio de requisitos.

7.4.1.1. Definición del dominio.

Para llevar a cabo la medición del software por medio de la aplicación de SMF, se necesitó en primer lugar el metamodelo de dominio. La Figura 7-3 muestra el metamodelo de requisitos (requirements.ecore), que es el metamodelo que caracteriza el tipo de entidad que se va a medir. Este metamodelo se definió en ecore y es una de las entradas del proceso de SMF.

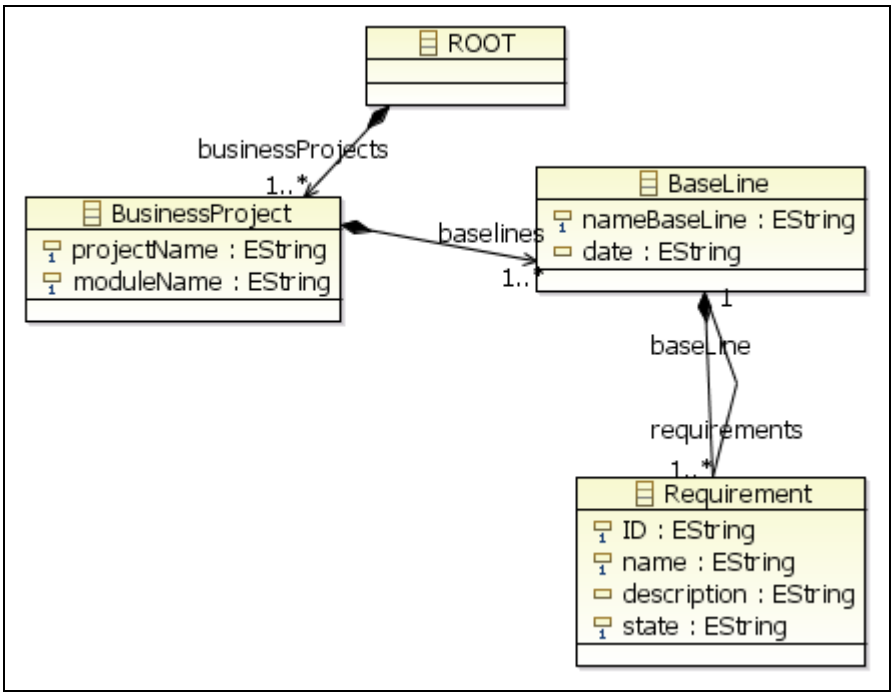


Figura 7-3. Metamodelo de requisitos

La segunda entrada del proceso de medición es la instancia o modelo del dominio que se va a medir (véase Figura 7-4). Este modelo de dominio se definió a partir del metamodelo presentado en la Figura 7-3 y contenía todo el listado de requisitos que se habían definido para el módulo ISHOS del proyecto IndraSalud.

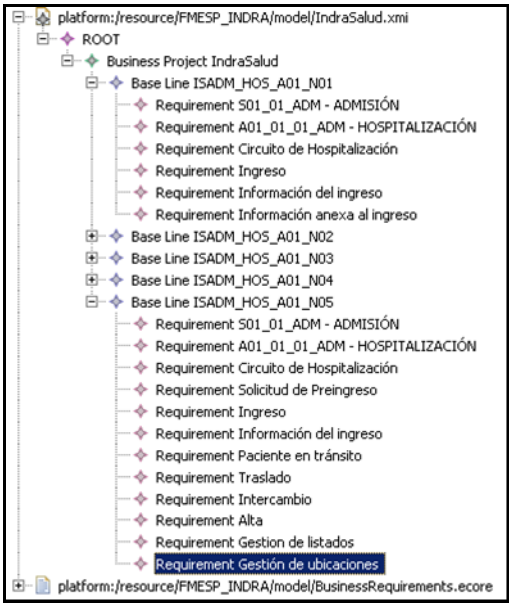


Figura 7-4. Fragmento del listado de elementos del modelo de requisitos del módulo ISHOS

7.4.1.2. Definición de los Modelos de Medición.

La tercera entrada necesaria para aplicar SMF es el modelo de medición del software. Las medidas que se seleccionaron para llevar a cabo una medición en el dominio de requisitos se presentan de forma textual en las tablas Tabla 7-2 y Tabla 7-3. Las medidas definidas pretenden satisfacer la necesidad de información del departamento de calidad de conocer la estabilidad de requisitos del módulo ISHOS para la versión v.01.00.01.

Elementos del metamodelo vs Instancias del Modelo				
Elementos del Metamodelo (clases MOF)	**Instancias M2 en el modelo de medición**	**Relaciones del Metamodelo (asociaciones MOF)**		**Links de M2 en el modelo de medición**
Atributo	Tamaño	Tiene	Entidad	Requirement
			Atributo	Tamaño
Entidad	Requirement	Definida por	Modelo de Calidad	ISO 9126
			Entidad	Requirement
Concepto medible	Estabilidad	Evalúa	Modelo de Calidad	ISO 9126
			Concepto medible	Estabilidad
Modelo de Calidad	ISO 9126	Relaciona	Atributo	Tamaño
			Concepto medible	Estabilidad
Necesidad de Información	Conocer la estabilidad de los requisitos	Está asociado con	Concepto medible	Estabilidad
			Necesidad de Información	Conocer la estabilidad de los requisitos
Medida	NCR, NAR, NADR, NMR, TOR	Definida por	Medida	NC, NTA, NLM, SIZE2, RS2xC
			Atributo	Tamaño
Indicador	RCI, RAI, RADI, RMI	Satisface	Indicador	ICXC
			Necesidad de Información	Conocer la estabilidad de los requisitos

Tabla 7-2. Modelo de medición para el dominio de “Requisitos” representados en una notación textual

Medidas en el Modelo de medición				
Medida Base	**Descripción**	**Método de medición**	**Escala**	**Unidad**
NCR	Número de requisitos anulados	Contar el número de requisitos cancelados (state=cancelled)	Entero	Requisito
NAR	Número de requisitos aprobados	Contar el número de requisitos aprobados (state=approved)	Entero	Requisito
NADR	Número de requisitos añadidos	Contar el número de requisitos añadidos (state=added)	Entero	Requisito
NMR	Número de requisitos modificados	Contar el número de requisitos modificados (state=modified)	Entero	Requisito
TOR	Total de requisitos	Contar el número total de requisitos	Entero	Requisito
Medida Derivada	**Descripción**	**Función de Cálculo**	**Escala**	**Unidad**
IR	Número de requisitos inestables	Contar el número de requisitos modificados y anulados. IR = NCR+NMR	Entero	Requisito

TIR[25]	Número total de requisitos inestables	Contar el número total requisitos inestables, es decir modificados, añadidos e inestables TIR= IR+NADR		Entero	Requisito
Indicador	**Descripción**	**Modelo de Análisis**	**Criterio de Decisión**	**Escala**	**Unidad**
RAI	Grado de aprobación de requisitos	NAR/TOR	Si 0 ≤ NAR/TOR < 0.30 → RAI='Muy Malo' Si 0.30 ≤ NAR/TOR < 0.60 → RAI='Malo' Si 0.60 ≤ NAR/TOR < 0.90 → RAI='Regular' Si NAR/TOR > 0.90 → RAI='Bueno'	Enumeración ('Bueno', 'Malo', 'Regular', 'Muy Malo')	Etiqueta de texto
RCI	Grado de anulación de requisitos	NCR/TOR	Si 0 ≤ NCR/TOR < 0.15 → RCI= 'Bueno' Si 0.15 ≤ NCR/TOR < 0.30 → RCI= 'Regular' Si 0.30 ≤ NCR/TOR < 0.60 → RCI= 'Malo' Si NCR/TOR > 0.60 →RCI='Muy malo'	Enumeración ('Bueno', 'Malo', 'Regular', 'Muy Malo')	Etiqueta de texto
RADI	Grado de adición de requisitos	NADR/TOR	Si 0 ≤ NADR/TOR < 0.30 →RADI='Bueno' Si 0.30 ≤ NADR/TOR < 0.60 →RADI='Regular' Si 0.60 ≤ NADR/TOR < 0.90 →RADI='Malo' Si NADR/TOR > 0.90 → RADI= 'Muy Malo'	Enumeración ('Bueno', 'Malo', 'Regular', 'Muy Malo')	Etiqueta de texto
RMI	Grado de modificación de requisitos	NMR/TOR	Si 0 ≤ NMR/TOR< 0.25 → RMI= 'Bueno' Si 0.25 ≤ NMR/TOR< 0.55 → RMI= 'Regular' Si 0.55 ≤ NMR/TOR< 0.85 → RMI= 'Malo' Si NMR/TOR> 0.85 →RMI= 'Muy Malo'	Enumeración ('Bueno', 'Malo', 'Regular', 'Muy Malo')	Etiqueta de texto
RS	Estabilidad de los requisitos	TIR/TOR	Si 0 ≤ TIR/TOR < 0.50 → RS= 'Bueno' Si 0.50 ≤ TIR/TOR < 0.60 → RS= 'Regular' Si 0.60 ≤ TIR/TOR > 0.70 → RS= 'Malo' Si TIR/TOR > 0.80 →RS= 'Muy Malo'	Enumeración ('Bueno', 'Malo', 'Regular', 'Muy Malo')	Etiqueta de texto

Tabla 7-3. Modelo de medición para el dominio de "Requisitos" representados en una notación textual (cont.)

A continuación se presentan los modelos de medición definidos a partir de la especificación textual. Debido a las dimensiones de los modelos, se ha divido el modelo representado con SMML-Editor en varias figuras.

[25] TIR=NCR+NMR+NADR, pero como SMML no permite operar con más de 2 medidas se tiene que descomponer en TIR = IR + NADR, donde IR = NCR + NMR.

En primer lugar la Figura 7-5 muestra los elementos que contienen información general sobre las necesidades de medición, es decir los elementos "Tamaño", "Necesidad de Información", "Entidad", "Concepto Medible" y "Modelo de Calidad"

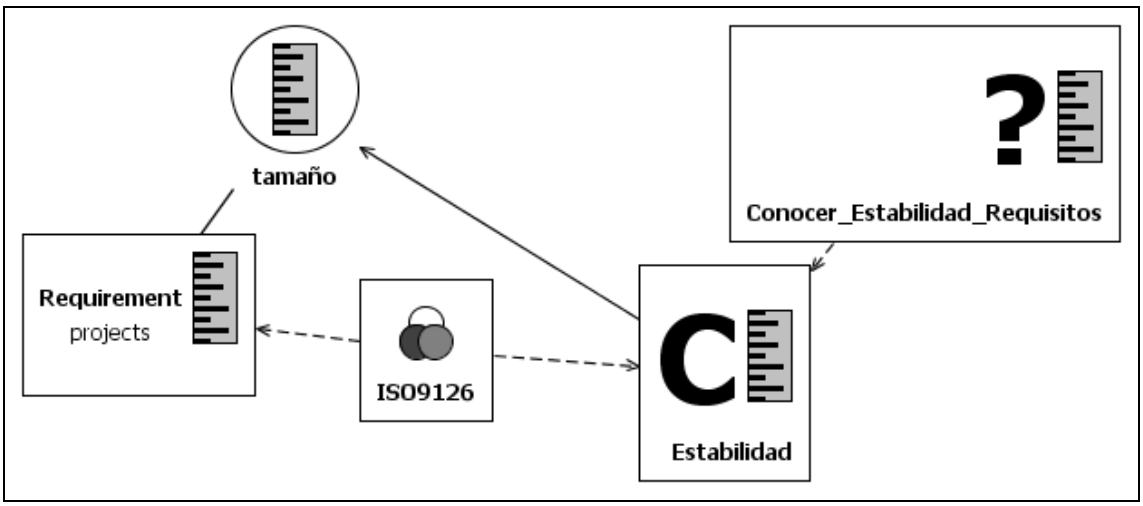


Figura 7-5. Representación gráfica de las instancias de Necesidad de Información, Concepto Medible, Entidad de Dominio, Atributo y Modelo de Calidad.

En la Figura 7-7, Figura 7-6, Figura 7-8, Figura 7-9 y Figura 7-10 se muestran las medidas (base y derivadas) con sus formas de medir asociados (métodos y funciones de medición) para calcular los indicadores RAI, RCI, RADI, RMI y RS respectivamente. Como se observa en las figuras se han añadido métodos de medición parametrizados para indicar el estado de los requisitos (*added, approved, cancelled* y *modified*) y la versión que se está midiendo (v01.00.01).

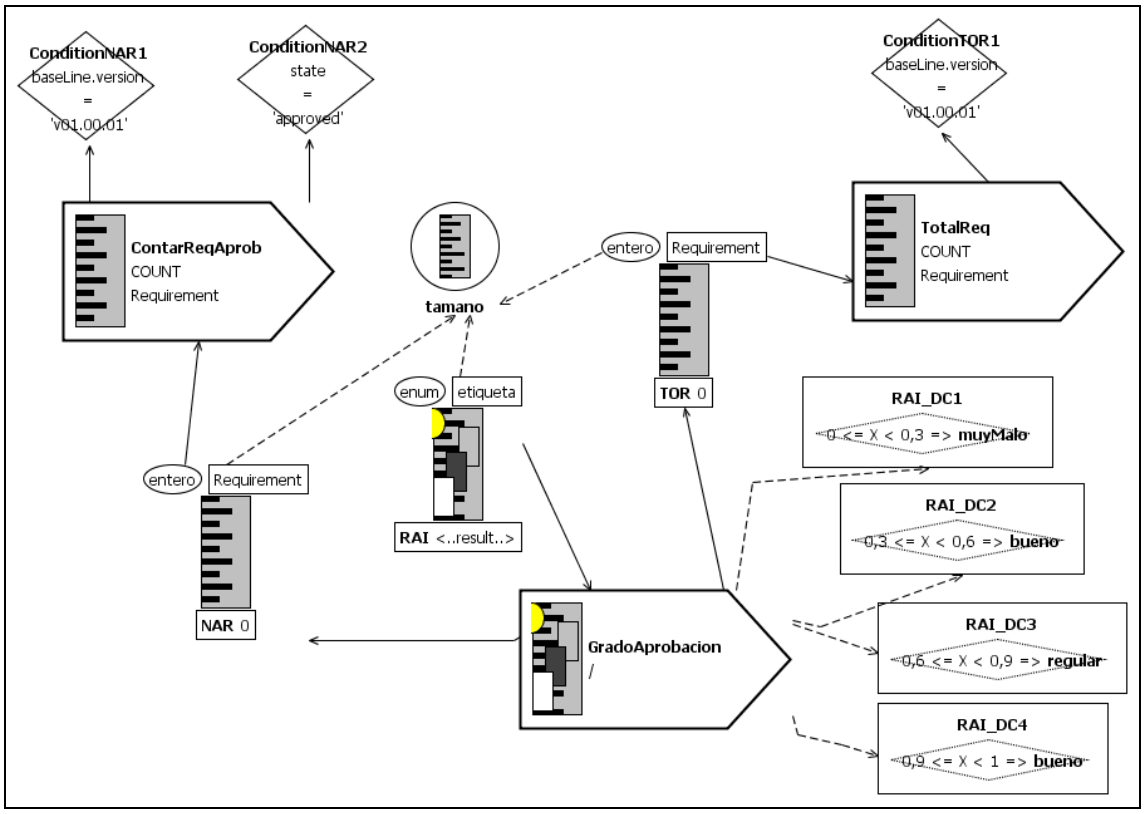


Figura 7-6. Representación gráfica para definir el indicador RAI

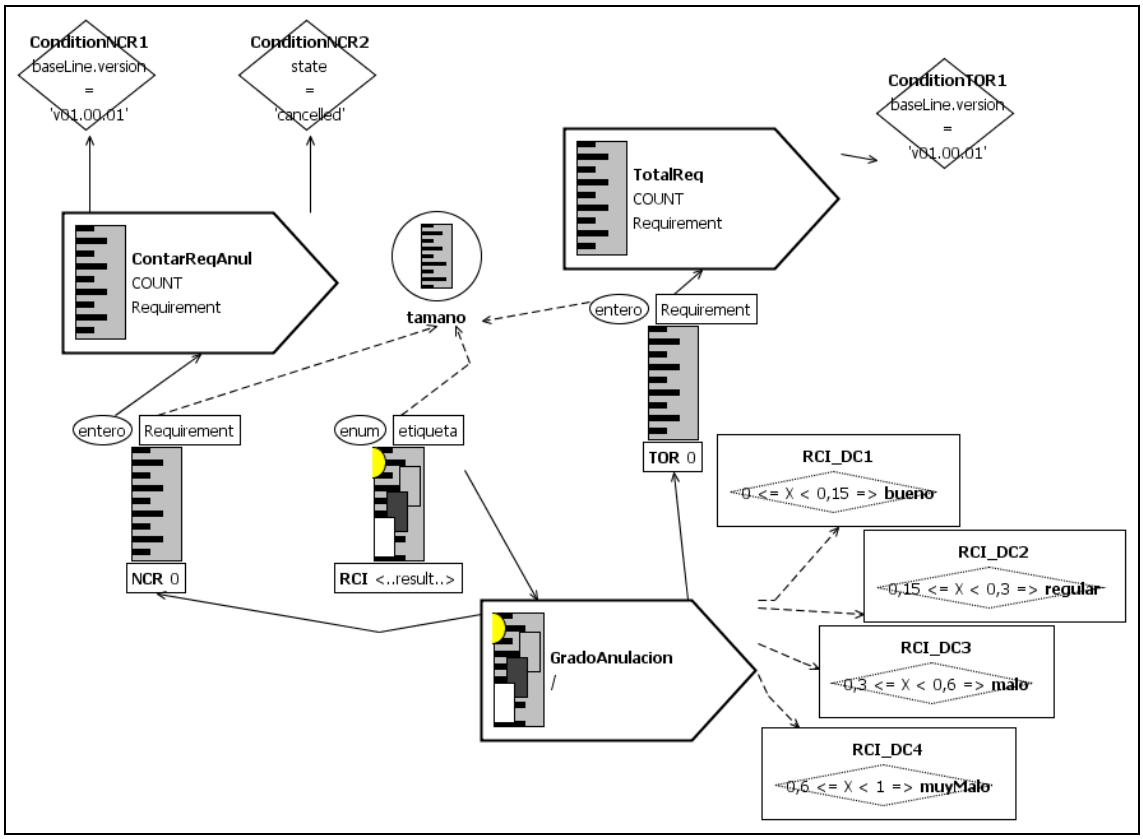


Figura 7-7. Representación gráfica para definir el indicador RCI

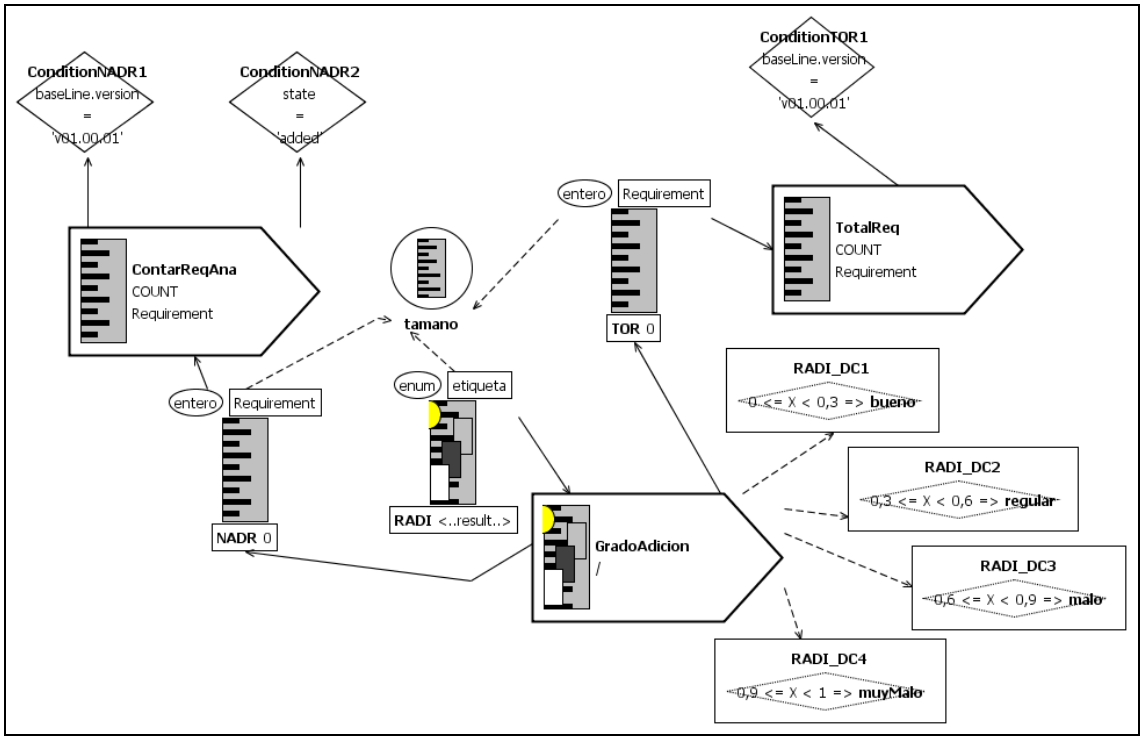


Figura 7-8. Representación gráfica para definir el indicador RADI

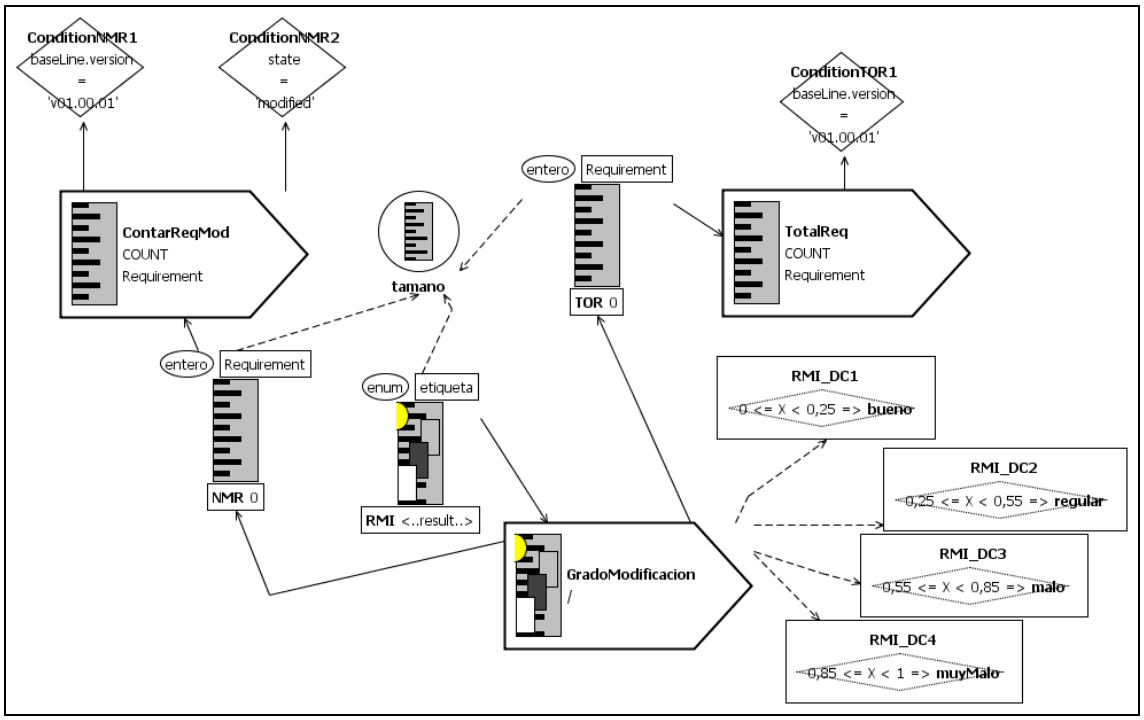


Figura 7-9. Representación gráfica para definir el indicador RMI

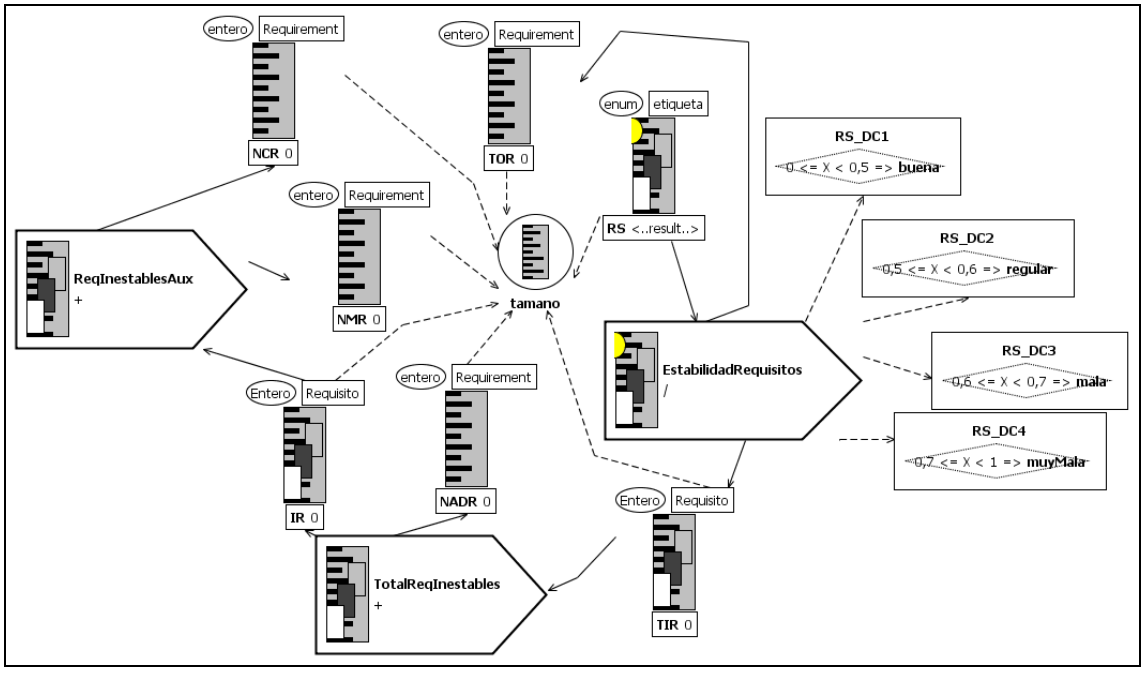


Figura 7-10. Representación gráfica para definir el indicador RS

7.4.2. Medición de Diagramas de Clases UML.

El segundo dominio que se evaluó fue “Diagramas de clases de diseño UML” del módulo ISURG del proyecto IndraSalud dentro de la fase de “Diseño del Sistema”.

Uno de los hitos que debe cumplir la empresa dentro del marco de calidad de Indra es la elaboración de diseños funcionales. Por ello, conocer la mantenibilidad (facilidad de mantenimiento) de los diagramas UML2 se convierte en un punto importante.

En los siguientes sub-apartados se muestran en detalle las etapas llevadas a cabo para aplicar la medición genérica en el dominio de diagramas de clases UML.

7.4.2.1. Incorporación del Metamodelo de dominio.

Para aplicar la medición del software se necesita en primer lugar el metamodelo de dominio que caracteriza el tipo de entidad que se va a medir. Este metamodelo (Figura 7-11) se descargó de la página pública Zoos[26] pero fue adaptado al caso de estudio, es decir, únicamente se representaron los elementos significativos para satisfacer las necesidades de medición para facilitar la comprensión y gestión del mismo. Debido a su extensión, en la Figura 7-11 se muestra un fragmento del metamodelo.

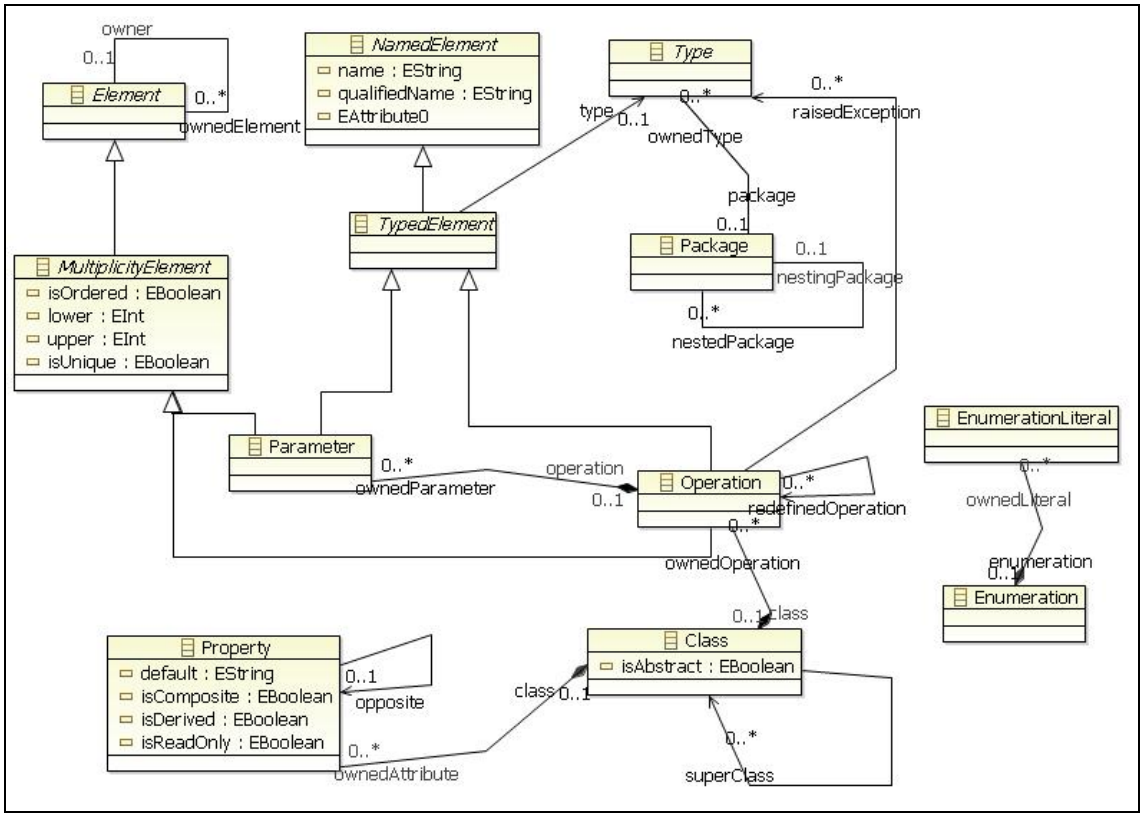


Figura 7-11. Fragmento del Metamodelo de Diagramas UML.

En segundo lugar se definió el modelo de diagrama de clases UML a partir del metamodelo representado en la Figura 7-11. Este modelo representa el diagrama de clases utilizado para diseño del modulo ISURG de IndraSalud, y está formado por las clases de dominio de la aplicación.

[26] Disponible en: http://www.emn.fr/z-info/atlanmod/index.php/Zoos. En esta página existe material que se puede utilizar en el dominio de MDE, como por ejemplo una lista extensa de metamodelos de multitude de dominios representados en ECORE.

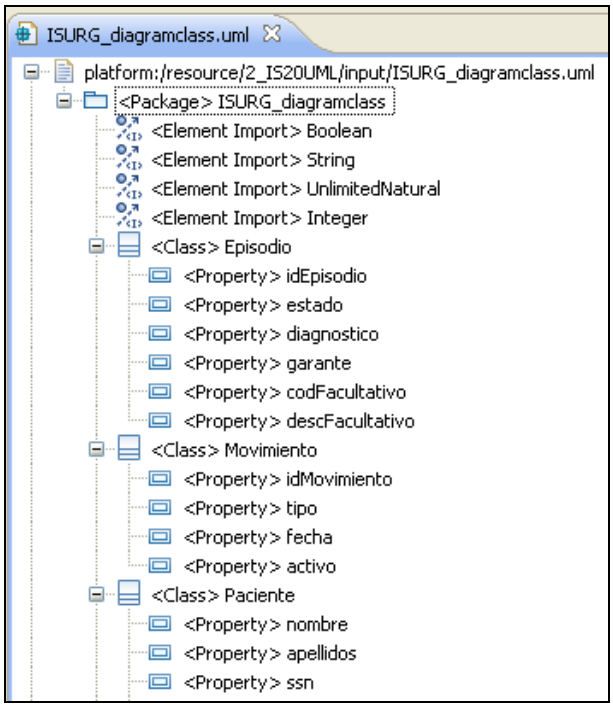


Figura 7-12. Fragmento del listado de elementos del modelo de Diagramas UML del módulo ISURG.

7.4.2.2. Definición del Modelo de Medición.

En tercer lugar se definieron las medidas necesarias para la medición en el dominio de diagramas de clases UML. Estas medidas se adoptaron de las propuestas en (Genero et al., 2005b) y se presentan de forma textual en la Tabla 7-4 y Tabla 7-5.

Con el objetivo de homogeneizar los nombres de las medidas, algunos nombres originales propuestos por Genero et al. (2005) se han cambiado (los nombres originales aparecen entre paréntesis).

Elementos del metamodelo vs Instancias del Modelo				
Elementos del Metamodelo (clases MOF)	**Elementos del Metamodelo (clases MOF)**	**Elementos del Metamodelo (clases MOF)**		**Elementos del Metamodelo (clases MOF)**
Atributo	Tamaño	Tiene	Entidad	Uml2
			Atributo	Tamaño
Entidad	Uml2	Definida por	Modelo de Calidad	ISO 9126
			Entidad	Uml2
Concepto medible	Mantenibilidad	Evalúa	Modelo de Calidad	ISO 9126
			Concepto medible	Mantenibilidad
Modelo de Calidad	ISO 9126	Relaciona	Atributo	Tamaño
			Concepto medible	Mantenibilidad
Necesidad de Información	Conocer la mantenibilidad de un diagrama de clases UML	Está asociado con	Concepto Medible	Mantenibilidad
			Necesidad de Información	Conocer la mantenibilidad de un diagrama de clases UML
Medida	NC, NTA, NLM, SIZE2, RS2xC	Definida por	Medida	NC, NTA, NLM, SIZE2, RS2xC
			Atributo	Tamaño

Indicador	ICXC	Satisface	Indicador	ICXC
			Necesidad de Información	Conocer la mantenibilidad de un diagrama de clases UML

Tabla 7-4. Modelo de medición para el dominio de “Diagramas UML” representados en una notación textual

Medidas del Modelo de Medición					
Medida Base	**Descripción**	**Método de medición**		**Escala**	**Unidad**
NC (TC)	Número total de clases en el diagrama	Contar el número de clases en el diagrama		Entero	Unidad
NTA (DAC)	Número total de atributos en el diagrama cuyo tipo es otra clase.	Contar el número de atributos en el diagrama cuyo tipo es otra clase		Entero	Unidad
NLM (NOM)	Número total de métodos locales en el diagrama	Contar el número total de métodos locales en el diagrama		Entero	Unidad
Medida Derivada	**Descripción**	**Measurement Function**		**Escala**	**Unidad**
SIZE2	Número total de atributos en el diagrama cuyo tipo son clase más el número de métodos locales en el diagrama	SIZE2 = NTA+NLM		Entero	Unidad
RS2xC	Ratio de atributos de tipo clase y métodos locales por clase	RS2xC = SIZE2/NC		Real Positivo	Ratio
Indicador	**Descripción**	**Medida Previa**	**Criterio de Decisión**	**Escala**	**Unidad**
ICxC	Índice de la media de complejidad de clases en el diagrama	RS2xC	Si RS2xC > 50 → ICxC =‘Alto’ Si 25 < RS2xC < 50 → ICxC = ‘Normal’ Si 10 < RS2xC < 25 → ICxC = ‘Bajo’ Si RS2xC ≤ 10 → ICxC = ‘Muy Bajo’	Enumeración (‘Muy Bajo’, ‘Bajo’, ‘Normal’, ‘Alto’)	Etiqueta de texto

Tabla 7-5. Modelo de medición para el dominio de “Diagramas UML” representados en una notación textual (cont.)

A partir de las medidas definidas en las tablas anteriores se ha definido mediante SMML-Editor el diagrama de la Figura 7-13.

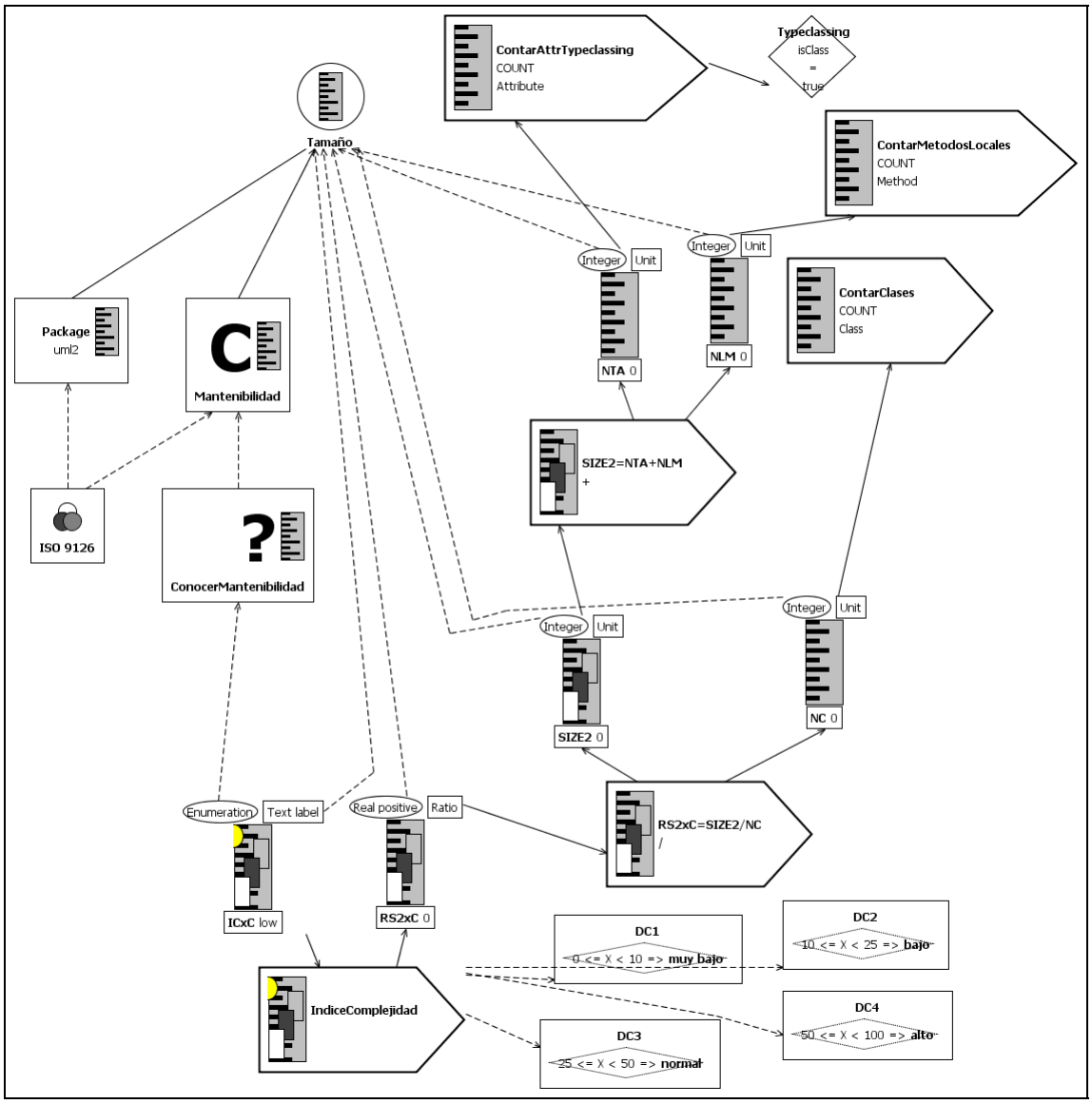


Figura 7-13. Representación gráfica del modelo de medición para el dominio “Diagrama de Clases UML”

7.4.3. Medición de Diagramas Bases de Datos Relacionales.

El tercer dominio que se evaluó fue el dominio de “Bases de datos Relacionales” del módulo ISFAR del proyecto IndraSalud dentro de la fase de diseño.

Uno de los principales objetivos de negocio de la compañía Indra es dar soporte a sus bases de datos proporcionando los medios necesarios para facilitar la mejora (y por consiguiente el mantenimiento) de los modelos conceptuales y lógicos de las bases de datos. Para conseguirlo, Indra necesita saber la mantenibilidad (facilidad de mantenimiento) de sus esquemas conceptuales de bases de datos, representados en notación E/R, y de sus esquemas relacionales.

En los siguientes sub-apartados se muestran en detalle las etapas llevadas a cabo para aplicar la medición genérica en el dominio de esquemas de bases de datos relacionales.

7.4.3.1. Incorporación del metamodelo de dominio.

Para aplicar la medición del software se necesita en primer lugar el metamodelo de dominio. La Figura 7-14 muestra el metamodelo de esquemas de bases de datos relacionales (relational.ecore), que es el metamodelo que caracteriza el tipo de entidad que se van a medir.

Este metamodelo se definió con las entidades necesarias para satisfacer las necesidades de medición.

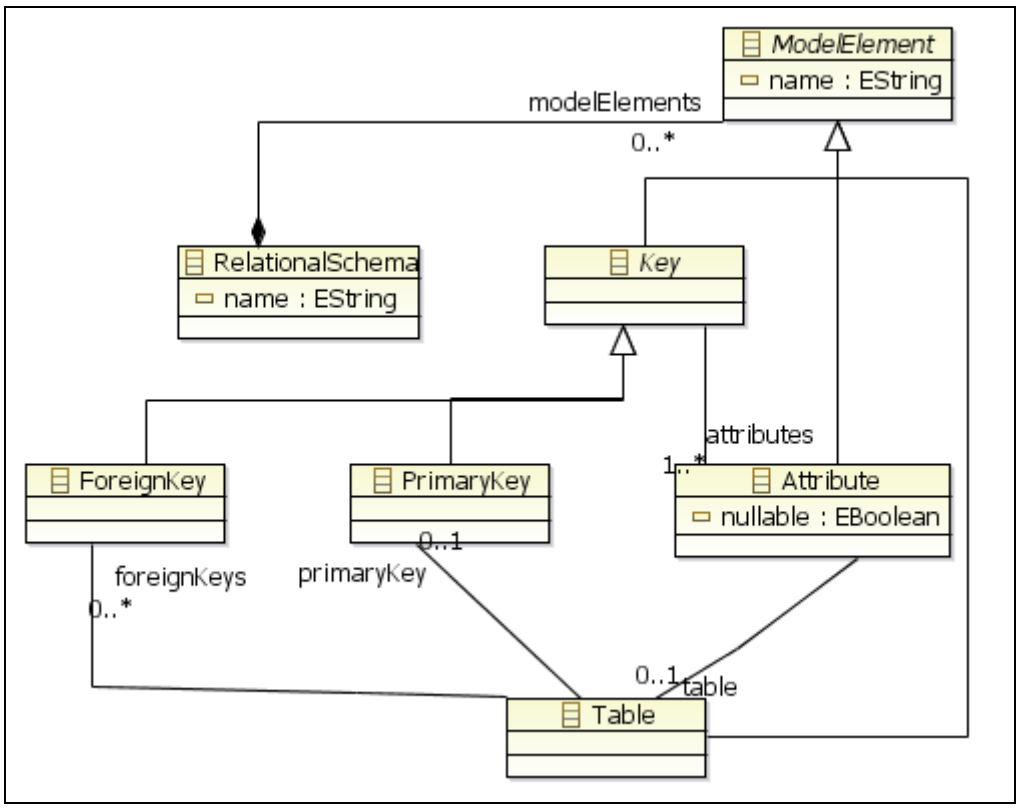


Figura 7-14. Metamodelo de esquemas relacionales

En segundo lugar se definió el modelo de esquema relacional a partir del metamodelo representado en la Figura 7-15. Este modelo representa el esquema relacional utilizados para el diseño del modulo ISFAR de IndraSalud.

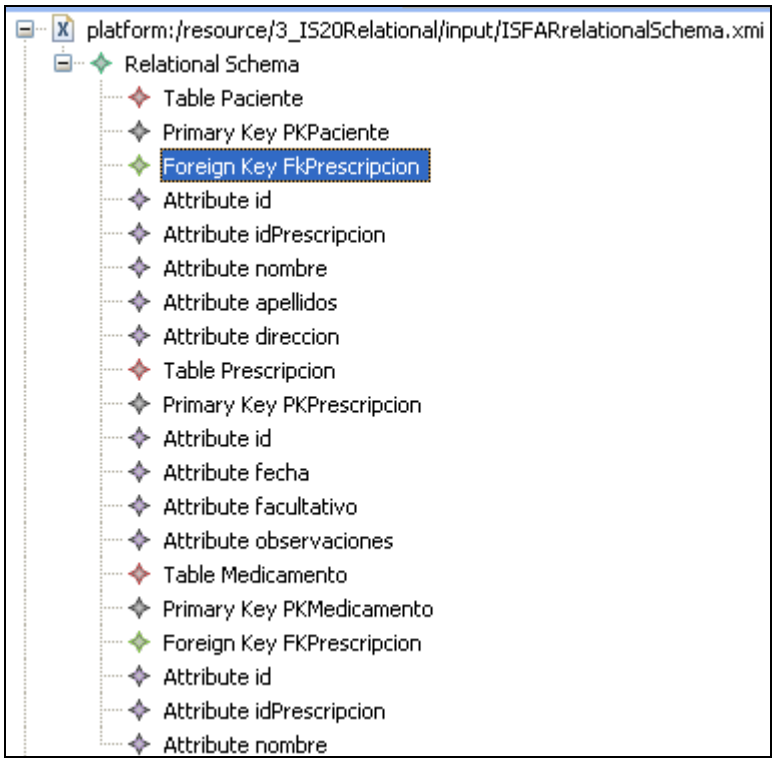


Figura 7-15. Fragmento del listado de elementos del modelo de Diagramas de bases de datos del módulo ISFAR.

7.4.3.2. Definición del modelo de medición.

En tercer lugar se definieron las medidas para llevar a cabo esta medición, estas medidas fueron tomadas de Calero et al. (Calero et al., 2001) y se presentan en la Tabla 7-6 y Tabla 7-7.

Elementos del metamodelo vs Instancias del Modelo				
Elementos del Metamodelo (clases MOF)	**Elementos del Metamodelo (clases MOF)**	**Elementos del Metamodelo (clases MOF)**		**Elementos del Metamodelo (clases MOF)**
Atributo	Tamaño	Tiene	Entidad	Uml2
			Atributo	Tamaño
Entidad	Uml2	Definida por	Modelo de Calidad	ISO 9126
			Entidad	Uml2
Concepto medible	Mantenibilidad	Evalúa	Modelo de Calidad	ISO 9126
			Concepto medible	Mantenibilidad
Modelo de Calidad	ISO 9126	Relaciona	Atributo	Tamaño
			Concepto medible	Mantenibilidad
Necesidad de Información	Conocer la mantenibilidad de un esquema de bases de datos relacional	Está asociado con	Concepto Medible	Mantenibilidad
			Necesidad de Información	Conocer la mantenibilidad de un esquema de bases de datos relacional
Medida	NC, NTA, NLM, SIZE2, RS2xC	Definida por	Medida	NC, NTA, NLM, SIZE2, RS2xC
			Atributo	Tamaño

Indicador	ICXC	Satisface	Indicador	ICXC
			Necesidad de Información	Conocer la mantenibilidad de un esquema de bases de datos relacional

Tabla 7-6. Modelo de medición para el dominio de “Esquema de bases de datos relacionales” representados en una notación textual

Medidas del Modelo de Medición					
Medida Base	**Descripción**	**Método de medición**		**Escala**	**Unidad**
NT	Número total de tablas en el esquema	Contar el número total de tablas en el esquema		Entero	Unidad
NA	Número total de atributos en el esquema	Contar el número total de atributos en el esquema		Entero	Unidad
NFK	Número total de Foreign Key en el esquema	Contar el número total de Foreign Key en el esquema		Entero	Unidad
Indicador	**Descripción**	**Modelo de Análisis**	**Criterio de Decisión**	**Escala**	**Unidad**
TMI	Índice de mantenimiento de tablas	NA/NT	Si NA/NT ≥ 18 → TMI =‘Muy alto’ Si 12 ≤ NA/NT < 18 → TMI = ‘Alto’ Si 6≤ NA/NT < 12 → TMI = ‘Normal’ Si 0≤ NA/NT < 6 → TMI = ‘Bajo’	Enumeración (‘Bajo’, ‘Normal’, ‘Alto’, ‘Muy Alto’)	Etiqueta de texto
SCI	Índice de conectividad en el esquema	NFK/NT	Si NFK/NT ≥ 2 → SCI =‘Muy alto’ Si 1,5 ≤ NFK/NT < 2 → SCI = ‘Alto’ Si 1≤ NFK/NT < 1,5 → SCI = ‘Normal’ Si 0 ≤ NFK/NT < 1 → SCI = ‘Bajo’	Enumeración (‘Bajo’, ‘Normal’, ‘Alto’, ‘Muy Alto’)	Etiqueta de texto

Tabla 7-7. Modelo de medición para el dominio de “Esquema de bases de datos relacionales” representados en una notación textual (cont.)

A partir de las medidas definidas en las tablas anteriores se ha definido el modelo de medición gráfico mediante SMML-Editor (Figura 7-16).

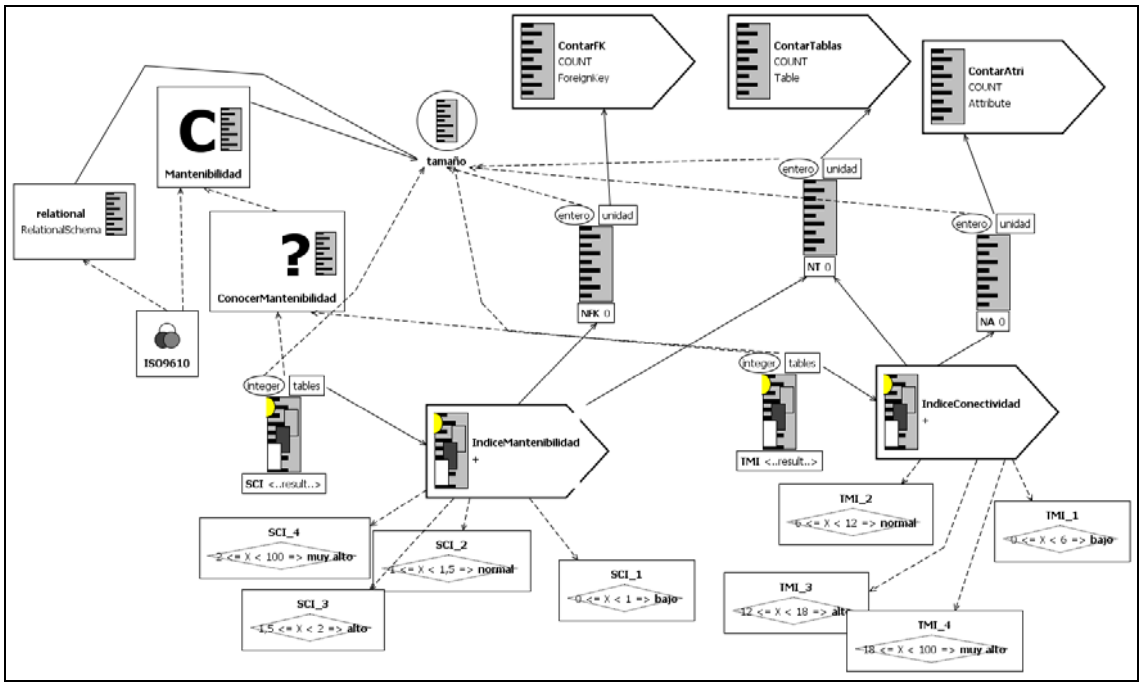


Figura 7-16. Representación gráfica del modelo de medición para el dominio “Esquema de bases de datos relacionales”

7.4.4. Ejecución de la Medición.

Una vez que el analista de la medición dispuso de los metamodelos de dominio y los modelos de dominio y medición, los responsables de los módulos llevaron a cabo la ejecución de la medición usando SMTool siguiendo los pasos del método de SMF:.

1. **Creación del proyecto SMTool.** Cada responsable se creó un proyecto llamado *“Calidad_IndraSalud20_NOMBRE_MODULO”*. Que contenía toda la información necesaria para gestionar las mediciones propuestas en cada dominio.
2. **Incorporación de los metamodelos de dominio.** Cada responsable incorporó al proyecto los metamodelos de cada dominio (véase Figura 7-3, Figura 7-11 y Figura 7-14). Los metamodelos definidos con ECORE para cada dominio eran los siguientes:
   - Requirements.ecore
   - UMLDiagramClass.ecore
   - Relational.ecore
3. **Definición de los modelos de dominio.** Cada responsable incorporó al proyecto los modelos de dominio (véase Figura 7-4, Figura 7-12 y Figura 7-15) previamente definidos por el analista de la medición. Estaban definidos en xmi y eran los siguientes:
   - Requirements.xmi
   - UMLDiagramClass.xmi
   - Relational.xmi
4. **Creación de los modelos de medición.** Cada responsable incorporó al proyecto los modelos de medición previamente definidos por el analista de la medición (véanse apartados 7.4.1.2, 7.4.2.2 y 7.4.3.2) y eran los siguientes:
   - Requirements.smm

- UMLDiagramClass.smm
- Relational.smm

5. **Ejecución de la Medición.** Una vez incorporados al proyecto todos los elementos de entrada de cada medición crearon una instancia de medición del software en SMTool ("Measurement Specification") para cada dominio (véase Figura 7-17), que se denominaron del siguiente modo:
   - Requirements measurement
   - UMLDiagramClass measurement
   - Relational measurement

En cada instancia de medición se seleccionaron los elementos que ya estaban previamente definidos e incorporados al proyecto:

- **Metamodelo y modelos de entrada:** rutas donde están almacenados los ficheros del metamodelo de dominio, el modelo de dominio y el modelo de medición.
- **Ruta de Transformación:** ruta del directorio donde se almacenará el fichero de especificación qvt.
- **Modelo de Salida de Medición:** ruta y nombre del modelo de medición con los resultados incluidos. Este modelo es la extensión del modelo de medición de entrada.

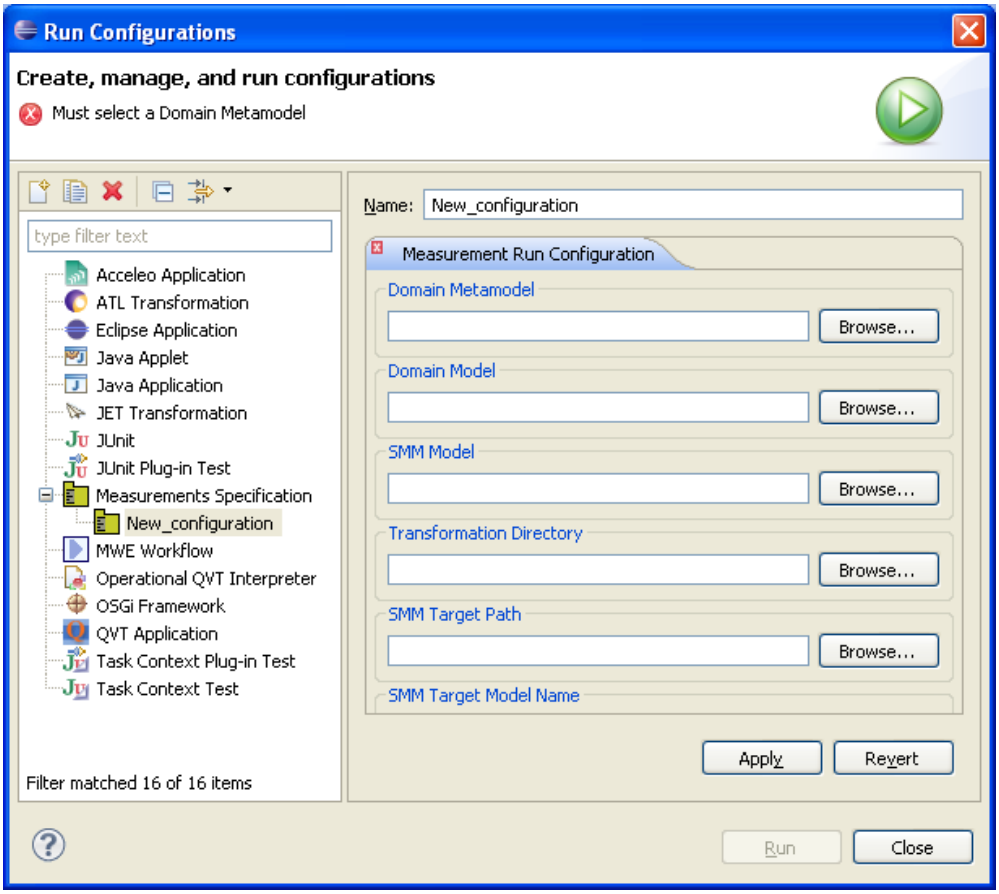


Figura 7-17. Formulario de "Measurement Specification" de SMTool

Una vez definida cada instancia de medición los responsables de los módulos procedieron a su ejecución obteniendo los resultados de medición presentados en las Tabla 7-8, Tabla 7-9, Tabla 7-10, Tabla 7-11, Tabla 7-12 y Tabla 7-13; correspondientes a los dominios de requisitos, diagramas de clases UML y esquemas de bases de datos relacionales respectivamente

Medida	Valor
NAR Número de requisitos aprobados	95
NCR Número de requisitos anulados	8
NADR Número de requisitos añadidos	12
NMR Número de requisitos modificados	28
TOR Total de requisitos	143
IR Número de requisitos inestables IR = NCR+NMR	IR = 8 + 28 =36
TIR Número total de requisitos inestables TIR= IR+NADR	TIR = 36+12 = 48

Tabla 7-8. Resultados de las medidas base y derivadas en el dominio de Requisitos

Indicador	Análisis de Medición	Resultado Indicador
RAI Grado de aprobación de requisitos NAR/TOR	95/143= 0.66	Regular
RCI Grado de anulación de requisitos NCR/TOR	8/143= 0.05	Bueno
RADI Grado de adición de requisitos NADR/TOR	12/143= 0.06	Bueno
RMI Grado de modificación de requisitos NMR/TOR	28/143= 0.19	Regular
RS Estabilidad de los requisitos TIR/TOR	48/143=0.33	Buena

Tabla 7-9. Resultados de los indicadores en el dominio de Requisitos

Medida	Valor
NC Número total de clases en el diagrama	20
NTA Número total de atributos en el diagrama cuyo tipo es otra clase	35
NLM Número total de métodos locales en el diagrama	220
SIZE2 SIZE2 = NTA+NLM	SIZE2=35+220=255
RS2xC RS2xC = SIZE2/NC	RS2xC=255/20=12.75

Tabla 7-10. Resultados de las medidas base y derivadas en el dominio de diagramas UML

Indicador	Análisis de Medición	Resultado Indicador
ICxC Índice de la media de complejidad de clases en el diagrama de clases RS2xC	195/20=9.75	Bajo

Tabla 7-11. Resultados de la medición en el dominio de diagramas UML

Medida	Valor
NT Número total de tablas en el esquema	17
NA Número total de atributos en el esquema	91
NFK Número total de Foreign Key en el esquema	6

Tabla 7-12. Resultados de las medidas base y derivadas en el dominio de bases de datos relacionales

Medida	Análisis Model	Resultado Indicador
TMI Índice de mantenimiento de tablas NA/NT	91/17 = 5.35	Bajo
SCI Índice de conectividad en el esquema NFK/NT	6/17 = 0.35	Bajo

Tabla 7-13. Resultados de los indicadores en el dominio de bases de datos relacionales

Como se puede observar, el resultado final de la estabilidad de requisitos para el sistema ISHOS v01.00.01 es “Buena” (Tabla 7-9), el resultado final de la complejidad en el diagrama de Clases es “Bajo” (Tabla 1-12) y el resultado final del índice de mantenimiento de tablas y de la conectividad del esquema es Bajo (Tabla -14).

7.5. Lecciones Aprendidas.

Las lecciones aprendidas en la aplicación del marco de trabajo a este caso de estudio se pueden abordar desde dos puntos de vista:

- **Punto de Vista de la Empresa:** Las principales aportaciones para la empresa expuestas por los responsables de los módulo ISHOS, ISURG e ISFAR y expuestas en los cuestionarios, han sido las siguientes:
  - Medición
    - Se les proporciona un marco de trabajo para llevar a cabo mediciones en cualquier dominio. Para conseguirlo deben disponer del metamodelo y modelo de dominio, y el modelo de medición del software.
    - Posibilidad de reutilizar los modelos de medición. Los modelos de medición definidos para un dominio concreto, pueden reutilizarse para otros proyectos que tengan las mismas necesidades de medición.
    - Hasta el momento la forma de medición de la empresa se basaba en definir indicadores aislados para evaluar la calidad y productividad de un proyecto (requisitos, cobertura de pruebas, líneas de código, etc). Con el modelado de las mediciones, además de poder representar los indicadores, se pueden representar las necesidades de información que satisfacen, así como todos los elementos que hay que definir claramente hasta llegar a obtener los indicadores (entidades, atributos, conceptos medibles, medidas base, medidas derivadas, métodos de medición, funciones de medición, modelos de análisis, etc.).
    - Se ha aportado el metamodelo de dominio y modelo de medición necesario para llevar a cabo mediciones en el dominio de Requisitos, Diagramas de Clases UML y Esquemas de bbdd relacionales.
  - Entorno Tecnológico:
    - Se ha proporcionado un entorno tecnológico basado en Eclipse que permite por un lado la gestión del modelado: definición de metamodelos y modelo de dominio y de medición; y por otro lado la ejecución de las mediciones definidas.
    - Se ha proporcionado un lenguaje específico de dominio que permite definir los modelos de medición de una forma gráfica.
    - Al estar basado en eclipse, el entorno puede extenderse con nuevas funcionalidades, como por ejemplo la inclusión de nuevos lenguajes específicos de dominio para la definición de modelos de dominio, o dar al entorno la posibilidad de utilizar un repositorio de gestión de mediciones con CVS o SVN.
    - El entorno desarrollado homogeneiza el proceso de medición del software ya que únicamente es necesaria una herramienta para llevarlo a cabo. Hasta ahora existían tres herramientas diferentes para definir los modelos de los distintos dominios: Caliber RM para los requisitos, Borland Together para los diagramas de clases y Power Designer para el esquema de BBDD. Aunque algunas de estas herramientas permiten calcular medidas predeterminadas[27] no permiten definir nuevas medidas ni indicadores sobre los modelos definidos. En la Tabla

[27] Todas las medidas predeterminadas son medidas base calculadas a partir del método contar.

7-14 se muestra las posibilidades, desde el punto de vista de la medición, que ofrece cada una de las herramientas. Como se observa en la Tabla 7-14, SMTool cubre todas las necesidades de las herramientas anteriores y además aporta nuevos beneficios en el campo de la medición como: reutilización de modelos de medición, definición y ejecución automática de modelos de medición y representación gráfica de los modelos de medición.

Herramienta	Modelos Definidos	Medidas predeterminadas	Indicadores	Definición Medidas
Caliber RM	Requisitos	SI	NO	NO
Borland Together	Diagramas UML	NO	NO	NO
Power Designer	Esquema de BBDD	SI	NO	NO
SMTool	Cualquier dominio [28]	SI	SI	SI

Tabla 7-14. Resumen de las características de las herramientas.

- **Punto de Vista de la Investigación.** La aplicación del marco de trabajo en Indra ha permitido poder llevar a cabo con éxito el método de trabajo "Investigación en Acción" (ver Capítulo 2). Ello ha facilitado la validación y refinamiento de la propuesta, lo que ha permitido confirmar la utilidad de:
  - Tener un marco de trabajo basado en MDE.
  - Poder definir modelos de medición homogéneos.
  - Disponer de un lenguaje visual para la representación de modelos de medición.
  - Seguir un método de medición para llevar a cabo las mediciones de software.
  - Transformar modelos de medición para obtener los resultados de los modelos de medición.
  - Disponer de herramientas adecuadas para automatizar y gestionar de forma efectiva los procesos de medición del software. La aplicación de SMF en la empresa nos ha permitido realizar importantes refinamientos en las herramientas.

Además, la aplicación práctica de SMF nos ha aportado numerosas ideas y consideraciones para el futuro de la investigación en esta materia que se detallan en el Capítulo 8. Por otro lado, el éxito en la primera aplicación real de SMF ha motivado la necesidad de hacer una réplica del caso de estudio en más dominios de medición dentro de la misma empresa (tales como procesos de negocio, cobertura de pruebas o diagramas de casos de uso) y, por supuesto, en otras empresas, como PYMES.

[28] Siempre que se disponga de su metamodelo de dominio.

“La calidad nunca es un accidente; siempre es el resultado de un esfuerzo de la inteligencia” (John Ruskin)

8. Conclusiones y Trabajo Futuro.

En este capítulo final se realiza una valoración global de los resultados del trabajo llevado a cabo en esta tesis. En primer lugar, se realiza un análisis de la consecución de objetivos y de los principales resultados obtenidos para, a continuación, presentar las publicaciones que avalan los resultados de la tesis y establecer las líneas de trabajo abiertas en la investigación.

8.1. Análisis de la Consecución de Objetivos.

Hoy en día, la medición del software se ha convertido en uno de los objetivos estratégicos fundamentales en las organizaciones a la hora de promover la calidad de sus productos software. Para poder implantar un plan efectivo de medición, que pueda abarcar cualquier tipo de entidad software, es muy importante establecer un buen marco de trabajo que permita definir modelos de medición genéricos y ejecutarlos en un dominio determinado.

En la presente tesis doctoral se ha propuesto un marco de trabajo para dar soporte a la medición genérica del software. También se ha presentado un conjunto de métodos de medición genéricos que sirven para gran cantidad de medidas. Además, se han desarrollado y presentado un lenguaje de dominio específico que permite definir los modelos de medición de manera gráfica y una herramienta que automatiza el proceso de medición genérica.

En el primer capítulo se expuso el siguiente objetivo global de la tesis:

Definir un marco de trabajo basado en MDE para la medición genérica del software

A continuación se presenta una valoración de la consecución de los diferentes objetivos parciales indicados en el Capítulo 1:

- **Objetivo 1.** Adaptación del Metamodelo para la Medición del Software basado en la ontología de la medición del software.

El Metamodelo de Medición del Software se adaptó (indicar donde se incluye dicha adaptación) para dar soporte a las mediciones genéricas y nuevas formas de medir genéricas presentadas en el Capítulo 4. El metamodelo de medición también se adaptó (decir donde se explica esta adaptación) con el objetivo de poder definir el lenguaje gráfico SMML y que los modelos de medición genéricos expresados con dicho lenguaje pueden ser ejecutados (obtención automática de los resultados de las medidas).

- **Objetivo 2.** Definición de formas de medición genéricas que permiten obtener medidas independientes del dominio.

Se definieron cinco métodos de medición genéricos y tres funciones de cálculo genéricas (presentado en el Capítulo 4).

- **Objetivo 3.** Definición de la sintaxis y la semántica de un DSL gráfico para representar modelos de medición software.

Se elaboró el lenguaje gráfico SMML para definir modelos de medición del software a partir del metamodelo adaptado del objetivo 1. Este lenguaje SMML, presentado en el Capítulo 5, se especificó con una semántica y sintaxis abstracta y concreta.

- **Objetivo 4.** Elaboración de una herramienta para la creación de SMMs mediante instanciación del metamodelo definido en el Objetivo 1 y usando, indistintamente, los lenguajes definidos en el Objetivo 2.

Se desarrolló SMTool (presentado en el Capítulo 5) para dar soporte a las mediciones genéricas. SMTool permite ejecutar mediciones genéricas a partir de un modelo de medición aplicado sobre un modelo de dominio. Tiene además integrado un editor de SMML (SMML-Editor) que permite definir modelos de medición de manera gráfica que son entrada del proceso de medición genérica. .

- **Objetivo 5.** Validación empírica del lenguaje.

Se realizó un experimento para validar la usabilidad y mantenibilidad de SMML basándose en la ISO 9126. El desarrollo y conclusiones del experimento se presentan en el Capítulo 5.

- **Objetivo 6.** Validación del marco de trabajo.

Se llevó a cabo un caso de estudio, presentado en el Capítulo 7, en la empresa Indra Software Labs de Ciudad Real para evaluar la utilidad del marco de trabajo SMF. Consistió en la medición en tres dominios distintos (requisitos, diagramas de clases UML y esquema de bases de datos relacionales) para satisfacer las necesidades de calidad de un proyecto real de la empresa.

8.2. Contraste de Resultados.

Los resultados y propuestas de esta tesis han sido publicados en diversos foros científicos. En la Tabla 8-1 se muestra una estadística de las publicaciones obtenidas, que están enmarcadas dentro del ámbito de la tesis. Las publicaciones sometidas a evaluación en estos momentos se han indicado con un signo "+".

Tipo	Total
Artículos en revistas internacionales	2+1
Artículos en revistas iberoamericanas	1
Libros	0
Capítulos de libros	1
Congresos internacionales	4
Congresos nacionales	3
Congresos iberoamericanos	2
Informes técnicos oficiales	3
TOTAL	16 + 1

Tabla 8-1. Estadística de las publicaciones de la tesis.

En la Tabla 8-2 se resumen las publicaciones clasificadas en función de la temática, dentro del trabajo de la tesis, que abordan de manera central. Para identificar cada publicación se ha utilizado un código (año y siglas) que se detalla en las tablas posteriores.

Publicación	SMF: Marco Conceptual				SMML		SMTool		
	Características y estructura general	Arquitectura Conceptual y Método de Trabajo	Proceso de Transformación	Métodos de medición genéricos	Definición del lenguaje: Sintaxis y Semántica	Validación empírica	Características Generales	SMML Editor	Motor de medición
2006-UCLM	X								
2010-KERJ					X			X	
2006b-UCLM	X								
2007-IDEAS	X	X							
2007-JISBD			X						X
2007-DSDM			X						X
2008-ICEIS					X			X	
2008-ICSOFT					X			X	
2008-OOPSLA	X	X							
2010-IEEEL	X	X							
2008- UCLM					X				
2010-QSIC				X					
2009-JISBD					X			X	
2009-CibSE					X				
2010- Ra-ma					X				
2010-SQJ						X			
2011-JSCT							X	X	X

Tabla 8-2. Lista de publicaciones clasificadas por temas.

En los siguientes sub-apartados se incluye la referencia completa de cada publicación, así como un resumen de sus aportaciones.

8.2.1. Capítulos de Libro.

Código	Referencia	Publicación
2010-Ra-ma	(Mora et al., 2010b)	**Mora, B.**, García, F., Ruiz, F. (2010b): SMML: Lenguaje para la representación de Modelos de Medición del Software. En "*Calidad del Producto y proceso software*". **Ra-Ma** (España), capítulo 4, pp. 91-115. ISBN: 978-84-7897-961-5

Tabla 8-3. Capítulos de Libro

8.2.2. Revistas.

Código	Referencia	Publicación
2010-IEEEL	(Mora et al., 2008d)	**Mora, B.**, García, F., Ruiz, F., Piattini, M., Boronat, A., Gómez, A., Carsí, J. Á. y Ramos, I. (2008d): Marco de Trabajo basado en MDA para la Medición Genérica del Software, **IEEE Latina**. Vol 6 (4), pp. 363-369
2010-KERJ	(García et al., 2009)	García, F., Ruiz, F., Calero, C., Bertoa, M. F., Vallecillo, A., **Mora, B.** y Piattini, M. (2009): On the Effective Use of Ontologies in Software Measurement, **KERJ**. Vol 24 (1), pp 23-40 Factor de impacto JCR 2009: 1,143
2010-SQJ	(Mora et al., 2010a)	**Mora, B.**, García, F., Ruiz, F. y Piattini, M. (2010): Graphical versus Textual Software Measurement Modelling: An Empirical Study. **Software Quality Journal.** (In press), DOI: 10.1007/s11219-010-9111-x. Factor de Impacto JCR 2009: 0,977
2011-JSCT	Mora et al., 2011	**Mora, B.**, Ruiz, F., García, F. y Piattini, M. (2011): SMTool: A Tool for Generic Software Measurement. **Journal of Science Computer and Technology.** (Pendiente Revisión)

Tabla 8-4. Lista de publicaciones en Revistas

8.2.3. Congresos.

8.2.3.1. Internacionales.

Código	Referencia	Publicación
2008-ICEIS	(Mora et al., 2008a)	**Mora, B.,** García, F., Ruiz, F., Piattini, M., Boronat, A., Gómez, A., Carsí, J. Á. y Ramos, I. (2008a) Software Measurement by using QVT Transformation in an MDA context. 10th International Conference on Enterprise Information Systems - **ICEIS** 2008. Vol 1, pp 117-124.
2008-ICSOFT	(Mora et al., 2008b)	**Mora, B.,** García, F., Ruiz, F. y Piattini, M. (2008b) Supporting Software Process Measurement by Using Metamodels: A DSL and a Framework, The Third International Workshop on Metamodelling - Utilization in Software Engineering, MUSE'09 in **ICSOFT**. Oporto (Portugal). (2008). Vol. SE/GSDCA/MUSE. pp. 305-312.
2008-OOPSLA	(Mora et al., 2008c)	**Mora, B.,** García, F., Ruiz, F. y Piattini, M. (2008c): SMML: Software Measurement Modeling Language. Proceedings of The 8th **OOPSLA** Workshop on Domain-Specific Modeling. pp. 52-59.
2010-QSIC	(Mora et al., 2009a)	**Mora, B.**, García, F., Ruiz, F. y Piattini, M. (2009b): Model-Driven Software Measurement Framework: a case study. The 9th International Conference on Quality Software QSIC. Pages: 239-248

Tabla 8-5. Artículos en Congresos Internacionales

8.2.3.2. Iberoamericanos.

Código	Referencia	Publicación
2007-IDEAS	(Mora et al., 2007a)	**Mora, B.,** Ruiz, F., García, F. y Piattini, M. (2007a): Experiencia en transformación de modelos de procesos de negocios desde BPMN a XPDL, IDEAS'07: X Workshop iberoamericano de Ingeniería de Requisitos y Ambientes de Software. pp.165-178. (2007).
2009-CibSE	(Mora et al., 2009b)	**Mora, B.**, García, F., Ruiz, F. y Piattini, M. (2009d): SMML: Lenguaje para la Representación de Modelos de Medición del Software, **CIbSE 2009,** pp 181-194

Tabla 8-6. Artículos en Congresos Iberoamericanos

8.2.3.3. Nacionales.

Código	Referencia	Publicación
2007-JISBD	(Mora et al., 2007b)	**Mora, B.,** García, F., Ruiz, F., Piattini, M., Boronat, A., Gómez, A., Carsí, J. Á. y Ramos, I. (2007b): Marco de Trabajo basado en MDA para la Medición Genérica del Software. Actas de las XII Jornadas de Ingeniería del Software y Bases de Datos, JISBD'2007. pp.211-220.
2007-DSDM	(Mora et al., 2007c)	**Mora, B.**, Ruiz, F., García, F. y Piattini, M. (2007c): Medición del Software mediante Transformaciones QVT. DSDM 2007: Desarrollo de Software Dirigido por Modelos, MDA y Aplicaciones (JISBD'07). Vol 6 (1), pp.119-135.
2009-JISBD	(Mora et al., 2009b)	**Mora, B.,** García, F., Ruiz y F., Piattini (2009c): Software Measurement Tool. XIV Jornadas de Ingeniería del Software y Bases de Dato. JISBD 2009. pp 418-421

Tabla 8-7. Artículos en Congresos Nacionales

8.2.4. Informes Técnicos.

Código	Referencia	Publicación
2006-UCLM	(Ferreira et al., 2006)	Ferreira, M., García, F., Ruiz, M., Bertoa, M. F., Calero, Vallecillo, A., **Mora, B** y Piattini, M. (2006): Medición de Software: Ontología y Metamodelo
2006b-UCLM	(Mora et al., 2006)	**Mora, B.,** Ruiz, F., García, F. y Piattini, M., (2006): Definición de Lenguajes de Modelos MDA Vs DSL
2008-UCLM	(Mora et al., 2008e)	**Mora, B.,** Ruiz, F., García, F. y Piattini, M., (2008e): SMML: Software Measurement Modeling Language

Tabla 8-8. Informes Técnicos

8.3. Líneas de Trabajo Futuras.

Las principales líneas futuras abiertas como resultado de esta investigación son las siguientes:

1. **Entorno SMF:**

- Estudiar las medidas genéricas de SMF que tienen aplicación en las medidas software de la actualidad. Para ello se realizará una revisión sistemática de las medidas base que existen en la bibliografía que usan los métodos de medición propuestos en SMF.
- Ampliar la lista de formas de medir genéricas definidas en SMF. En concreto, para las funciones de cálculo genéricas se utilizarán funciones estadísticas diseñadas en OCL (Cabot et al., 2010). A continuación se muestran algunos ejemplos de formas de medir genéricas:
  - Media: calcula el valor medio de una lista de valores.
  - Absoluto: calcula el valor absoluto de un valor.
  - Superior: obtiene el valor entero inmediatamente superior o igual a un número.

- Inferior: devuelve el valor entero inmediatamente inferior o igual a un número
- Modulo: devuelve el resto resultante de dividir dos números.
- Potencia: calcula la potencia de un numero
- Redondear: redondea números con el número de dígitos de precisión indicados.
- Raíz: calcula la raíz cuadrada de un número
- Varianza: calcula la varianza de un conjunto de valores.
- Truncar: trunca números para que tengan una cierta cantidad de dígitos de precisión.
- Mínimo: devuelve el mínimo de una lista de valores.

- **Realizar** nuevos **casos de estudio** y replicaciones del ya realizado para demostrar mejor la utilidad del marco de trabajo SMF a la hora llevar a cabo mediciones genéricas.

2. SMML:

- **Realizar** nuevos **experimentos** para consolidar la validación del uso de SMML.

3. SMTool:

- **Realizar** validaciones en la herramienta para comprobar su usabilidad con respecto a otras herramientas existentes.
- **Implementar** más formas de medir genéricas.

La calidad nunca es un accidente; siempre es el resultado de un esfuerzo de la inteligencia (John Ruskin)

Anexos.

- Material de los Experimentos de SMML
- Gráficas Estadísticas
- Protocolo de Revisión Sistemática
- Cuestionario del Caso de Estudio

A. Material de los Experimentos de SMML.

En este anexo se presenta el material de los experimentos de SMML entregado a los sujetos. En contenido es el siguiente:

- Bloque de ejercicios de entendibilidad para el grupo de sujetos A (apartado A.1).
- Bloque de ejercicios de modificabilidad para el grupo de sujetos A (apartado A.2).
- Bloque de ejercicios de entendibilidad para el grupo de sujetos B (apartado A.3).
- Bloque de ejercicios de modificabilidad para el grupo de sujetos B (apartado A.1).
- Cuestionario para valorar los elementos de SMML (apartado A.5).

A.1. Bloque de ejercicios de entendibilidad para el grupo de sujetos A .

EJERCICIO 1. Tareas a realizar:

Anotar la hora de inicio (indique hh:mm:ss): _____

1) **Conteste verdadero o falso las siguientes afirmaciones en base al diagrama SMML que se muestra al final del ejercicio:**

   a) Los *atributos* Tamaño, Complejidad y Longitud **están relacionados** con el *concepto medible* Mantenibilidad

      V ☐ F ☐

   b) La *medida* NA **está definida para** el *atributo* Tamaño

      V ☐ F ☐

   c) El *indicador* TMI **satisface** la *necesidad de información* Conocer la mantenibilidad de los esquemas relacionales

      V ☐ F ☐

   d) La medida SCI **tiene** la escala Ratio: entre 0 y 1 y la unidad Clave Ajena por Tabla

      V ☐ F ☐

   e) El *modelo de calidad* ISO9126 **evalúa** el *concepto medible* Usabilidad

      V ☐ F ☐

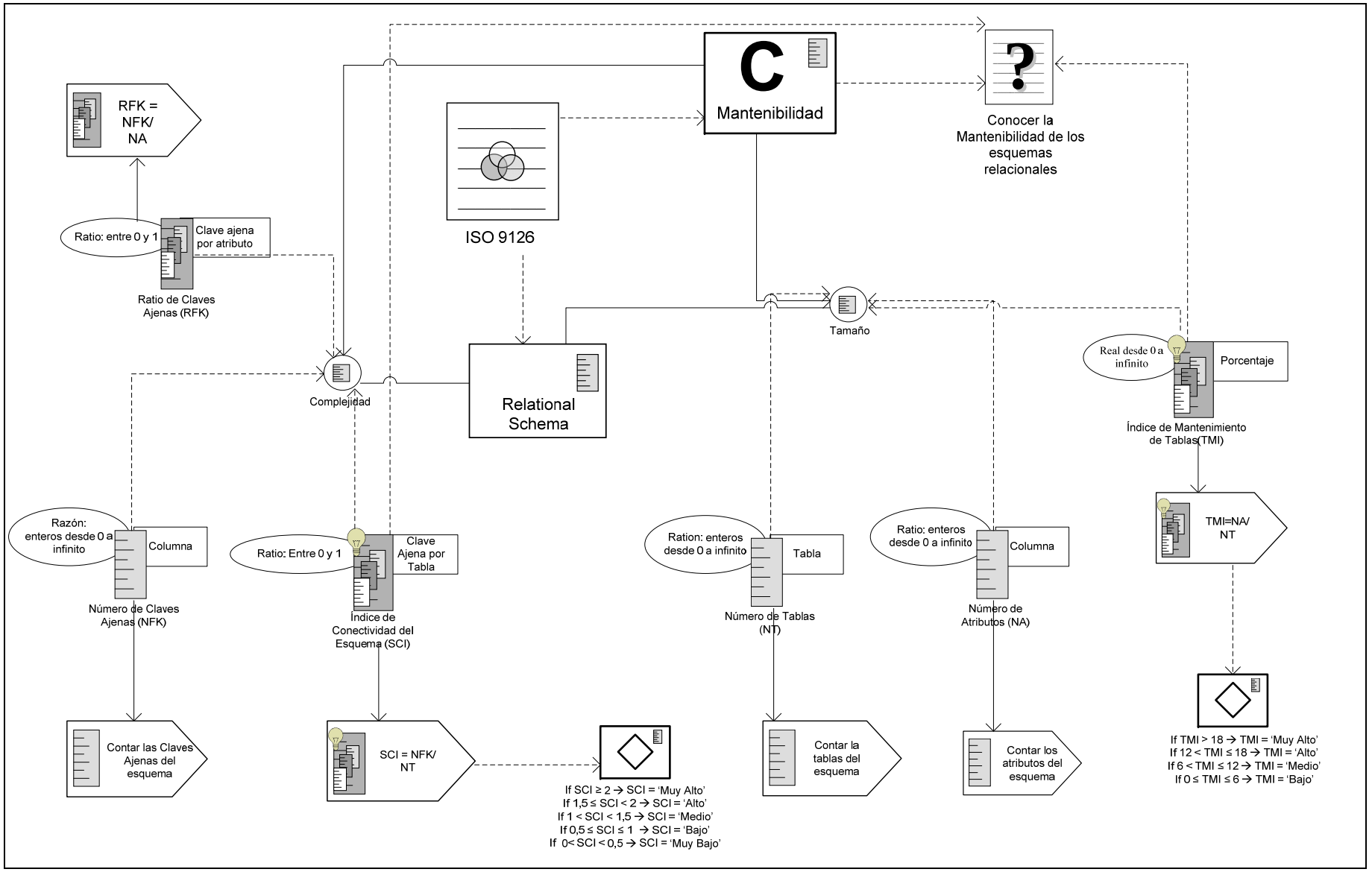
   f) La *medida derivada* RFK **se calcula con** la *función de cálculo* Contar las claves ajenas del esquema

      V ☐ F ☐

Anotar la hora de fin (indique hh:mm:ss): _____

2) **Intente comprender el diagrama SMML y valore según su criterio la COMPLEJIDAD del modelo de Medición del Software**

   Muy simple ☐ Algo simple ☐ Normal ☐ Algo complejo ☐ Muy complejo ☐


C
Mantenibilidad
?
Conocer la Mantenibilidad de los esquemas relacionales
RFK = NFK/ NA
ISO 9126
Ratio: entre 0 y 1
Clave ajena por atributo
Ratio de Claves Ajenas (RFK)
Tamaño
Complejidad
Relational Schema
Real desde 0 a infinito
Porcentaje
Índice de Mantenimiento de Tablas(TMI)
Razón: enteros desde 0 a infinito
Columna
Número de Claves Ajenas (NFK)
Ratio: Entre 0 y 1
Clave Ajena por Tabla
Índice de Conectividad del Esquema (SCI)
Ration: enteros desde 0 a infinito
Tabla
Número de Tablas (NT)
Ratio: enteros desde 0 a infinito
Columna
Número de Atributos (NA)
TMI=NA/ NT
Contar las Claves Ajenas del esquema
SCI = NFK/ NT
If SCI ≥ 2 → SCI = 'Muy Alto'
If 1,5 ≤ SCI < 2 → SCI = 'Alto'
If 1 < SCI < 1,5 → SCI = 'Medio'
If 0,5 ≤ SCI ≤ 1 → SCI = 'Bajo'
If 0< SCI < 0,5 → SCI = 'Muy Bajo'
Contar la tablas del esquema
Contar los atributos del esquema
If TMI > 18 → TMI = 'Muy Alto'
If 12 < TMI ≤ 18 → TMI = 'Alto'
If 6 < TMI ≤ 12 → TMI = 'Medio'
If 0 ≤ TMI ≤ 6 → TMI = 'Bajo'

EJERCICIO 2: Tareas a realizar:

Anotar la hora de inicio (indique hh:mm:ss): _____

1) Conteste verdadero o falso las siguientes afirmaciones en base al diagrama SMML que se muestra al final del ejercicio:

a) La *medida derivada* SIZE **se calcula con** el *modelo de análisis* SIZE=DAC+NOM

V ☐ F ☐

b) El *atributo* Tamaño **pertenece** a la *entidad* Diagrama clases UML

V ☐ F ☐

c) El *indicador* SIZE2 **está definida para** el *atributo* Tamaño

V ☐ F ☐

d) El *atributo* Tamaño **tiene asociadas sólo** las *medidas* DAC y NOM

V ☐ F ☐

e) El *indicador* SIZE2 **satisface** la *necesidad de información* Conocer la usabilidad de los diagramas de clases UML

V ☐ F ☐

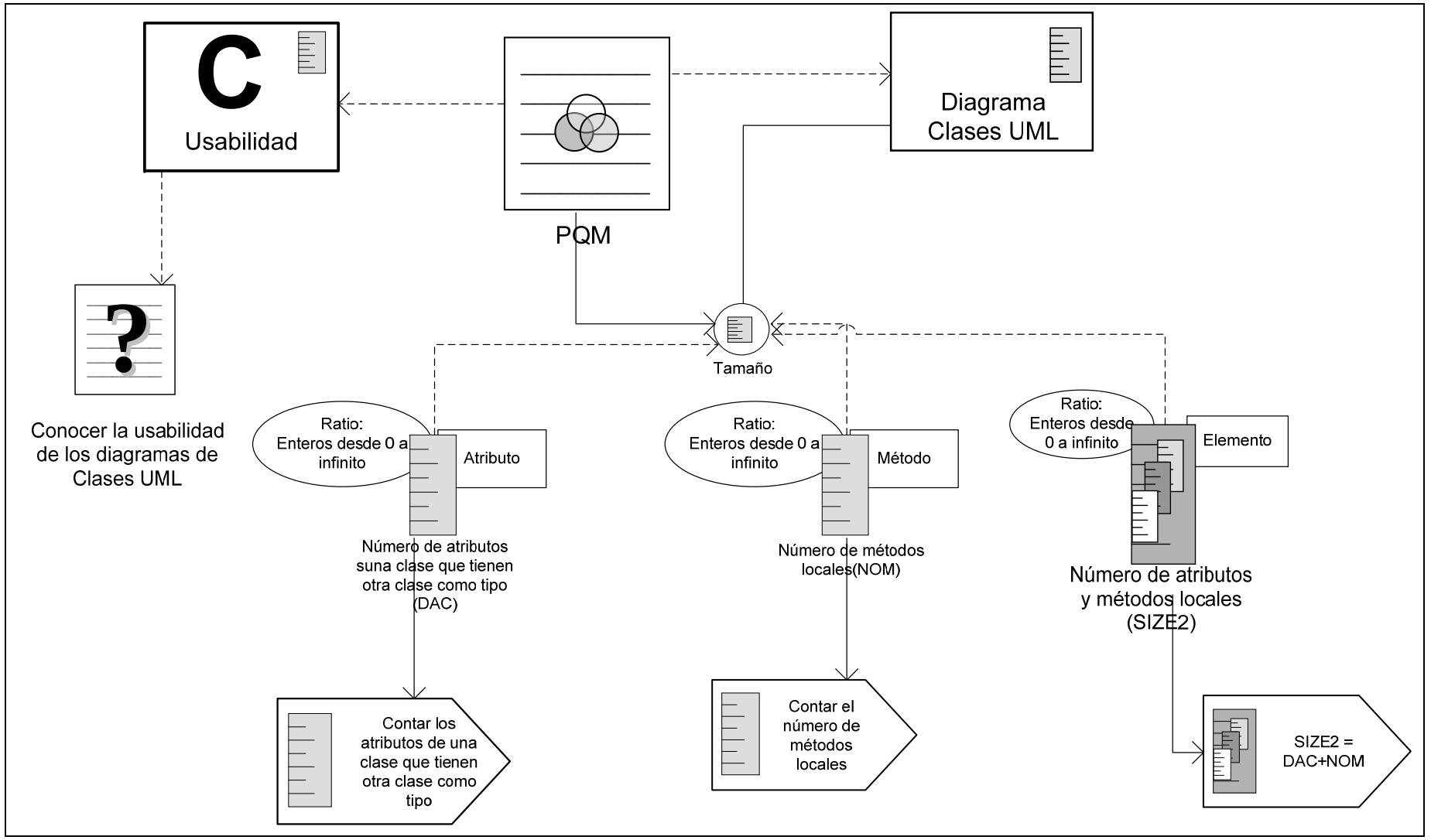
f) El *concepto medible* Usabilidad **está relacionado con** la *necesidad de información* Conocer la usabilidad de los diagramas de clases UML

V ☐ F ☐

Anotar la hora de fin (indique hh:mm:ss): _____

2) Intente comprender el diagrama SMML y valore según su criterio la COMPLEJIDAD del modelo de Medición del Software

Muy simple ☐ Algo simple ☐ Normal ☐ Algo complejo ☐ Muy complejo ☐


C
Usabilidad
Diagrama
Clases UML
PQM
?
Conocer la usabilidad
de los diagramas de
Clases UML
Tamaño
Ratio:
Enteros desde 0 a
infinito
Atributo
Número de atributos
suna clase que tienen
otra clase como tipo
(DAC)
Contar los
atributos de una
clase que tienen
otra clase como
tipo
Ratio:
Enteros desde 0 a
infinito
Método
Número de métodos
locales(NOM)
Contar el
número de
métodos
locales
Ratio:
Enteros desde
0 a infinito
Elemento
Número de atributos
y métodos locales
(SIZE2)
SIZE2 =
DAC+NOM

EJERCICIO 3: Tareas a realizar:

Anotar la hora de inicio (indique hh:mm:ss): _____

1) Conteste verdadero o falso las siguientes afirmaciones en base a las tablas que se muestran al final del ejercicio:

a) El *modelo de calidad* DQ Representational **está definido para** la *entidad* Portal Web

V ☐ F ☐

b) El *concepto medible* Atracción **relaciona** el *atributo* Representación consistente

V ☐ F ☐

c) El *atributo* Representación consistente **tiene asociado** el *indicador* LCsR

V ☐ F ☐

d) El *modelo de análisis* LCsR=PSSD*0,5+SDCD*0,5 **usa** el *criterio de decisión* Si LCsR > 0,7 --> Alto, si 0,4<LCsR<=0,7 --> Medio, Si 0<=LCsR<=0,4 --> Bajo

V ☐ F ☐

e) La *medida base* SDCD **utiliza** el *método de medición* SDCD=LTC/LnC

V ☐ F ☐

f) La medida LTC **tiene** la escala Ratio: entero desde 0 a 1 y la unidad Vínculo

V ☐ F ☐

Anotar la hora de fin (indique hh:mm:ss): _____

2) Intente comprender las tablas y valore según su criterio la COMPLEJIDAD del modelo de Medición del Software

Muy simple ☐ Algo simple ☐ Normal ☐ Algo complejo ☐ Muy complejo ☐

Especificación Textual:

Elementos del metamodelo de medición (clases MOF)	Modelo de medición (Instancias M2)	Relaciones del metamodelo de medición (Asociaciones MOF)		Modelo de medición (links M2)
Atributo	Representación consistente	Tiene	Categoría de entidad	Portal Web
			Atributo	Representación consistente
Categoría de entidad	Portal Web	Se define para	Modelo de calidad	DQ Representational
			Categoría de entidad	Portal Web
Modelo de calidad	DQ Representational	Evalúa	Modelo de calidad	DQ Representational
			Concepto medible	Entendibilidad Atracción Representación
Concepto medible	Entendibilidad Atracción Representación	Relaciona	Atributo	Representación consistente
			Concepto medible	Entendibilidad Atracción Representación
Necesidad de información	Conocer el nivel de calidad de representación de datos de un portal web	Se asocia con	Concepto medible	Entendibilidad Atracción Representación
			Necesidad de información	Conocer el nivel de calidad de representación de datos de un portal web
Medida	PgC, StPgCs, LnC, LTC, MaSS, SDCD, PSSD, LCsR	Se define para	Medida	PgC, StPgCs, LnC, LTC MaSS, SDCD, PSSD, LCsR
			Atributo	Representación consistente
Indicador	LCsR	Satisface	Indicador	LCsR
			Necesidad de Información	Conocer el nivel de calidad de representación de datos de un portal web

Medida Base	Descripción	Método de medición	Escala	Unidad
PgC	Número de páginas	Contar las páginas de un portal web	Naturales [0-∞)	Página
StPgCs	Páginas con estilos	Contar las páginas con estilos dados	Naturales [0-∞)	Página

Medida Base	Descripción	Método de medición	Escala	Unidad
LnC	Número de vínculos	Contar los vínculos de un portal web	Naturales [0-∞)	Vínculo
LTC	Correspondencia del vínculo del texto	Contar los vínculos con palabras comunes	Naturales [0-∞)	Vínculo

Medida Derivada	Descripción	Función de cálculo	Escala	Unidad
MaSS	Número de páginas con mayor número de estilos usados	Mass = MAX ($\forall_i$, $StPgC_i$)	Naturales [0-∞)	Página
SDCD	Grado de correspondencia origen destino	SDCD = LTC/LnC	Naturales [0-∞)	Página
PSSD	Páginas con el mismo grado de estilo	PSSD = MaSS/PgC	Naturales [0-∞)	Página

Indicador	Descripción	Modelo de análisis	Criterio de decisión	Escala	Unidad
LCsR	Nivel de representación consistente	LCsR = PSSD*0.5+SDCD*0.5	Si LCsR > 0,7 → LCsR = **'Alto'** Si 0,4 < LCsR ≤ 0,7 → TMI = **'Medio'** Si 0 ≤ LCsR ≤ 0,4 → LCsR = **'Bajo'**	Naturales [0-∞)	Página

EJERCICIO 4: Tareas a realizar:

Anotar la hora de inicio (indique hh:mm:ss): _____

1) Conteste verdadero o falso las siguientes afirmaciones en base a las tablas que se muestran al final del ejercicio:

a) El *atributo* Tamaño **tiene asociada** las *medida base* NDA

V ☐ F ☐

b) El *criterio de decisión* Si IME > 15, IME= 'Muy Alto', Si 10 < IME≤ 15, IME= 'Alto', Si 5 < IME≤ 10, IME= 'Medio', Si 0 ≤ IME≤ 5, IME= 'Bajo' **se usa en** el *modelo de análisis* IME=NA/NE

V ☐ F ☐

c) La medida NA **tiene** la escala Ratio: enteros desde 0 a infinito y la unidad Atributo

V ☐ F ☐

d) La *necesidad de información* Conocer la mantenibilidad de diagramas E/R **se relaciona** el *concepto medible* Mantenibilidad y Usabilidad

V ☐ F ☐

e) La *entidad* Diagrama E/R **tiene** el *atributo* Longitud

V ☐ F ☐

f) La *medida derivada* NA **se calcula con** la *función de cálculo* Contar el número de tablas de un diagrama E/R

V ☐ F ☐

Anotar la hora de fin (indique hh:mm:ss): _____

2) Intente comprender las tablas y valore según su criterio la COMPLEJIDAD del modelo de Medición del Software

Muy simple ☐ Algo simple ☐ Normal ☐ Algo complejo ☐ Muy complejo ☐

Especificación Textual:

Elementos del metamodelo de medición (clases MOF)	Modelo de medición (instancias M2)	Relaciones del metamodelo de medición (asociaciones MOF)		Modelo medición (links M2)
Atributo	Tamaño	Tiene	Categoría de entidad	Diagrama E/R
			Atributo	Tamaño
Categoría de entidad	Diagrama E/R	Se define para	Modelo de calidad	WQM
			Categoría de entidad	Diagrama E/R
Modelo de calidad	WQM	Evalúa	Modelo de calidad	WQM
			Concepto medible	Mantenibilidad
Concepto medible	Mantenibilidad	Relates	Concepto medible	Mantenibilidad
			Atributo	Tamaño
Necesidad de información	Conocer la mantenibilidad de los diagramas E/R	Se asocia con	Concepto medible	Mantenibilidad
			Necesidad de información	Conocer la mantenibilidad de los diagramas E/R
Medida	NE, NAS, NCA, NDA, NA, IME	Se define para	Medida	NE, NAS, NCA, NDA, NA, IME
			Atributo	Tamaño
Indicador	IME	Satisface	Indicador	IME
			Necesidad de información	Conocer la mantenibilidad de los diagramas E/R

Medida Base	Descripción	Método de medición	Escala	Unidad
NE	Número de entidades	Contar el número de entidades del diagrama E/R	Ratio: enteros desde 0 a infinito	Entidad
NAS	Número de atributos simples	Contar el número de atributos simples del diagrama E/R	Ratio: enteros desde 0 a infinito	Atributo
NCA	Número de atributos compuestos	Contar el número de atributos compuestos del diagrama E/R	Ratio: enteros desde 0 a infinito	Atributo
NDA	Número de atributos derivados	Contar el número de atributos derivados del diagrama E/R	Ratio: enteros desde 0 a infinito	Atributo

Medida Derivada	Descripción	Función de cálculo	Escala	Unidad
NA	Número total de atributos del diagrama E/R	NA = NAS+NCA+NDA	Ratio: enteros desde 0 a infinito	Atributo

Indicador	Descripción	Modelo de análisis	Criterio de decisión	Escala	Unidad
IME	Indicador de mantenimiento de entidad	IME = NA/NE	Si IME > 15 → IME= **'Muy Alto'** Si 10 < IME≤ 15 → IME= **'Alto'** Si 5 < IME≤ 10 → IME= **'Medio'** Si 0 ≤ IME≤ 5 → IME= **'Bajo'**	Ratio: enteros desde 0 a infinito	Porcentaje

A.2. Bloque de ejercicios de modificabilidad para el grupo de sujetos A.

EJERCICIO 1: Tareas a realizar:

Anotar la hora de inicio (indique hh:mm:ss): _____

1) **Realice las modificaciones necesarias en las tablas (indique el número de apartado en cada modificación), representadas al final del ejercicio, para satisfacer los siguientes requisitos de medición del software:**

   a) Se desea eliminar *la medida base* NT, elimina si fuera necesario las medidas que usan esta *medida base*

   b) Se desea añadir la *medida base* NPK (Número de claves primaria) **definida para** el *atributo* Complejidad y que **usa** el *método de medición* contar el número de claves primarias. Con *escala* Ratio: enteros desde 0 a infinito y *unidad* clave primaria.

   c) Se desea modificar el *modelo de calidad* ISO 9126 para que **evalúe** los conceptos medibles Mantenibilidad *y* Portabilidad

   d) Se desea modificar la *Necesidad de información* conocer la portabilidad de los esquemas relacionales para que **se relaciona** con el *concepto medible* Portabilidad.

Anotar la hora de fin (indique hh:mm:ss): _____

2) **Intente comprender las tablas y valore según su criterio la COMPLEJIDAD del modelo de Medición del Software**

   Muy simple ☐ Algo simple ☐ Normal ☐ Algo complejo ☐ Muy complejo ☐

Especificación Textual:

Elementos del metamodelo de medición (clases MOF)	Modelo de medición (instancia M2)	Relaciones del metamodelo de medición (asociaciones MOF)		Modelo medición (links M2)
Atributo	Tamaño Complejidad	Tiene	Categoría de entidad	Esquema relacional
			Atributo	Tamaño Complejidad
Categoría de entidad	Esquema Relacional	Se define para	Modelo de calidad	ISO 9126
			Categoría de entidad	Esquema Relacional
Modelo de Calidad	ISO 9126	Evalúa	Modelo de Calidad	ISO 9126
			Concepto Medible	Mantenibilidad
Concepto medible	Mantenibilidad	Relaciona	Concepto medible	Mantenibilidad
			Atributo	Tamaño, Complejidad
Necesidad de Información	Conocer la mantenibilidad de los Esquemas Relacionales	Se asocia con	Concepto Medible	Mantenibilidad
			Necesidad de Información	Conocer la mantenibilidad de los Esquemas relacionales
Medida	NFK, RFK, SCI	Se define para	Medida	NFK, RFK, SCI
			Atributo	Complejidad
Medida	NT, NA, TMI	Se define para	Medida	NT, NA, TMI
			Atributo	Tamaño
Indicador	TMI, SCI	Satisface	Indicador	TMI, SCI
			Necesidad de información	Conocer la mantenibilidad de los esquemas relacionales

Medida Base	Descripción	Método de medición	Escala	Unidad
NFK	Número de Claves Ajenas	Contar las claves ajenas del esquema	Ratio: enteros desde 0 a infinito	Columna
NT	Número de Tablas	Contar las tablas de un esquema	Ratio: enteros desde 0 a infinito	Tabla
NA	Número de Atributos	Contar los atributos del esquema	Ratio: enteros desde 0 a infinito	Columna

Medida Derivada	Descripción	Función de cálculo	Escala	Unidad
RFK	Ratio de claves ajenas	RFK = NFK/NA	Ratio: entre 0 y 1	Clave ajena por atributo

Indicator	Descripción	Modelo de análisis	Criterio de decisión	Escala	Unidad
SCI	Índice de Conectividad del Esquema	SCI = NFK/NT	Si SCI ≥ 2 → SCI = 'Muy alto' Si 1,5 ≤ SCI < 2 → SCI = 'Alta' Si 1 < SCI < 1,5 → SCI = 'Medio' Si 0,5 ≤ SCI ≤ 1 → SCI = 'Bajo' Si 0< SCI < 0,5 → SCI = 'Muy Bajo'	Ratio: entre 0 y 1	Clave ajena por Tabla
TMI	Índice de Mantenimiento de Tablas	TMI = NA/NT	Si TMI > 18 → TMI = 'Muy Alto' Si 12 < TMI ≤ 18 → TMI = 'Alto' Si 6 < TMI ≤ 12 → TMI = 'Medio' Si 0 ≤ TMI ≤ 6 → TMI = 'Bajo'	Ratio: entre 0 y 1	Porcentaje

EJERCICIO 2: Tareas a realizar:

Anotar la hora de inicio (indique hh:mm:ss): _____

1) **Realice las modificaciones necesarias en las tablas (indique el número de apartado en cada modificación), representadas al final del ejercicio, para satisfacer los siguientes requisitos de medición del software:**

   a) Se desea modificar la *medida* DAC para que **utilice** la *escala* ratio: reales desde 0 a infinito

   b) Se desea modificar el *atributo* Tamaño para que **sólo tenga asociadas** las *medidas* NOM y DAC

   c) Se desea modificar la *categoría de entidad* Diagrama Clases UML para que **tenga definido** el *modelo de calidad* PQM y el *modelo de calidad* WQM

   d) Se desea modificar el *concepto medible* usabilidad para que **sea evaluado** por el *modelo de calidad* PQM y por el *modelo de calidad* WQM

Anotar la hora de fin (indique hh:mm:ss): _____

2) **Intente comprender las tablas y valore según su criterio la COMPLEJIDAD del modelo de Medición del Software**

   Muy simple ☐ Algo simple ☐ Normal ☐ Algo complejo ☐ Muy complejo ☐

Especificación Textual:

Elementos del metamodelo de medición (clases MOF)	Modelo de medición (instancia M2)	Relaciones del metamodelo de medición (asociaciones MOF)		Modelo de medición (links M2)
Atributo	Tamaño	Tiene	Categoría de entidad	Diagrama Clases UML
			Atributo	Tamaño
Categoría de entidad	Diagrama Clases UML	Se define para	Modelo de calidad	PQM
			Categoría de entidad	Diagrama Clases UML
Modelo de calidad	PQM	Evalúa	Modelo de calidad	PQM
			Concepto medible	Usabilidad
Concepto medible	Usabilidad	Relaciona	Concepto medible	Usabilidad
			Atributo	Tamaño
Necesidad de información	Conocer la usabilidad de los diagramas de clases UML	Se asocia con	Concepto medible	Usabilidad
			Necesidad de información	Conocer la usabilidad de los diagramas de clases UML
Medida	DAC, NOM, SIZE2	Se define para	Medida	DAC, NOM, SIZE2
			Atributo	Tamaño

Medida Base	Descripción	Método de medición	Escala	Unidad
DAC	Número de atributos en una clase que tiene otra clases como tipo	Contar el número de atributos en una clase que tiene otra clase como tipo.	Ratio: enteros de 0 a infinito.	Atributo
NOM	Número de métodos locales	Contar el número de métodos locales.	Ratio: enteros de 0 a infinito.	Método

Medida Derivada	Descripción	Función de cálculo	Escala	Unidad
SIZE2	Número de atributos y métodos locales (SIZE2)	SIZE2 = DAC + NOM	Ratio: enteros de 0 a infinito.	Elemento

EJERCICIO 3 - Tareas a realizar:

Anotar la hora de inicio (indique hh:mm:ss): _____

1) **Realice las modificaciones necesarias en el diagrama SMML (indique el número de apartado en cada modificación), representado al final del ejercicio, para satisfacer los siguientes requisitos de medición del software:**

   a) Se desea modificar el *atributo* Representación consistente para que (sólo) **esté relacionado con** los *conceptos medibles* Atracción y Representación

   b) Se desea modificar el *atributo* Representación consistente para que (sólo) **tenga asociadas** las *medidas* base LnC y LTC

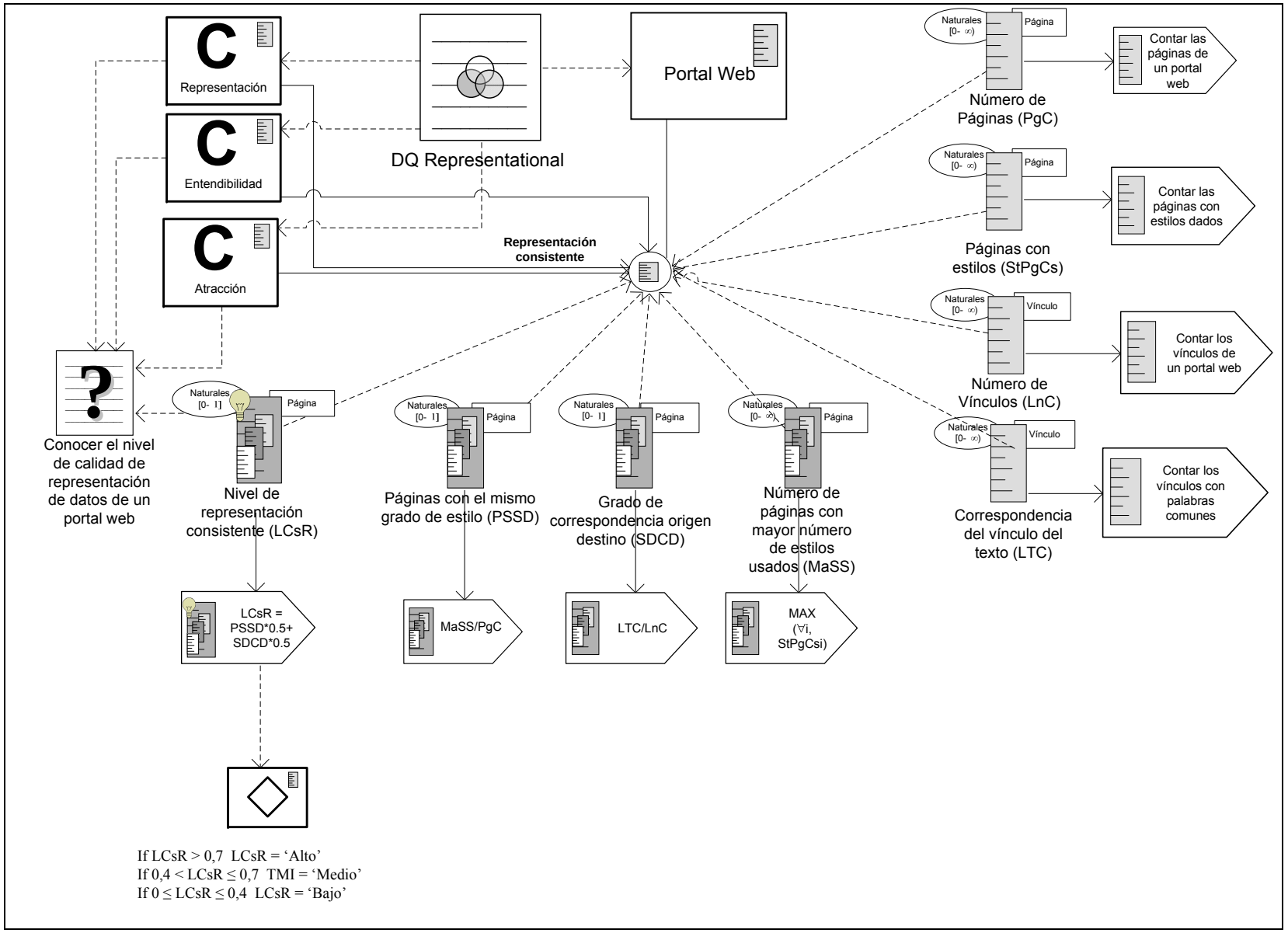
   c) Se desea modificar el *concepto medible* Atracción para que (sólo) **relacione** los *atributos* Representación consistente, y Complejidad, que **pertenecen a** *la categoría de entidad* Portal Web.

   d) Se desea modificar la *necesidad de información* Conocer la representación de los portales web para que (sólo) **se relacione con** el *concepto medible* Atracción.

Anotar la hora de fin (indique hh:mm:ss): _____

2) **Intente comprender el diagrama SMML y valore según su criterio la COMPLEJIDAD del modelo de Medición del Software**

   Muy simple ☐ Algo simple ☐ Normal ☐ Algo complejo ☐ Muy complejo ☐


C
Representación
C
Entendibilidad
C
Atracción
DQ Representational
Portal Web
Representación consistente
Naturales [0- ∞)
Página
Contar las páginas de un portal web
Número de Páginas (PgC)
Contar las páginas con estilos dados
Páginas con estilos (StPgCs)
Vínculo
Contar los vínculos de un portal web
Número de Vínculos (LnC)
Contar los vínculos con palabras comunes
Correspondencia del vínculo del texto (LTC)
Conocer el nivel de calidad de representación de datos de un portal web
Naturales [0- 1]
Nivel de representación consistente (LCsR)
Páginas con el mismo grado de estilo (PSSD)
Grado de correspondencia origen destino (SDCD)
Número de páginas con mayor número de estilos usados (MaSS)
LCsR = PSSD*0.5+ SDCD*0.5
MaSS/PgC
LTC/LnC
MAX (∀i, StPgCsi)
If LCsR > 0,7 LCsR = 'Alto'
If 0,4 < LCsR ≤ 0,7 TMI = 'Medio'
If 0 ≤ LCsR ≤ 0,4 LCsR = 'Bajo'

EJERCICIO 4: Tareas a realizar:

Anotar la hora de inicio (indique hh:mm:ss): _____

1) **Realice las modificaciones necesarias en el diagrama SMML (indique el número de apartado en cada modificación), representado al final del ejercicio, para satisfacer los siguientes requisitos de medición del software:**

   a) Se desea modificar El *atributo* Tamaño para que (sólo) **tenga asociadas** las *medidas* base NCA, NAS y NDA

   b) Se desea modificar el *concepto medible* mantenibilidad para que **relacione** el *atributo* complejidad

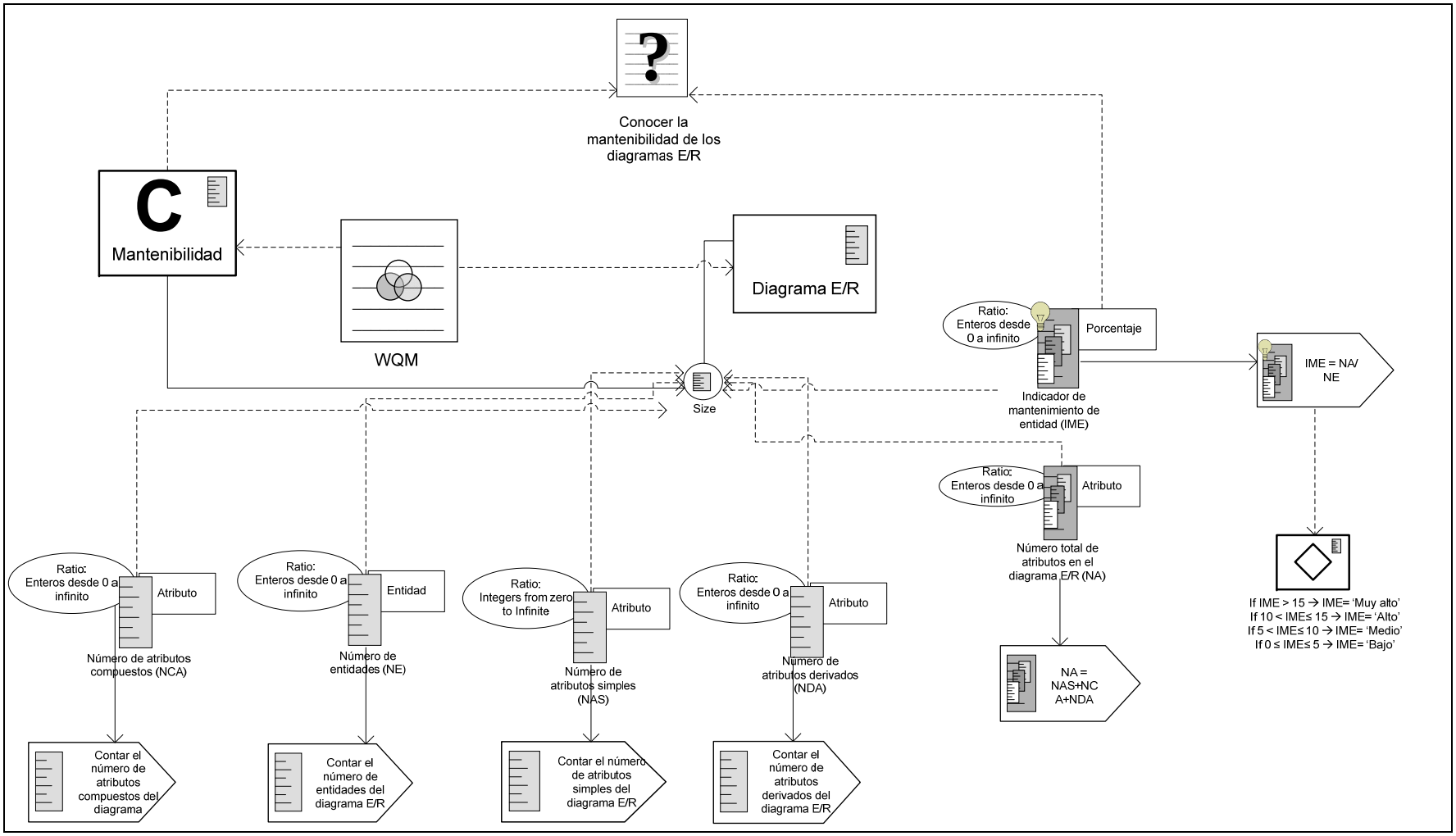
   c) Se desea modificar la *categoría de entidad* Diagrama E/R para que **tenga** los *atributos* tamaño y complejidad

   d) Se desea añadir una nueva *medida base* NrefR (Número de relaciones reflexivas) **definida para** el *atributo* complejidad, que **usa** el *método de medición* contar el número de relaciones reflexivas del diagrama E/R, y **tiene** la Escala "naturales [0-∞]" y *Unidad* Relación

Anotar la hora de fin (indique hh:mm:ss): _____

2) **Intente comprender el diagrama SMML y valore según su criterio la COMPLEJIDAD del modelo de Medición del Software**

   Muy simple ☐ Algo simple ☐ Normal ☐ Algo complejo ☐ Muy complejo ☐


Conocer la mantenibilidad de los diagramas E/R
C
Mantenibilidad
WQM
Diagrama E/R
Size
Ratio: Enteros desde 0 a infinito
Porcentaje
Indicador de mantenimiento de entidad (IME)
IME = NA/ NE
If IME > 15 → IME= 'Muy alto'
If 10 < IME≤ 15 → IME= 'Alto'
If 5 < IME≤ 10 → IME= 'Medio'
If 0 ≤ IME≤ 5 → IME= 'Bajo'
Ratio: Enteros desde 0 a infinito
Atributo
Número total de atributos en el diagrama E/R (NA)
NA = NAS+NCA+NDA
Ratio: Enteros desde 0 a infinito
Atributo
Número de atributos compuestos (NCA)
Contar el número de atributos compuestos del diagrama
Ratio: Enteros desde 0 a infinito
Entidad
Número de entidades (NE)
Contar el número de entidades del diagrama E/R
Ratio: Integers from zero to Infinite
Atributo
Número de atributos simples (NAS)
Contar el número de atributos simples del diagrama E/R
Ratio: Enteros desde 0 a infinito
Atributo
Número de atributos derivados (NDA)
Contar el número de atributos derivados del diagrama E/R

A.3. Bloque de ejercicios de entendibilidad para el grupo de sujetos B.

EJERCICIO 1. Tareas a realizar:

Anotar la hora de inicio (indique hh:mm:ss): _____

1) **Conteste verdadero o falso las siguientes afirmaciones en base a las tablas que se muestra al final del ejercicio:**
   a. Los *atributos* Tamaño, Complejidad y Longitud **están relacionados** con el *concepto medible* Mantenibilidad

      V ☐ F ☐

   b. La *medida* NA **está definida para** el *atributo* Tamaño

      V ☐ F ☐

   c. El *indicador* TMI **satisface** la *necesidad de información* Conocer la mantenibilidad de los esquemas relacionales

      V ☐ F ☐

   d. La medida SCI **tiene** la escala Ratio: entre 0 y 1 y la unidad Clave Ajena por Tabla

      V ☐ F ☐

   e. El *modelo de calidad* ISO9126 **evalúa** el *concepto medible* Usabilidad

      V ☐ F ☐

   f. La *medida derivada* RFK **se calcula con** la *función de cálculo* Contar las claves ajenas del esquema

      V ☐ F ☐

Anotar la hora de fin (indique hh:mm:ss): _____

2) **Intente comprender las tablas y valore según su criterio la COMPLEJIDAD del modelo de Medición del Software**

   Muy simple ☐ Algo simple ☐ Normal ☐ Algo complejo ☐ Muy complejo ☐

Especificación Textual:

Elementos del metamodelo de medición (clases MOF)	Modelo de medición (instancia M2)	Relaciones del metamodelo de medición (asociaciones MOF)		Modelo medición (links M2)
Atributo	Tamaño Complejidad	Tiene	Categoría de entidad	Esquema relacional
			Atributo	Tamaño Complejidad
Categoría de entidad	Esquema Relacional	Se define para	Modelo de calidad	ISO 9126
			Categoría de entidad	Esquema Relacional
Modelo de Calidad	ISO 9126	Evalúa	Modelo de Calidad	ISO 9126
			Concepto Medible	Mantenibilidad
Concepto medible	Mantenibilidad	Relaciona	Concepto medible	Mantenibilidad
			Atributo	Tamaño, Complejidad
Necesidad de Información	Conocer la mantenibilidad de los Esquemas Relacionales	Se asocia con	Concepto Medible	Mantenibilidad
			Necesidad de Información	Conocer la mantenibilidad de los Esquemas relacionales
Medida	NFK, RFK, SCI	Se define para	Medida	NFK, RFK, SCI
			Atributo	Complejidad
Medida	NT, NA, TMI	Se define para	Medida	NT, NA, TMI
			Atributo	Tamaño
Indicador	TMI, SCI	Satisface	Indicador	TMI, SCI
			Necesidad de información	Conocer la mantenibilidad de los esquemas relacionales

Medida Base	Descripción	Método de medición	Escala	Unidad
NFK	Número de Claves Ajenas	Contar las claves ajenas del esquema	Ratio: enteros desde 0 a infinito	Columna
NT	Número de Tablas	Contar las tablas de un esquema	Ratio: enteros desde 0 a infinito	Tabla
NA	Número de Atributos	Contar los atributos del esquema	Ratio: enteros desde 0 a infinito	Columna

Medida Derivada	Descripción	Función de cálculo	Escala	Unidad
RFK	Ratio de claves ajenas	RFK = NFK/NA	Ratio: entre 0 y 1	Clave Ajena por Atributo

Indicator	Descripción	Modelo de análisis	Criterio de decisión	Escala	Unidad
SCI	Índice de Conectividad del Esquema	SCI = NFK/NT	Si SCI ≥ 2 → SCI = 'Muy alto' Si 1,5 ≤ SCI < 2 → SCI = 'Alta' Si 1 < SCI < 1,5 → SCI = 'Medio' Si 0,5 ≤ SCI ≤ 1 → SCI = 'Bajo' Si 0< SCI < 0,5 → SCI = 'Muy Bajo'	Ratio: entre 0 y 1	Clave Ajena por Tabla
TMI	Índice de Mantenimiento de Tablas	TMI = NA/NT	Si TMI > 18 → TMI = 'Muy Alto' Si 12 < TMI ≤ 18 → TMI = 'Alto' Si 6 < TMI ≤ 12 → TMI = 'Medio' Si 0 ≤ TMI ≤ 6 → TMI = 'Bajo'	Ratio: entre 0 y 1	Porcentaje

EJERCICIO 2: Tareas a realizar:

Anotar la hora de inicio (indique hh:mm:ss): _____

1) Conteste verdadero o falso las siguientes afirmaciones en base a las tablas que se muestran al final del ejercicio:

a. La *medida derivada* SIZE **se calcula con** el *modelo de análisis* SIZE=DAC+NOM

V ☐ F ☐

b. El *atributo* Tamaño **pertenece** a la *entidad* Diagrama Clases UML

V ☐ F ☐

c. El *indicador* SIZE2 **está definida para** el *atributo* Tamaño

V ☐ F ☐

d. El *atributo* Tamaño **tiene asociadas sólo** las *medidas* DAC y NOM

V ☐ F ☐

e. El *indicador* SIZE2 **satisface** la *necesidad de información* Conocer la usabilidad de los diagramas de clases UML

V ☐ F ☐

f. El *concepto medible* Usabilidad **está relacionado con** la *necesidad de información* Conocer la usabilidad de los diagramas de clases UML

V ☐ F ☐

Anotar la hora de fin (indique hh:mm:ss): _____

2) Intente comprender las tablas y valore según su criterio la COMPLEJIDAD del modelo de Medición del Software

Muy simple ☐ Algo simple ☐ Normal ☐ Algo complejo ☐ Muy complejo ☐

Especificación Textual:

Elementos del metamodelo de medición (clases MOF)	Modelo de medición (instancia M2)	Relaciones del metamodelo de medición (asociaciones MOF)		Modelo de medición (links M2)
Atributo	Tamaño	Tiene	Categoría de entidad	Diagrama Clases UML
			Atributo	Tamaño
Categoría de entidad	Diagrama Clases UML	Se define para	Modelo de calidad	PQM
			Categoría de entidad	Diagrama Clases UML
Modelo de calidad	PQM	Evalúa	Modelo de calidad	PQM
			Concepto medible	Usabilidad
Concepto medible	Usabilidad	Relaciona	Concepto medible	Usabilidad
			Atributo	Tamaño
Necesidad de información	Conocer la usabilidad de los diagramas de clases UML	Se asocia con	Concepto medible	Usabilidad
			Necesidad de información	Conocer la usabilidad de los diagramas de clases UML
Medida	DAC, NOM, SIZE2	Se define para	Medida	DAC, NOM, SIZE2
			Atributo	Tamaño

Medida Base	Descripción	Método de medición	Escala	Unidad
DAC	Número de atributos en una clase que tiene otra clases como tipo	Contar el número de atributos en una clase que tiene otra clase como tipo.	Ratio: enteros de 0 a infinito.	Atributo
NOM	Número de métodos locales	Contar el número de métodos locales.	Ratio: enteros de 0 a infinito.	Método

Medida Derivada	Descripción	Función de cálculo	Escala	Unidad
SIZE2	Número de atributos y métodos locales (SIZE2)	SIZE2 = DAC + NOM	Ratio: enteros de 0 a infinito.	Elemento

EJERCICIO 3: Tareas a realizar:

Anotar la hora de inicio (indique hh:mm:ss): _____

1) **Conteste verdadero o falso las siguientes afirmaciones en base al diagrama que se muestra al final del ejercicio:**

   a) El *modelo de calidad* DQ Representational **está definido para** la *entidad* Portal Web

      V ☐ F ☐

   b) El *concepto medible* Atracción **relaciona** el *atributo* Representación consistente

      V ☐ F ☐

   c) El *atributo* Representación consistente **tiene asociado** el *indicador* LCsR

      V ☐ F ☐

   d) El *modelo de análisis* LCsR=PSSD*0,5+SDCD*0,5 **usa** el *criterio de decisión* Si LCsR > 0,7 --> Alto, si 0,4<LCsR<=0,7 --> Medio, Si 0<=LCsR<=0,4 --> Bajo

      V ☐ F ☐

   e) La *medida base* SDCD **utiliza** el *método de medición* SDCD=LTC/LnC

      V ☐ F ☐

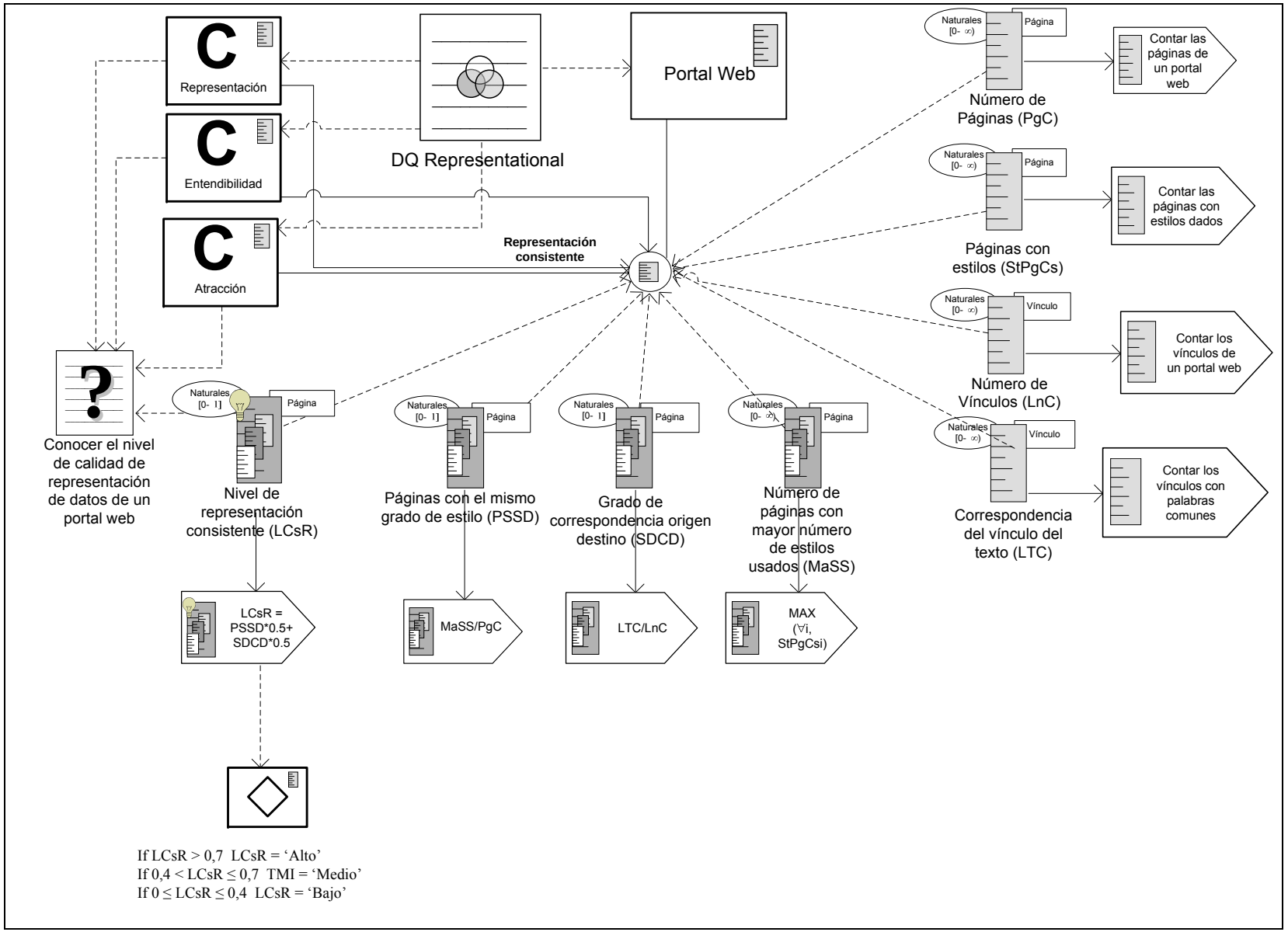
   f) La medida LTC **tiene** la escala Ratio: entero desde 0 a 1 y la unidad Vínculo

      V ☐ F ☐

Anotar la hora de fin (indique hh:mm:ss): _____

2) **Intente comprender el diagrama y valore según su criterio la COMPLEJIDAD del modelo de Medición del Software**

   Muy simple ☐ Algo simple ☐ Normal ☐ Algo complejo ☐ Muy complejo ☐


C
Representación
C
Entendibilidad
C
Atracción
DQ Representational
Portal Web
Representación consistente
Naturales [0- ∞)
Página
Contar las páginas de un portal web
Número de Páginas (PgC)
Contar las páginas con estilos dados
Páginas con estilos (StPgCs)
Vínculo
Contar los vínculos de un portal web
Número de Vínculos (LnC)
Contar los vínculos con palabras comunes
Correspondencia del vínculo del texto (LTC)
Naturales [0- 1]
Conocer el nivel de calidad de representación de datos de un portal web
Nivel de representación consistente (LCsR)
Páginas con el mismo grado de estilo (PSSD)
Grado de correspondencia origen destino (SDCD)
Número de páginas con mayor número de estilos usados (MaSS)
LCsR = PSSD*0.5+ SDCD*0.5
MaSS/PgC
LTC/LnC
MAX (∀i, StPgCsi)
If LCsR > 0,7 LCsR = 'Alto'
If 0,4 < LCsR ≤ 0,7 TMI = 'Medio'
If 0 ≤ LCsR ≤ 0,4 LCsR = 'Bajo'

EJERCICIO 4: Tareas a realizar:

Anotar la hora de inicio (indique hh:mm:ss): _____

1) **Conteste verdadero o falso las siguientes afirmaciones en base al diagrama que se muestra al final del ejercicio:**

   a) El *atributo* Tamaño **tiene asociada** las *medida base* NDA

      V ☐ F ☐

   b) El *criterio de decisión* Si IME > 15, IME= 'Muy Alto', Si 10 < IME≤ 15, IME= 'Alto', Si 5 < IME≤ 10, IME= 'Medio', Si 0 ≤ IME≤ 5, IME= 'Bajo' **se usa en** el *modelo de análisis* IME=NA/NE

      V ☐ F ☐

   c) La medida NA **tiene** la escala Ratio: enteros desde 0 a infinito y la unidad Atributo

      V ☐ F ☐

   d) La *necesidad de información* Conocer la mantenibilidad de diagramas E/R **se relaciona** el *concepto medible* Mantenibilidad y Usabilidad

      V ☐ F ☐

   e) La *entidad* Diagrama E/R **tiene** el *atributo* Longitud

      V ☐ F ☐

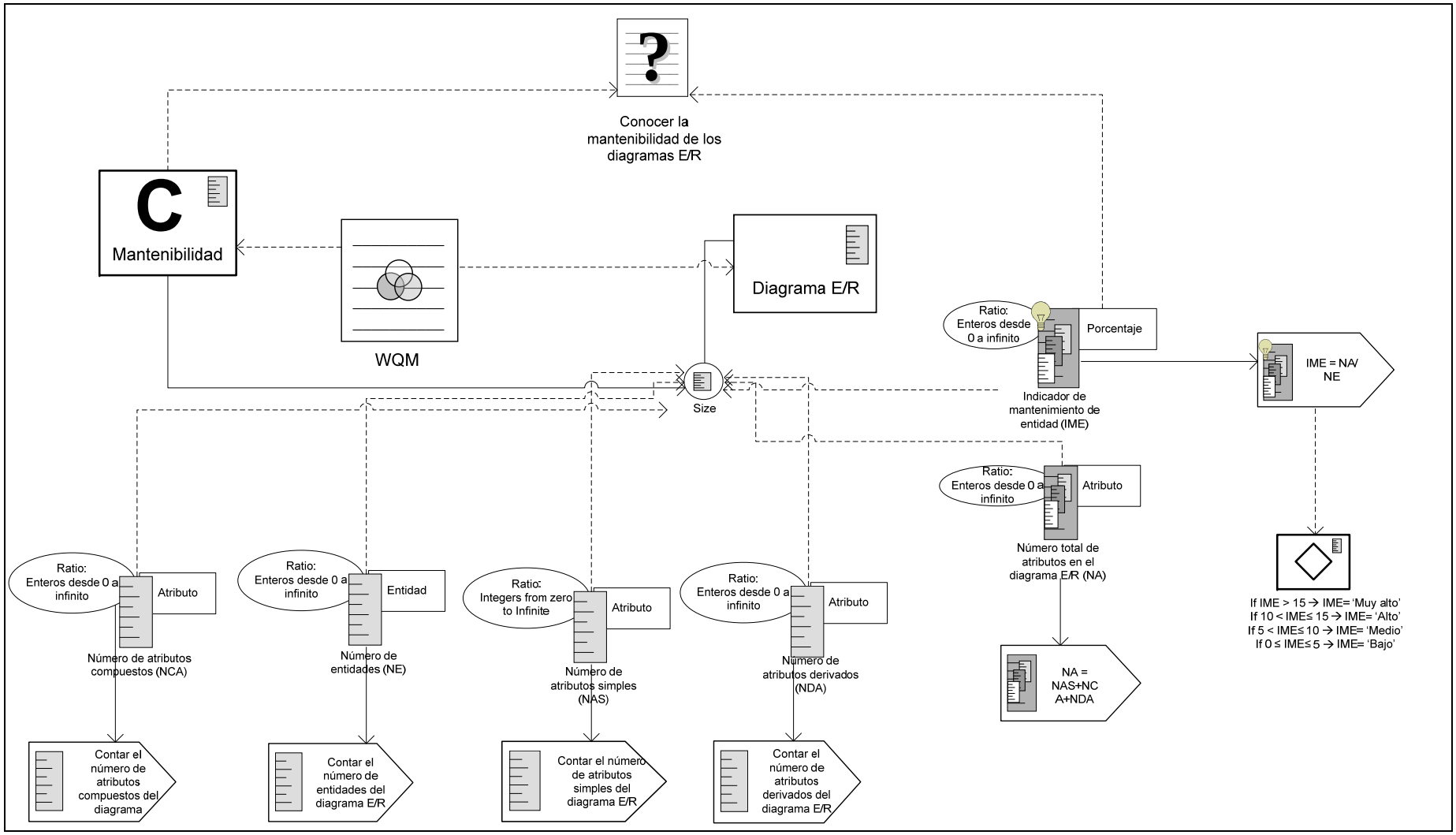
   f) La *medida derivada* NA **se calcula con** la *función de cálculo* Contar el número de tablas de un diagrama E/R

      V ☐ F ☐

Anotar la hora de fin (indique hh:mm:ss): _____

2) **Intente comprender el diagrama y valore según su criterio la COMPLEJIDAD del modelo de Medición del Software**

   Muy simple ☐ Algo simple ☐ Normal ☐ Algo complejo ☐ Muy complejo ☐


Conocer la mantenibilidad de los diagramas E/R
C
Mantenibilidad
WQM
Diagrama E/R
Size
Ratio: Enteros desde 0 a infinito
Porcentaje
Indicador de mantenimiento de entidad (IME)
IME = NA/ NE
If IME > 15 → IME= 'Muy alto'
If 10 < IME≤ 15 → IME= 'Alto'
If 5 < IME≤ 10 → IME= 'Medio'
If 0 ≤ IME≤ 5 → IME= 'Bajo'
Ratio: Enteros desde 0 a infinito
Atributo
Número total de atributos en el diagrama E/R (NA)
NA = NAS+NCA+NDA
Ratio: Enteros desde 0 a infinito
Atributo
Número de atributos compuestos (NCA)
Contar el número de atributos compuestos del diagrama
Ratio: Enteros desde 0 a infinito
Entidad
Número de entidades (NE)
Contar el número de entidades del diagrama E/R
Ratio: Integers from zero to Infinite
Atributo
Número de atributos simples (NAS)
Contar el número de atributos simples del diagrama E/R
Ratio: Enteros desde 0 a infinito
Atributo
Número de atributos derivados (NDA)
Contar el número de atributos derivados del diagrama E/R

A.4. Bloque de ejercicios de modificabilidad para el grupo de sujetos B.

EJERCICIO 1: Tareas a realizar:

Anotar la hora de inicio (indique hh:mm:ss): _____

1) **Realice las modificaciones necesarias en el diagrama SMML (indique el número de apartado en cada modificación), representado al final del ejercicio, para satisfacer los siguientes requisitos de medición del software:**

   a) Se desea eliminar *la medida base* NT, elimina si fuera necesario las medidas que usan esta *medida base*

   b) Se desea añadir la *medida base* NPK (Número de claves primaria) **definida para** el *atributo* Complejidad y que **usa** el *método de medición* contar el número de claves primarias. Con *escala* Ratio: enteros desde 0 a infinito y *unidad* clave primaria.

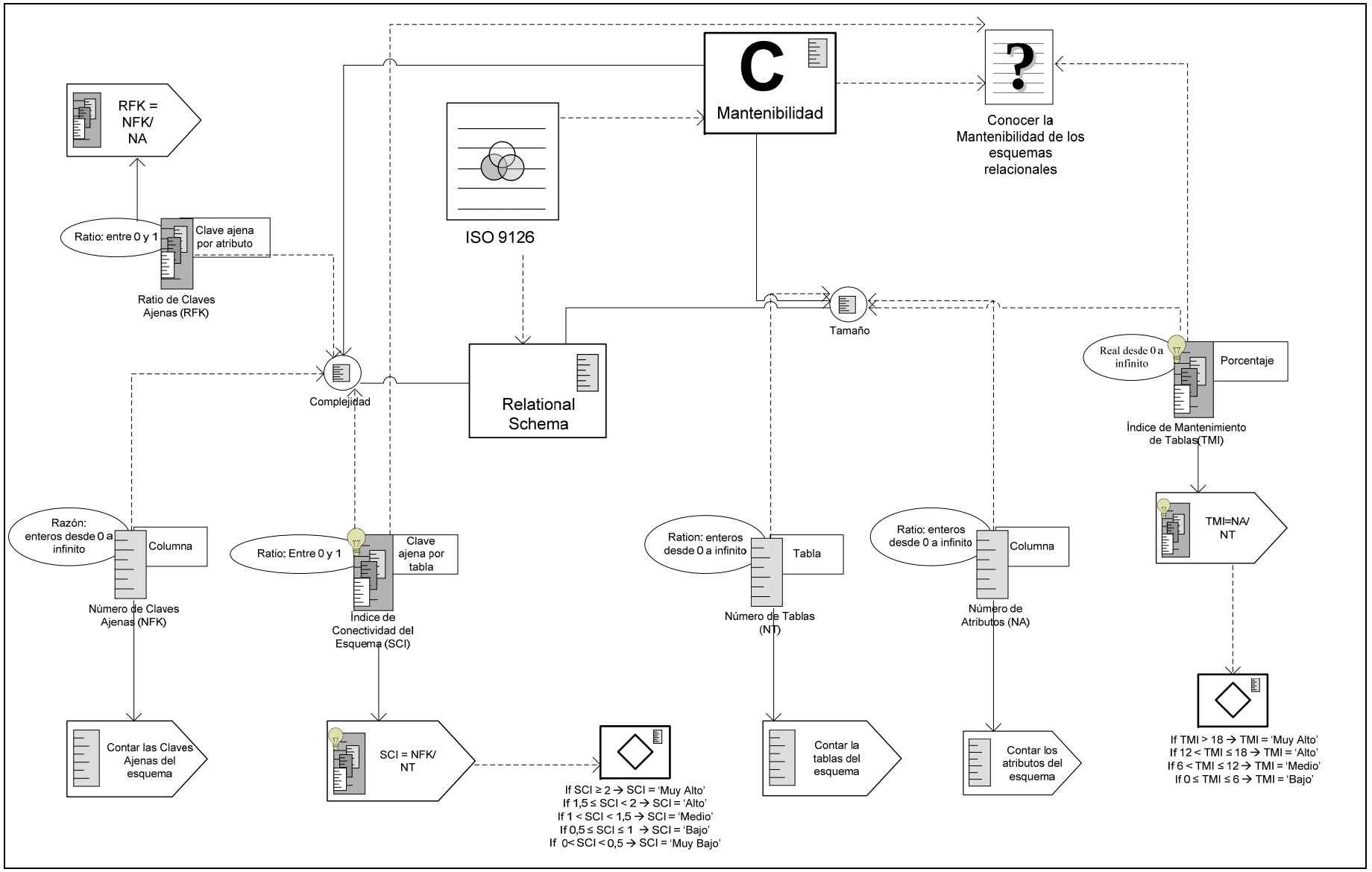
   c) Se desea modificar el *modelo de calidad* ISO 9126 para que **evalúe** los conceptos medibles Mantenibilidad *y* Portabilidad

   d) Se desea modificar la *Necesidad de información* conocer la portabilidad de los esquemas relacionales para que **se relaciona** con el *concepto medible* Portabilidad.

Anotar la hora de fin (indique hh:mm:ss): _____

2) **Intente comprender el diagrama SMML y valore según su criterio la COMPLEJIDAD del modelo de Medición del Software**

   Muy simple ☐ Algo simple ☐ Normal ☐ Algo complejo ☐ Muy complejo ☐


C
Mantenibilidad
?
Conocer la Mantenibilidad de los esquemas relacionales
RFK = NFK/ NA
ISO 9126
Ratio: entre 0 y 1
Clave ajena por atributo
Ratio de Claves Ajenas (RFK)
Tamaño
Complejidad
Relational Schema
Real desde 0 a infinito
Porcentaje
Índice de Mantenimiento de Tablas(TMI)
Razón: enteros desde 0 a infinito
Columna
Número de Claves Ajenas (NFK)
Ratio: Entre 0 y 1
Clave ajena por tabla
Índice de Conectividad del Esquema (SCI)
Ration: enteros desde 0 a infinito
Tabla
Número de Tablas (NT)
Ratio: enteros desde 0 a infinito
Columna
Número de Atributos (NA)
TMI=NA/ NT
Contar las Claves Ajenas del esquema
SCI = NFK/ NT
If SCI ≥ 2 → SCI = 'Muy Alto'
If 1,5 ≤ SCI < 2 → SCI = 'Alto'
If 1 < SCI < 1,5 → SCI = 'Medio'
If 0,5 ≤ SCI ≤ 1 → SCI = 'Bajo'
If 0< SCI < 0,5 → SCI = 'Muy Bajo'
Contar la tablas del esquema
Contar los atributos del esquema
If TMI > 18 → TMI = 'Muy Alto'
If 12 < TMI ≤ 18 → TMI = 'Alto'
If 6 < TMI ≤ 12 → TMI = 'Medio'
If 0 ≤ TMI ≤ 6 → TMI = 'Bajo'

EJERCICIO 2: Tareas a realizar:

Anotar la hora de inicio (indique hh:mm:ss): _____

1) **Realice las modificaciones necesarias en el diagrama SMML (indique el número de apartado en cada modificación), representado al final del ejercicio, para satisfacer los siguientes requisitos de medición del software:**

   a) Se desea modificar la *medida* DAC para que **utilice** la *escala* ratio: reales desde 0 a infinito.

   b) Se desea modificar el *atributo* Tamaño para que **sólo tenga asociadas** las *medidas* NOM y DAC

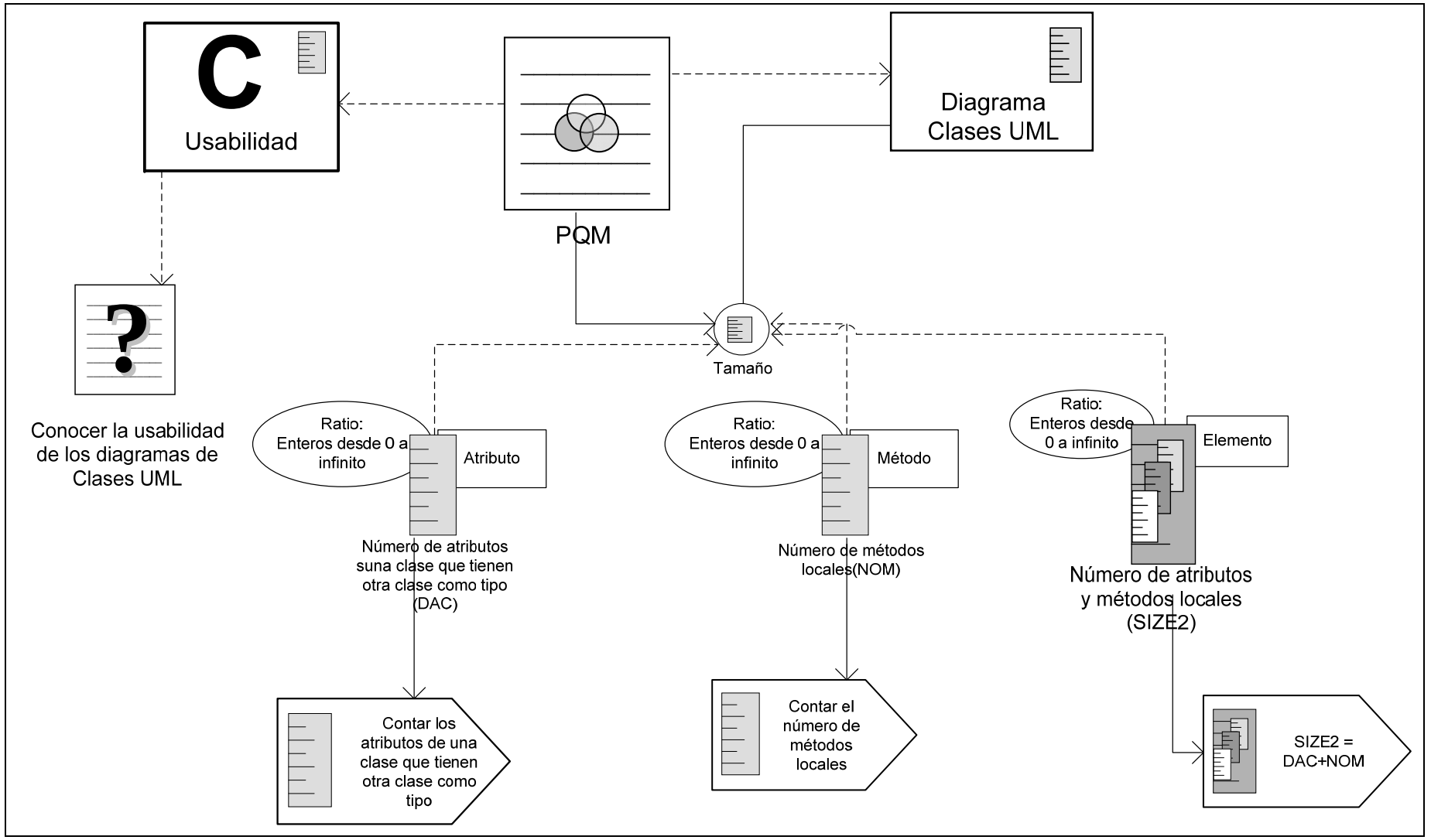
   c) Se desea modificar la *categoría de entidad* Diagrama Clases UML para que **tenga definido** el *modelo de calidad* PQM y el *modelo de calidad* WQM

   d) Se desea modificar el *concepto medible* usabilidad para que **sea evaluado** por el *modelo de calidad* PQM y por el *modelo de calidad* WQM

Anotar la hora de fin (indique hh:mm:ss): _____

2) **Intente comprender el diagrama SMML y valore según su criterio la COMPLEJIDAD del modelo de Medición del Software**

   Muy simple ☐ Algo simple ☐ Normal ☐ Algo complejo ☐ Muy complejo ☐


C
Usabilidad
PQM
Diagrama
Clases UML
?
Conocer la usabilidad
de los diagramas de
Clases UML
Tamaño
Ratio:
Enteros desde 0 a
infinito
Atributo
Número de atributos
suna clase que tienen
otra clase como tipo
(DAC)
Contar los
atributos de una
clase que tienen
otra clase como
tipo
Ratio:
Enteros desde 0 a
infinito
Método
Número de métodos
locales(NOM)
Contar el
número de
métodos
locales
Ratio:
Enteros desde
0 a infinito
Elemento
Número de atributos
y métodos locales
(SIZE2)
SIZE2 =
DAC+NOM

EJERCICIO 3 - Tareas a realizar:

Anotar la hora de inicio (indique hh:mm:ss): _____

1) **Realice las modificaciones necesarias en las tablas (indique el número de apartado en cada modificación), representadas al final del ejercicio, para satisfacer los siguientes requisitos de medición del software:**

   a) Se desea modificar el *atributo* Representación consistente para que (sólo) **esté relacionado con** los *conceptos medibles* Atracción y Representación

   b) Se desea modificar el *atributo* Representación consistente para que (sólo) **tenga asociadas** las *medidas* base LnC y LTC

   c) Se desea modificar el *concepto medible* Atracción para que (sólo) **relacione** los *atributos* Representación consistente, y Complejidad, que **pertenecen a** *la categoría de entidad* Portal Web.

   d) Se desea modificar la *necesidad de información* Conocer la representación de los portales web para que (sólo) **se relacione con** el *concepto medible* Atracción.

Anotar la hora de fin (indique hh:mm:ss): _____

2) **Intente comprender las tablas y valore según su criterio la COMPLEJIDAD del modelo de Medición del Software**

   Muy simple ☐ Algo simple ☐ Normal ☐ Algo complejo ☐ Muy complejo ☐

Especificación Textual:

Elementos del metamodelo de medición (clases MOF)	Modelo de medición (Instancias M2)	Relaciones del metamodelo de medición (Asociaciones MOF)		Modelo de medición (links M2)
Atributo	Representación consistente	Tiene	Categoría de entidad	Portal Web
			Atributo	Representación consistente
Categoría de entidad	Portal Web	Se define para	Modelo de calidad	DQ Representational
			Categoría de entidad	Portal Web
Modelo de calidad	DQ Representational	Evalúa	Modelo de calidad	DQ Representational
			Concepto medible	Entendibilidad Atracción Representación
Concepto medible	Entendibilidad Atracción Representación	Relaciona	Atributo	Representación consistente
			Concepto medible	Entendibilidad Atracción Representación
Necesidad de información	Conocer el nivel de calidad de representación de datos de un portal web	Se asocia con	Concepto medible	Entendibilidad Atracción Representación
			Necesidad de información	Conocer el nivel de calidad de representación de datos de un portal web
Medida	PgC, StPgCs, LnC, LTC, MaSS, SDCD, PSSD, LCsR	Se define para	Medida	PgC, StPgCs, LnC, LTC MaSS, SDCD, PSSD, LCsR
			Atributo	Representación consistente
Indicador	LCsR	Satisface	Indicador	LCsR
			Necesidad de Información	Conocer el nivel de calidad de representación de datos de un portal web

Medida Base	Descripción	Método de medición	Escala	Unidad
PgC	Número de páginas	Contar las páginas de un portal web	Naturales [0-∞)	Página
StPgCs	Páginas con estilos	Contar las páginas con estilos dados	Naturales [0-∞)	Página
LnC	Número de vínculos	Contar los vínculos de un portal web	Naturales [0-∞)	Vínculo
LTC	Correspondencia del vínculo del texto	Contar los vínculos con palabras comunes	Naturales [0-∞)	Vínculo

Medida Derivada	Descripción	Función de cálculo	Escala	Unidad
MaSS	Número de páginas con mayor número de estilos usados	Mass = MAX ($\forall_i$, $StPgC_i$)	Naturales [0-∞)	Página
SDCD	Grado de correspondencia origen destino	SDCD = LTC/LnC	Naturales [0-∞)	Página
PSSD	Páginas con el mismo grado de estilo	PSSD = MaSS/PgC	Naturales [0-∞)	Página

Indicador	Descripción	Modelo de análisis	Criterio de decisión	Escala	Unidad
LCsR	Nivel de representación consistente	LCsR = PSSD*0.5+SDCD*0.5	Si LCsR > 0,7 → LCsR = **'Alto'** Si 0,4 < LCsR ≤ 0,7 → TMI = **'Medio'** Si 0 ≤ LCsR ≤ 0,4 → LCsR = **'Bajo'**	Naturales [0-∞)	Página

EJERCICIO 4: Tareas a realizar:

Anotar la hora de inicio (indique hh:mm:ss): _____

1) **Realice las modificaciones necesarias en las tablas (indique el número de apartado en cada modificación), representadas al final del ejercicio, para satisfacer los siguientes requisitos de medición del software:**

   a) Se desea modificar El *atributo* Tamaño para que sólo **tenga asociadas** las *medidas* base NCA, NAS y NDA.

   b) Se desea modificar el *concepto medible* mantenibilidad para que **relacione** el *atributo* complejidad

   c) Se desea modificar la *categoría de entidad* Diagrama E/R para que **tenga** los *atributos* tamaño y complejidad

   d) Se desea añadir una nueva *medida base* NrefR (Número de relaciones reflexivas) **definida para** el *atributo* complejidad, que **usa** el *método de medición* contar el número de relaciones reflexivas del diagrama E/R, y **tiene** la Escala “naturales [0-∞]” y *Unidad* Relación

Anotar la hora de fin (indique hh:mm:ss): _____

2) **Intente comprender las tablas y valore según su criterio la COMPLEJIDAD del modelo de Medición del Software**

   Muy simple ☐ Algo simple ☐ Normal ☐ Algo complejo ☐ Muy complejo ☐

Especificación Textual:

Elementos del metamodelo de medición (clases MOF)	Modelo de medición (instancias M2)	Relaciones del metamodelo de medición (asociaciones MOF)		Modelo medición (links M2)
Atributo	Tamaño	Tiene	Categoría de entidad	Diagrama E/R
			Atributo	Tamaño
Categoría de entidad	Diagrama E/R	Se define para	Modelo de calidad	WQM
			Categoría de entidad	Diagrama E/R
Modelo de calidad	WQM	Evalúa	Modelo de calidad	WQM
			Concepto medible	Mantenibilidad
Concepto medible	Mantenibilidad	Relates	Concepto medible	Mantenibilidad
			Atributo	Tamaño
Necesidad de información	Conocer la mantenibilidad de los diagramas E/R	Se asocia con	Concepto medible	Mantenibilidad
			Necesidad de información	Conocer la mantenibilidad de los diagramas E/R
Medida	NE, NAS, NCA, NDA, NA, IME	Se define para	Medida	NE, NAS, NCA, NDA, NA, IME
			Atributo	Tamaño
Indicador	IME	Satisface	Indicador	IME
			Necesidad de información	Conocer la mantenibilidad de los diagramas E/R

Medida Base	Descripción	Método de medición	Escala	Unidad
NE	Número de entidades	Contar el número de entidades del diagrama E/R	Ratio: enteros desde 0 a infinito	Entidad
NAS	Número de atributos simples	Contar el número de atributos simples del diagrama E/R	Ratio: enteros desde 0 a infinito	Atributo
NCA	Número de atributos compuestos	Contar el número de atributos compuestos del diagrama E/R	Ratio: enteros desde 0 a infinito	Atributo
NDA	Número de atributos derivados	Contar el número de atributos derivados del diagrama E/R	Ratio: enteros desde 0 a infinito	Atributo

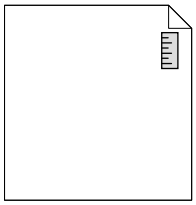
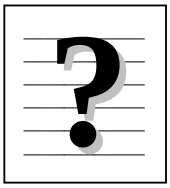
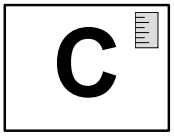
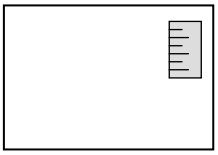
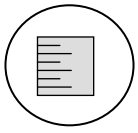
Medida Derivada	Descripción	Función de cálculo	Escala	Unidad
NA	Número total de atributos del diagrama E/R	NA = NAS+NCA+NDA	Ratio: enteros desde 0 a infinito	Atributo

Indicador	Descripción	Modelo de análisis	Criterio de decisión	Escala	Unidad
IME	Indicador de mantenimiento de entidad	IME = NA/NE	Si IME > 15 → IME= **'Very High'** Si 10 < IME≤ 15 → IME= **'High'** Si 5 < IME≤ 10 → IME= **'Medium'** Si 0 ≤ IME≤ 5 → IME= **'Low'**	Ratio: enteros desde 0 a infinito	Porcentaje

A.5. Cuestionario para valorar los elementos de SMML.

Valoración de SMML: Valore del 1 (muy adecuado) al 5 (muy inadecuado) los símbolos utilizados para los elementos y relaciones del lenguaje SMML.

Paquete Caracterization and Objetives *(Caracterización y Objetivos)*

Elemento	Icono	Definición	Valoración	Comentarios
Description *(Descripción)*		Descripción del elemento de medición en concreto		
Information Need *(Información de necesidad)*		Información necesaria para gestionar un proyecto (sus objetivos, hitos, riesgos y problemas).		
Measurable Concept *(Concepto medible)*		Relación abstracta entre *Attributes* y *Information Need*.		
Entity Class *(Categoría de entidad)*		Una colección de *entities* caracterizadas por satisfacer un cierto predicado común. Es el dominio.		
Attribute *(Atributo)*		Una propiedad mensurable, física o abstracta, que comparten todas las *Entity Class*.		

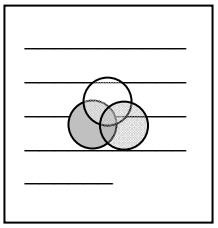
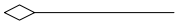
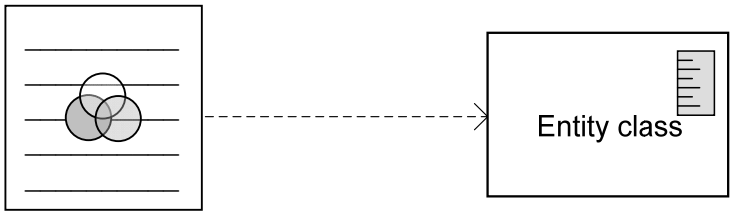
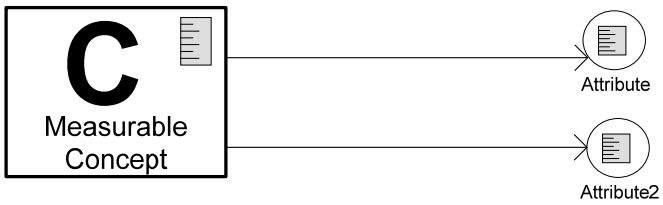
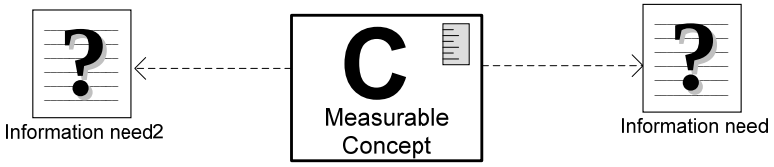
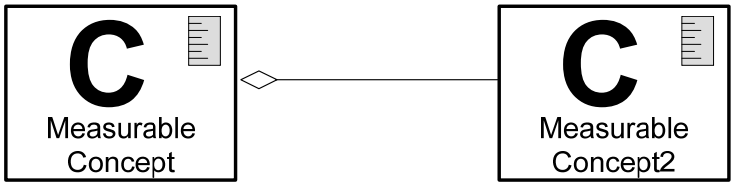
Elemento	Icono	Definición	Valoración	Comentarios
Quality Model *(Modelo de calidad)*		Un conjunto de *Measurable Concept* y relaciones entre ellos que proporciona la base para especificar requisitos de calidad y evaluar la calidad de *entity class*		

Tabla A - 1. Elementos del paquete “Caracterización y Objetivos”

Relación	Icono	Descripción	Valoración	Comentarios
Incluyes *(Incluye)*		Una *Entity Class* puede **incluir** una o varias *Entity Class*, y puede **estar incluida** en una o varias *Entity Class*.		
				
Defined for *(Definido para)*		Un *quality Model* **está definido para** una determinada *entity class*. Una *entity class* puede **tener definidos** varios *quality model*.		
				
Evaluates *(Evalúa)*		Un *quality model* **evalúa** uno o varios *measurable concept*. Un *measurable concept* **es evaluado** por uno o más *quality model*.		

Relación	Icono	Descripción	Valoración	Comentarios
				
Relates *(Relaciona)*		Un *measurable concept* **relaciona** uno o más *attributes*. Un *attribute* **está relacionado** con uno o más *measurable concept*.		
				
Is associated with *(Está asociado con)*		Un *measurable concept* **está relacionado con** una o varias *information need*. Una *information need* **se relaciona con** un *measurable concept*.		
				
Includes *(Incluye)*		Un *measurable concept* puede **incluir** a varios *measurable concept* y puede **estar incluido** en otros *measurable concept*.		
				
Has *(Tiene)*		Una *entity class* **tiene** uno o varios *attributes*. Un *attribute* solo puede pertenecer a una *entity class*.		

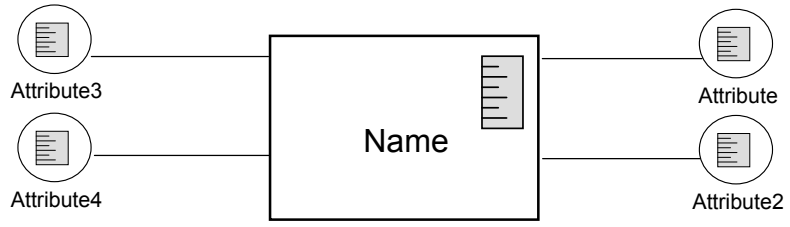
Relación	Icono	Descripción	Valoración	Comentarios
				

Tabla A - 2. Relaciones del paquete “Caracterización y Objetivos”

Paquete Software Measures *(Medidas Software)*

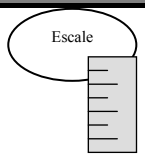
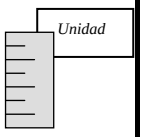
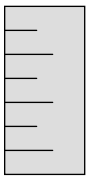
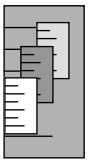
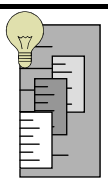
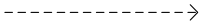
Elemento	Icono	Definición	Valoración	Comentarios
Scale *(Escala)*		Un conjunto de valores con propiedades definidas.		
Unit of measurement *(Unidad de medición)*		Una cantidad particular, definida y adoptada por convención, con la que se puede comparar otras cantidades de la misma clase para expresar sus magnitudes respecto a esa cantidad particular		
Base Measure *(Medida Base)*		Una medida de un *attribute* que no depende de ninguna otra medida, y cuya *measurement approach* es un *measurement method*.		
Derived measure *(Medida derivada)*		Una medida que es derivada de otra *base measure* o *derived measure*, utilizando una *función de cálculo* como *forma de medir*.		
Indicator *(Indicador)*		Una medida que es derivada de otras medidas utilizando un *analysis model* como *measurement approach*.		

Tabla A - 3. Elementos del paquete Medidas Software

Relación	Icono	Descripción	Valoración	Comentarios
Defined for *(Definido para)*		Una *medida (base measure, derived measure e indicator)* **está definida para** uno o más *attributes*. Un *attribute* puede tener varias *medidas* asociadas.		

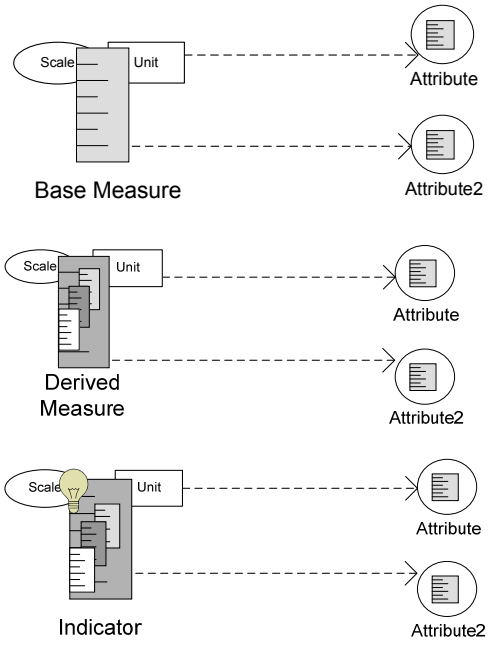


Tabla A - 4. Relaciones del paquete Medidas Software

Paquete Measurement approaches *(Formas de Medir)*

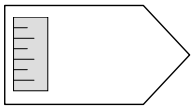
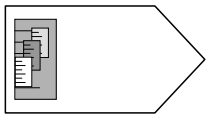
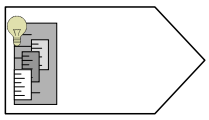
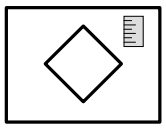
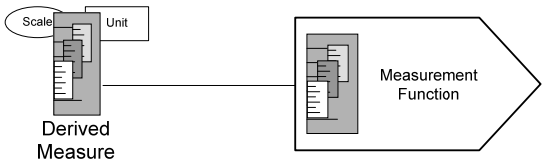
Elemento	Icono	Definición	Valoración	Comentarios
Measurement Method *(Método de medición)*		La *forma de medir* una *based measure*. Secuencia lógica de operaciones, descritas de forma genérica, usadas para realizar mediciones de un atributo respecto de una escala específica.		
Measurement Function *(Función de cálculo)*		La *forma de medir* una *derived measure*. Algoritmo o cálculo realizado para combinar dos o más medidas base y/o derivadas.		
Analysis Model *(Modelo de análisis)*		La *forma de medir* un *indicator*. Algoritmo o cálculo realizado para combinar una o más medidas (base, derivadas o indicadores) con *decision criteria* asociados.		
Decision Criteria *(Criterio de decisión)*		Valores umbral, objetivos, o patrones, usados para determinar la necesidad de una acción o investigación posterior, o para describir el nivel de confianza de un resultado dado.		

Tabla A - 5. Elementos del paquete Formas de medir.

Relación	Iconos	Descripción	Valoración	Comentarios
Calculated with *(Calculado con)*		Una *derived measure* es **calculada con** una *measurement function*. Una *measurement function* define una o más *derived measures*.		
				
Calculated with *(Calculado con)*		Un *indicator* es **calculado con** un *analysis model*. Un *análisis model* puede definir uno o más *indicators*.		

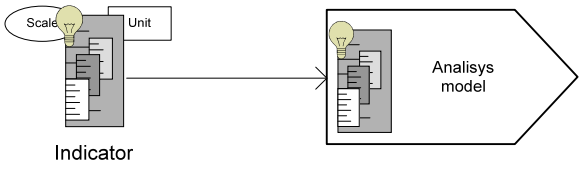
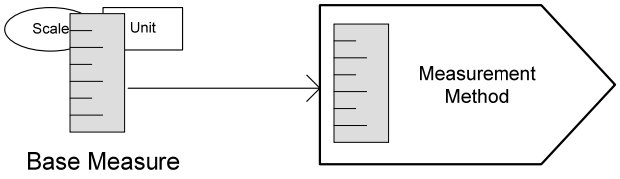
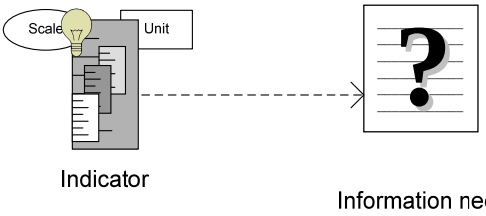
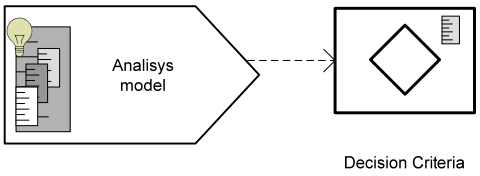
				
Uses *(Usa)*	→	Una *base measure* **usa** un *measurement method*. Un *measurement method* define una o más *base measures*.		
				
Satisfies *(Satisface)*	⇢	Un *indicador* **satisface** *information need*. Una *information need* se **satisface** con uno o más *indicators*.		
				
Uses	⇢	Un *análisis model* **usa** uno o más *decision criteria*. Un *decision criteria* puede **usarse** en uno o más *analysis model*.		
				

Tabla A - 6. Relaciones del paquete Formas de Medir

B. Gráficas Estadísticas.

En este anexo se presentan las gráficas del experimento de usabilidad de SMML que no se han mostrado en el Capítulo 5

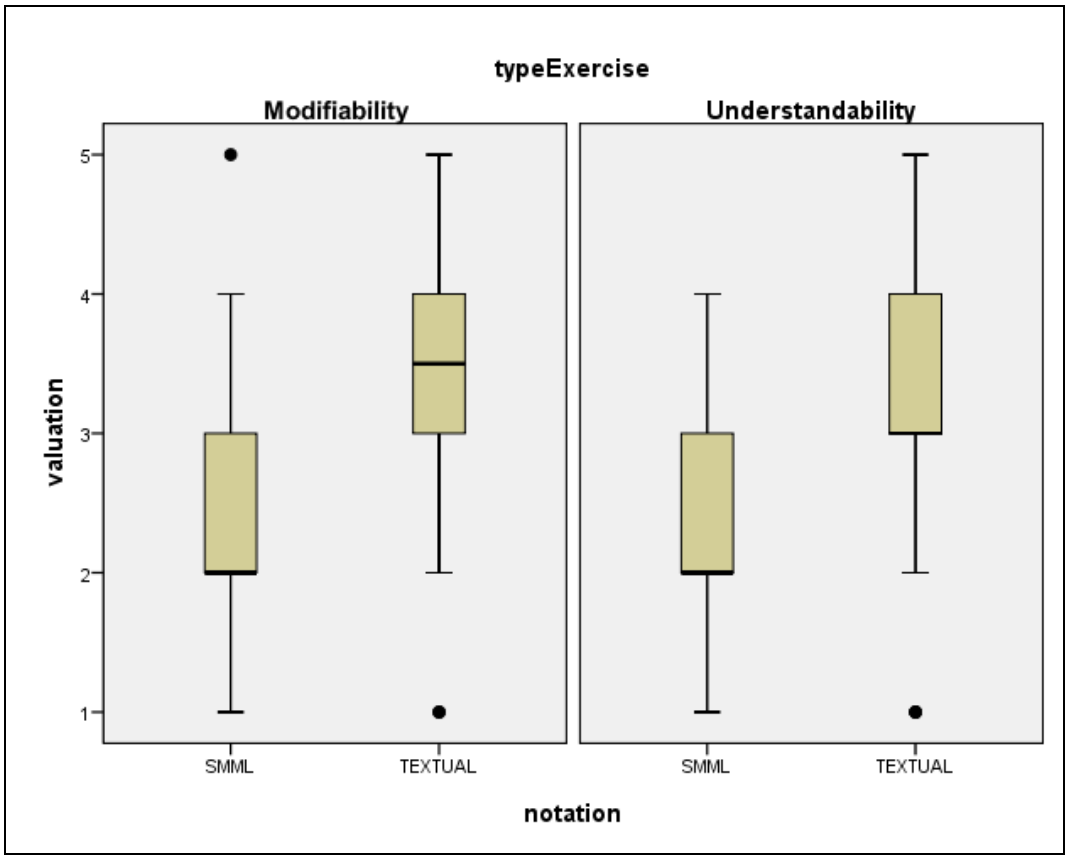


Figura B - 1. Diagrama de cajas de la interacción del UoD x uso de SMML en el experimento para la valoración subjetiva de la modificabilidad y la entendibilidad

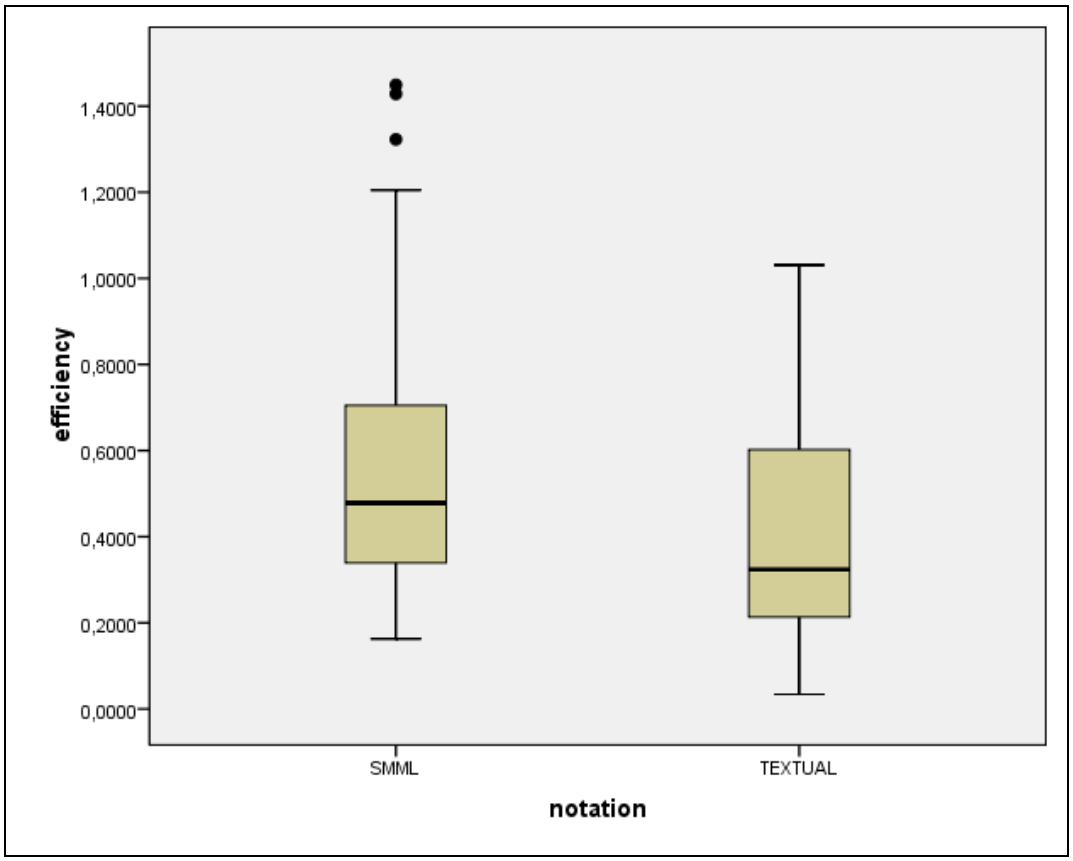


Figura B - 2. Diagrama de cajas del uso de SMML en el experimento para la eficiencia

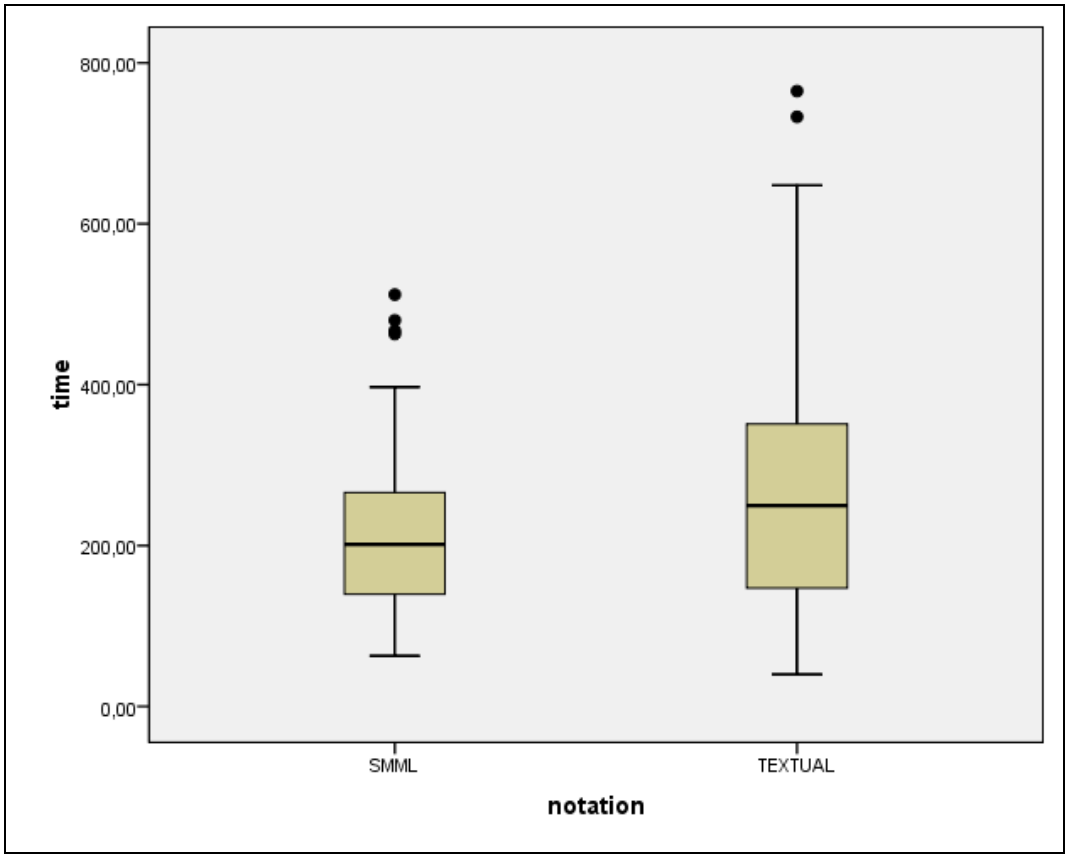


Figura B - 3. Diagrama de cajas del uso de SMML en el experimento para el tiempo

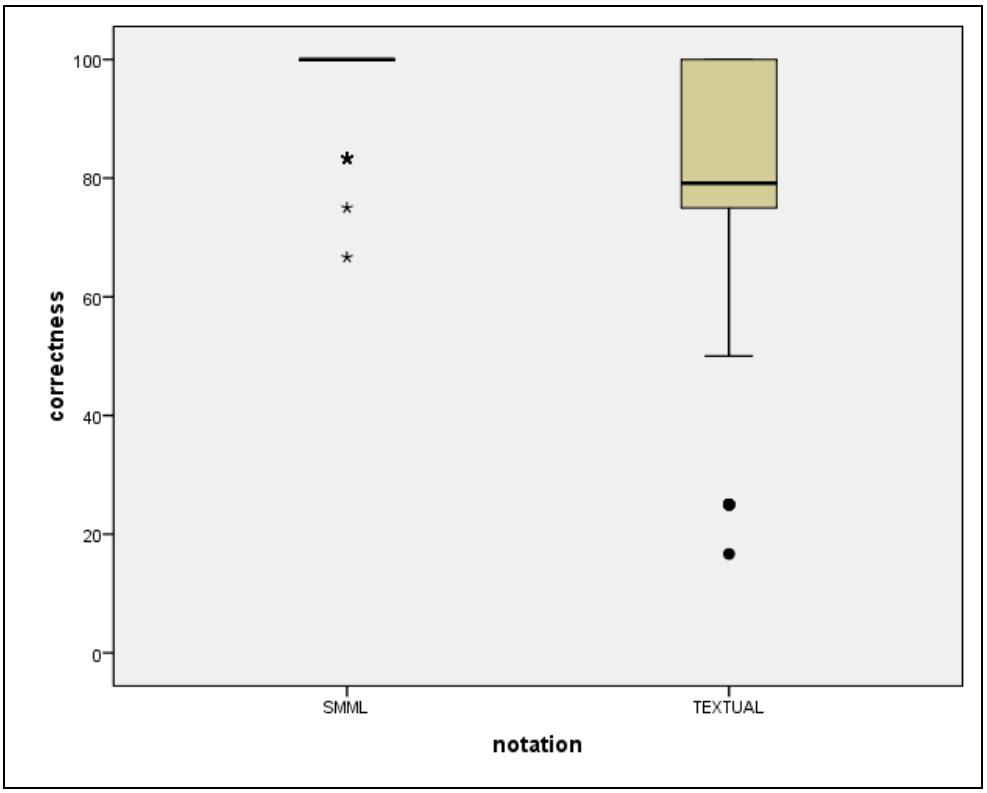


Figura B - 4. Diagrama de cajas del uso de SMML en el experimento para la exactitud

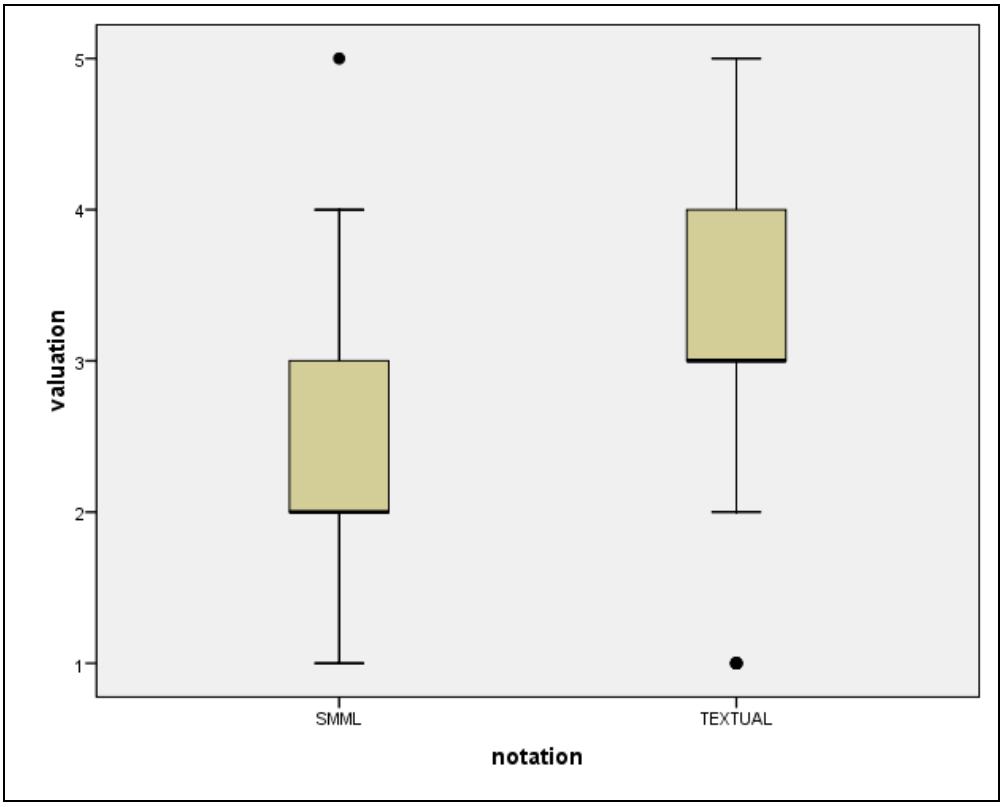


Figura B - 5. Diagrama de cajas del uso de SMML en el experimento para la valoración

C. Protocolo de Revisión Sistemática.

En este anexo se presenta la revisión sistemática realizada par

Una revisión sistemática en una tesis doctoral debe identificar la información existente para el trabajo del alumno investigador y ubicar la investigación en el estado actual del conocimiento de la materia.

Para llevar a cabo la revisión sistemática se han utilizado como modelos las revisiones de (Kitchenham, 2004) y (Biolchini et al., 2005)

El proceso va a ser iterativo. Un vez que se haya formulado la pregunta se entra en dos ciclos iterativos, el primero compuesto por las siguientes etapas:

- Selección de fuentes y estudios
- Evaluación de la planificación

Y el segundo:

- Ejecución de la revisión
- Evaluación de la ejecución

La selección de fuentes y estudios de la siguiente etapa se verá afectada por la experiencia adquirida en la selección de fuentes y estudios de la etapa anterior, esta experiencia se obtiene a partir de la evaluación de la planificación donde se decidirá si las fuentes y estudios encontrados aplicando una metodología han sido útiles o necesita realimentarse (véase Figura C - 1). Lo mismo para la ejecución de la revisión (véase Figura C - 2).

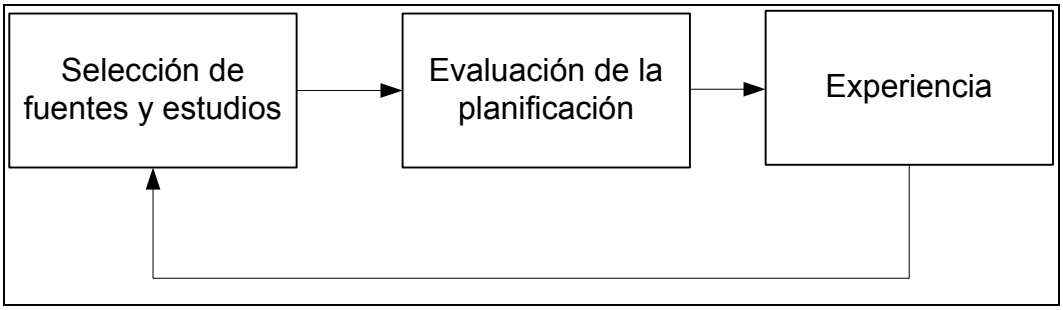


Figura C - 1. Selección de fuentes y estudios

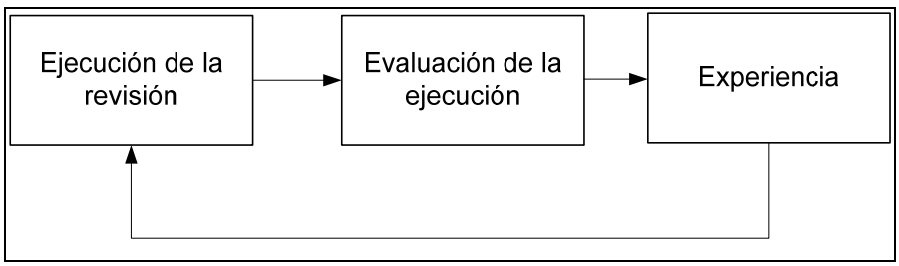


Figura C - 2. Ejecución de la revisión

C.1. Planificación de la revisión.

En este apartado, se definirán los objetivos de la investigación y la forma en la que se va a llevar a cabo la revisión. Incluye la formulación de las cuestiones de la investigación para planear como se lleva a cabo las fuentes y la selección de estudios.

C.1.1. Formulación de la pregunta.

En esta sección se definirán los objetivos de la investigación de forma clara. Incluye los siguientes apartados: foco de la pregunta y calidad y amplitud de la pregunta.

Foco de la pregunta

Se quiere desarrollar un informe técnico sobre un estudio comparativo entre los lenguajes de definición dentro de MDA y los Domain Specific Language, para ello es necesario hacer un estudio a fondo de cada uno de los lenguajes. El objetivo de esta revisión se centrará en abarcar los lenguajes de definición dentro de MDA para completar dicho estudio. El objetivo también se centra en los metamodelos de MDA: UML 2.0 y MOF.

Calidad y amplitud de la pregunta

- **Problema:** definición de lenguajes de modelos dentro de MDA.
- **Pregunta:** la pregunta que debe responder la revisión sistemática es a la definición de lenguajes de modelos dentro de MDA en general y a cada uno de los lenguajes de definición utilizados dentro de MDA en particular, es decir UML 2.0 y MOF.
- **Palabras clave y sinónimos:**
  - MDA, UML, MOF, metamodel, model, modeling.
- **Control:** el investigador no dispone de datos iniciales de los que partir aunque si de la plantilla del informe técnico sobre la definición de lenguajes de modelos dentro de MDA propuesto por su director y co-director de tesis.
- **Efecto:** lo que se espera a partir de esta revisión sistemática es obtener de forma detallado todo lo relacionado con la definición de lenguajes de modelos dentro de MDA.
- **Grupo:** el grupo está formado por un investigador principal y 3 investigadores expertos en *Model Driven* (Mario Piattini, Francisco Ruiz, Félix García).
- **Aplicación:** fundamentalmente los beneficios de la revisión están destinados a la tesis del investigador Principal, cuyo proyecto de investigación se basa en la medición del software.
- **Diseño experimental:** no se ha aplicado ningún método estadístico.

C.1.2. Selección de fuentes.

En esta sección estudiaremos las fuentes primarias de la revisión.

Definición del criterio de selección de fuentes

Se han utilizado las bibliotecas digitales (*digital library*) ya que permiten la búsqueda de artículos del dominio que interesa. En estos buscadores existes 2 tipos de búsqueda, las búsquedas básicas y las avanzadas que permiten el uso de expresiones regulares mediante operadores lógicos como AND, OR, NOT, “”, W/__, etc.

Lengua del estudio

Inglés

Identificación de fuentes

- **Métodos de búsqueda de fuentes:** búsqueda a través de las bibliotecas digitales
- **Cadenas de búsqueda:** en todas las fuentes seleccionadas aplicaremos las siguientes cadenas de búsqueda.
  - "MDA AND (UML OR MOF)": en los buscadores como IEEE y ACM que no permiten expresiones compuestas, se harán dos búsquedas equivalentes a la primera: MDA AND UML y MDA AND MOF.
  - "MDA AND Modeling": esta cadena de búsqueda se ha utilizado únicamente en los 3 primeros buscadores: Science@Direct, ACM y Wiley, y viendo que no aportaba estudios relevantes adiciones se ha decidido suprimir en el último buscador.
  - "MDA AND Metamodeling" y "MDA AND Languages AND Modeling": estas cadenas de búsqueda se han utilizado únicamente en Science@Direct. Al comprobar que no aportaban estudios relevantes y que prácticamente los estudios estaban repetidos se ha decidido suprimir estas cadenas de búsqueda.
- **Lista de fuentes:** a continuación se expondrán todas las fuentes utilizadas indicando los tipos, las restricciones y los criterios de búsqueda.

  - **Science@direct. http://www.sciencedirect.com**

  Consejos de búsqueda avanzada:

    - *Conectores lógicos:* AND, OR, NOT
    - *Whithin:* el conector W/nn especifica la proximidad en palabras entre otras dos. Así MDA W/15 MOF nos indica que entre estas dos palabras no debe haber más de 15. También se puede utilizar NOT W/nn
    - *Precedes:* el conector PRE/nn indica que la primera palabra tiene que preceder a la segunda en un número determinado de palabras.
    - *W/SEG:* indica que las palabras de su izquierda y derecha aparecen en el mismo campo del artículo
    - *Búsqueda de frase exacta:* encerrada entre dobles comillas.
    - *Expresiones regulares:* permite expresiones regulares mediante el uso de paréntesis y conectores lógicos. MDA AND (MOF OR UML).

  - **ACM Digital Library. http://portal.acm.org**

  Consejos de búsqueda avanzada:

    - Sensible a las mayúsculas y minúsculas cuando los términos están sueltos. Por ejemplo NeXT buscará páginas que contengan NeXT pero no next ni NEXT.
    - Búsquedas de frases literales encerradas entre comillas. Aquí no es sensible a mayúsculas ni minúsculas.
    - Exclusión de palabras: se hace mediante el operador -, así en la búsqueda MDA – Model se encontrará aquellas páginas que contengan MDA pero no Model.
    - Inclusión de palabras: el signo + indicará que las palabras o frases indicadas sean requisito en las páginas relevantes. Así la cadena de búsqueda +MDA + "Model

Driven Architecture" buscará páginas que contengan MDA y "Model Driven Architecture".

- No permite expresiones regulares compuestas como MDA AND (MOF OR UML), en este caso habría que hacer dos búsquedas avanzadas: MDA AND MOF y MDA AND UML.

- **Wiley InterScience. http://www3.interscience.wiley.com/**

Consejos de búsqueda avanzada:

- *Operadores booleanos:* AND, OR y NOT. En vez de utilizar OR se puede utilizar la coma, de esta forma UML OR MOF equivale a UML, MOF
- *Proximidad:* indicado mediante NEAR/númeroLineas,
- *Comodín:* mediante el *. Así meta* buscará documentos que contengan palabras que empiecen por meta.
- *Búsqueda de frase exacta:* encerrada entre dobles comillas.
- *Expresiones regulares:* permite expresiones regulares mediante el uso de paréntesis y conectores lógicos. MDA AND (MOF OR UML).

- **IEEE Digital Library.**

  http://www.computer.org/portal/site/csdl/index.jsp

Consejos de búsqueda avanzada:

- *Operadores booleanos:* AND, OR y NOT.
- *Proximidad:* indicado mediante NEAR/númeroLineas.
- *No permite expresiones regulares*

Selección de fuentes después de la evaluación

Todo el listado de fuentes ha satisfecho el criterio de calidad.

Comprobación de referencias

Todas las fuentes fueron probadas.

C.1.3. Selección de estudios.

Una vez que las fuentes están definidas, es necesario describir el proceso y el criterio para la selección y evaluación de estudios.

C.1.3.1. Definición de estudios.

- **Definición de los criterios de inclusión y exclusión de estudios:** Los estudios deben presentar los lenguajes de definición de Modelos de MDA tales como UML y MOF.
- **Definición de tipos de estudio:** se seleccionará todos los tipos de estudios relacionados con el tema de investigación, es decir el modelado en MDA, MOF y UML 2.0 en MDA.

- **Procedimientos para la selección de estudios:** las cadenas de búsqueda deben funcionar en todas las fuentes seleccionadas. Inicialmente, para seleccionar un conjunto de estudios, nos basaremos únicamente en el abstract de todos los estudios de la web y se evaluará de acuerdo con los criterios de inclusión y exclusión.

C.1.3.2. Ejecución de la selección.

La base de esta sección es registrar los procesos de selección de estudios primarios y realizar el informe de los estudios y los resultados obtenidos de su evolución.

- Selección de estudios iniciales:

  Para seleccionar los estudios primarios aplicaremos las cadenas de búsquedas en cada una de las fuentes seleccionadas. En cada fuente se tendrá en cuenta las propiedades de las cadenas de búsqueda. Los estudios encontrados en cada fuente se indicarán en tablas.

  En cada cadena de búsqueda hemos analizado los artículos que se han considerado relevantes, basándose en el punto 2.3.1., es decir, tomamos los estudios a partir del título y del *abstract*. Obtenidos los relevantes se analizan en detalle leyendo todo el estudio, de esta forma se obtendrán los estudios primarios.

  En cada búsqueda habrá que comprobar con las búsquedas anteriores que no hay artículos repetidos.

Cadena de búsqueda	Encontrados	No repetidos	Relevantes	Estudios Primarios
MDA AND (UML OR MOF)	15	0	5	0
MDA AND Modeling				
MDA AND Metamodeling	4	3	0	0
MDA AND Languages AND Modeling	6	1	0	0

Tabla C - 1. Extracción de los estudios encontrados en Science@Direct

Cadena de búsqueda	Encontrados	No repetidos	Relevantes	Estudios primarios
+MDA +UML	137	137	19	6
+MDA +MOF	40	2	0	0
+MDA +metamodeling	58	3	0	0

Tabla C - 2. Extracción de los estudios encontrados en ACM

Cadena de búsqueda	Encontrados	No repetidos	Relevantes	Estudios primarios
MDA AND (UML OR MOF)	3	0	5	0
MDA MODELING	4	3	0	0

Tabla C - 3. Extracción de los estudios encontrados en Wiley InterScience

Cadena de búsqueda	Encontrados	No repetidos	Relevantes	Estudios primarios
MDA AND UML	100	99	10	5
MDA AND MOF	50	16	0	0

Tabla C - 4: extracción de los estudios encontrados en IEEE

C.1.4. Extracción de la información.

Una vez que los estudios primarios han sido seleccionados, comienza la extracción de la información relevante. En esta sección, se describen el criterio de extracción y los resultados.

Definición del criterio de inclusión y exclusión de información

La información relativa a nuestro estudio es lo relacionado con el modelado dentro de MDA, sin faltar los lenguajes de definición de modelos.

Excluiremos aquellos estudios que aún siendo relevantes la información ya esté revisado en estudios anteriores.

Informes de extracción de datos

Para estandarizar la forma en la que la información se va a representar el investigador debe crear informes para recolectar todos los datos de los estudios seleccionados. Estos informes deben variar dependiendo del objetivo y el contexto de la revisión sistemática

Ejecución de la extracción

A partir de la selección de estudios se pueden obtener dos tipos de resultados: resultados objetivos y subjetivos.

- **Extracción de resultados objetivos:** los resultados objetivos son aquellos que se pueden obtener directamente de los estudios seleccionados. Estos resultados se pueden organizar de la manera siguiente:
  - **a) Identificación del estudio:** incluye el título de la publicación, el autor y la fuente de donde se ha obtenido.
  - **b) Fecha de la revisión de datos:** fecha en la que se ha extraído el estudio.
  - **c) Resumen del estudio:** resumen del estudio.
  - **d) Información relevante del estudio para el investigador:** de los estudios encontrados no todo el contenido o la mayoría del estudio es importante para el investigador.
  - **e) Peso:** valoración del 1 al 10 de la relevancia que tiene el estudio para el informe técnico.
- **Extracción de resultados subjetivos:** son aquellos que no se pueden obtener directamente de los estudios seleccionados. Existen dos formas de obtener tales resultados:
  - **a) Información a través de los autores:** consiste en que el revisor preguntar a los autores para resolver dudas.
  - **b) Abstracciones e impresiones generales:** donde los revisadotes obtienen sus propias conclusiones una vez leído el estudio.

A continuación por cada estudio extraído detallaremos los resultados objetivos y subjetivos.

Identificación del estudio	Oliver, I., *Applying UML and MDA to Real Systems Desing*, Finland, 2005
Fecha de la extracción de datos	03 de mayo de 2006
Resumen del estudio	Tradicionalmente el diseño de sistemas por medio de cajas negras de funcionalidad, a pesar de ser tan conocida presenta desventajas como incompatibilidad entre distintas arquitecturas, difícil mantenimiento, reutilización y depuración del código. El autor del estudio defiende ideas tales como la Arquitectura Basada en Modelos (*Model Based Architecture*) del OMG o Ingeniería Basada en Modelos (*Model Based Engineering* – MBE) y la utilización del UML para enfrentar los numerosos cambios de la tecnología actual.
Información relevante del estudio para el investigador	Se defiende la utilización de UML y MDA para el diseño de sistemas reales. MDA se expresa mediante los lenguajes UML y MOF, está basado en conceptos de modelos de transformaciones entre modelos. Los modelos se conocen como PIM (modelo independiente de la plataforma) y PSM (modelo específico de la plataforma). El autor habla también de la variedad de lenguajes de modelos dependiendo del nivel de abstracción. Así se tendrán numerosos perfiles dentro de UML. Con respecto a los perfiles de UML existe un problema de falta de coherencia y herramientas de soporte, pero el principal problema es que los conceptos en estos lenguajes son a menudo ambiguos incluso en una plataforma independiente o en un nivel genérico. En la aplicación de UML el problema encontrado es que la OO no se entiende correctamente y no se aplica de forma adecuada y no siempre los modelos contienen información relevante.
Peso	6
Identificación del estudio	Hnetykna, P., Plasil, F., *Distributed Versioning Model for MOF*
Fecha de la extracción de datos	5 de mayo de 2006
Resumen del estudio	Este artículo describe el DVM (Distributed Versioning Model) para MOF.
Información relevante del estudio para el investigador	Los puntos relevantes del artículo es el primero, la introducción, donde habla de MDA y MOF. Describe los principales objetivos de MDA: portabilidad, interoperabilidad y reutilización. Para alcanzar estos objetivos se proponen diversas herramientas de manipulación y modelos para utilizar en desarrollo del software como por ejemplo MOF y sus repositorios, herramientas de diseño UML, herramientas de soporte para transformaciones de modelos, etc. El artículo define MOF como lenguaje abstracto y marco de trabajo para la especificación, construcción y control de los metamodelos. MOF define un marco de trabajo para implementar repositorios que sostengan los meta-datos descritos por un meta-modelo. MOF especifica el formato XMI para intercambio de meta-datos entre repositorios. MOF está basado en una arquitectura de 4 capas, por convención están denotados como M0, M1, M2 y M3. La capa más baja es M0 y se corresponde con la capa de información que contiene la implementación de las entidades, los datos. La capa modelo, M1, contiene el modelo en particular, describe las entidades. La

	capa meta-modelo M2 determina la estructura y semántica del modelo en particular. Finalmente, la capa meta-meta-modelo, M3, describe la estructura y semántica del lenguaje abstracto MOF.
Peso	7
Identificación del estudio	Asuman, J., H., Kent, S., *Visualizing Model Mappings in UML*.
Fecha de la extracción de datos	5 de mayo del 2006
Resumen del estudio	El artículo propone una extensión de UML para expresar transformaciones entre modelos utilizando diagramas. Además ilustra como se puede utilizar esta extensión para el meta-modelado.
Información relevante del estudio para el investigador	Para el trabajo del investigador, se ha seleccionado únicamente la parte que hace referencia al meta-modelado dentro de MDA. El OMG utiliza el meta-modelado para definir lenguajes de modelado. El meta-modelado implica la construcción de un modelo de objeto de sintaxis y semántica del lenguaje, utilizando los diagramas de UML. El OMG ha definido MOF como un subconjunto de UML utilizado para el meta-modelado.
Peso	5
Identificación del estudio	Hearnden, D., Raymong, K., Steel, J., *MQL: a Powerful Extension to OCL for MOF Queries*
Fecha de la extracción de datos	8 de mayo del 2006
Resumen del estudio	El artículo presenta una visión por encima de las motivaciones para el desarrollo de MQL y también trata su sintaxis abstracta presentada como un modelo de MOF y su semántica.
Información relevante del estudio para el investigador	MDA precisa que todos sus meta-modelos se definan en MOF. Estos meta-modelos incluye UML, CWNM y MOF. El artículo describe en uno de sus puntos las características principales de MOF en Orientación a objetos: encapsulación, herencia y polimorfismo.
Peso	6
Identificación del estudio	Fenstermacher, K., D., *If I had a Model, I'd Model in the Mornin'*, university of Arizona.
Fecha de la extracción de datos	9 de mayo del 2006
Resumen del estudio	El artículo trata de la importancia del modelado, no es sólo importante en el desarrollo del software sino que también es objeto de estudio en muchos centros de investigación como base del MDA.
Información relevante del estudio para el investigador	La parte que más ha interesado al investigador es el punto que se centra en el modelado. Define modelo y modelado. Además identifica el rol del modelado en el desarrollo.

Peso	6
Identificación del estudio	Selic, B., *Tutorial H2: An overview of UML 2.0*, Rational Software Canada.
Fecha de la extracción de datos	10 de mayo del 2006
Resumen del estudio	Este artículo trata por encima los aspectos más importantes de la primera revisión más importante de UML 2.0.
Información relevante del estudio para el investigador	Interesa prácticamente todo el artículo puesto que resume la revisión de UML 2.0.
Peso	9
Identificación del estudio	Schattkowsky, T., *UML 2.0 – Overview and Perspectivas in SoC Design*, Paderborn, Germany.
Fecha de la extracción de datos	10 de mayo del 2006
Resumen del estudio	UML ha llegado a ser el estándar del software para sistemas de modelado. Aunque UML se define como un lenguaje de propósito general su aplicación para el hardware y para el diseño del hardware/software es muy limitado. Para la aplicación exitosa en estos campos, es esencial comprender sus capacidades y transformarlo en un nuevo dominio.
Información relevante del estudio para el investigador	Características del UML. El investigador ha tomado aquellos aspectos de UML que no han sido comentados en estudios anteriores.
Peso	8

Tabla C - 5: Fuentes primarias de ACM para la búsqueda +MDA +UML

Identificación del estudio	Wang, H., Zhang, D., *MDA-based Development of E-Learning System*, Carleton University, 2003.
Fecha de la extracción de datos	15 de mayo del 2006
Resumen del estudio	En el artículo propone la aproximación del model driven al sistema de desarrollo E-Learning basado en estándares del OMG, maximizando las facilidades proporcionadas por MDA.
Información relevante del estudio para el investigador	La parte más revelante en nuestro estudio ha sido la referente a los modelos y metamodelos dentro de MDA. De nuevo se habla de los niveles de abstracción de MOF, el meta-metamodelo. El artículo destaca que un aspecto clave de MDA es la habilidad para definir lenguajes especializados para varios aspectos del sistema y para diferentes niveles de abstracción.
Peso	7

Identificación del estudio	Khan, M., U., Geihs, K., Gutbrodt, F., Göhner, P., Trauter, R., AG, D., *Model-Driven Development of Real-Time Systems with UML 2.0 and C*, Germany, 2006.
Fecha de la extracción de datos	15 de mayo del 2006
Resumen del estudio	El artículo explica como llevar a cabo el desarrollo dirigido por modelos de sistemas basados en tiempo real utilizando UML 2.0 y C.
Información relevante del estudio para el investigador	Para el investigador la parte relevante es el punto que habla del modelado con UML 2.0. Explica el sistema de modelado de UML 2.0 además de sus requisitos, comportamiento, interacción y arquitectura de modelado.
Peso	7
Identificación del estudio	Selic, B., *Model-Driven Development of Real-Time Software Using OMG Standars,* Canada, 2003.
Fecha de la extracción de datos	16 de mayo del 2006
Resumen del estudio	Resume el desarrollo dirigido por modelos de Software basados en tiempo real utilizando los estándares del OMG.
Información relevante del estudio para el investigador	Para empezar se ha tenido en cuenta el apartado que se refiere a los modelos y al software. Indica las características de los modelos que son las responsables del éxito del uso de los modelos: abstracción, comprensibilidad, exactitud, predicción, bajo coste. También se habla de los beneficios del uso de los modelos. Otro punto importante es el que habla del rol de UML dentro de MDA, el lenguaje de modelado junto (con MOF) por excelencia de MDA. Por último se ha considerado relevante para el estudio del investigador el apartado que describe brevemente UML 2.0. Se han tomado las características no encontradas en otros estudios, la evolución de UML y las diferencias con respecto a UML 1.4.
Peso	8
Identificación del estudio	Robert B., Sudipto G., Trung D., *Model-Driven Development Using UML 2.0: Promises and Pitfalls*, Colorado State University, 2006.
Fecha de la extracción de datos	16 de mayo del 2006.
Resumen del estudio	El artículo habla de UML 2.0. En algunos aspectos presenta mejoras frente a versiones anteriores. A pesar de ello presenta problemas a los usuarios debido a su tamaño y su complejidad.
Información relevante del estudio para el investigador	Prácticamente todo el artículo es relevante para el investigador: UML 2.0, puntos de vista, conceptos de modelado…
Peso	9

Tabla C - 6: Fuentes primarias de IEEE para la búsqueda MDA AND UML

En las tablas anteriores se han omitido los estudios que aun siendo relevantes no se han añadido ya que la información ya se había extraído de otros estudios.

Identificación del estudio	Fuente
Atkinson, C., *Rechearting the UML Infrastructure,* Darmstadt University of Technology, 2002.	ACM
Bezivin, J., *From Object Composition to Model Transformation with the MDA,* University of Nantes, France, 2001.	IEEE
Hnetynka, P., Pise, M., *Hand-written vs. MOF-based Metadata Repositories: The SOFA Experience*, 2004.	IEEE
González-Pérez, C., *Tools for an Extended Object Modelling Environment,* Australia, 2007.	IEEE

Tabla C - 7: Estudios descartados por información repetida

En la revisión sistemática se han encontrado estudios que pueden resultar interesantes para futuros trabajos, los listamos en la tabla siguiente:

Identificación del estudio	Fuente	Tema futuro de interés
Favre, J., M., NGuyen, T., *Towards a Megamodel to Model Software Evolution Through Transformations,* Universtity of Grenoble, Francia, 2004.	Science@Direct	MDE
Estublier, J., Vega, G., *Reuse and Variability in Large Software Applications,* Grenoble, France, 2005.	ACM	Domain Specific Language (DSL) y MDE.

Tabla C - 8: Estudios útiles para futuros trabajos

Resolución y divergencias entre los revisores

En este caso existe un solo revisor por tanto no ha habido divergencias.

C.2. Evaluación de la planificación.

Antes de ejecutar la revisión sistemática, es necesario evaluar la planificación de la revisión. Una manera para llevar a cabo tal evaluación es pedir a expertos que nos revisen el protocolo. Otra forma para evaluar la planificación es comprobar la ejecución del protocolo. La revisión se ejecuta en un conjunto reducido de fuentes, si los resultados no son útiles o no encajan con nuestro estudio, se debe revisar el protocolo para crear una nueva versión.

Para planificar la revisión se ha tomado el apéndice primero del artículo (Kitchenham, 2004), exceptuando puntos finales dentro del apartado de resúmenes. El ejemplo disponible en el estudio ha servido de gran ayuda para guiar al investigador. También ha sido de gran utilidad la experiencia de compañeros en el curso de doctorado, que han sabido aconsejar y orientar al investigador.

C.3. Ejecución de la revisión.

Una vez evaluado la evaluación de la planificación, se puede iniciar la ejecución de la revisión sistemática. Durante esta fase, se tiene que ejecutar la búsqueda en las fuentes definidas y los estudios obtenidos se deben evaluar de acuerdo al criterio establecido. Para terminar, toda

la información relevante de la pregunta del estudio se debe extraer de los estudios seleccionados.

En este apartado se ha optado por hacer una revisión iterativa, es decir, ejecutar cada una de las fuentes según los criterios óptimos obtenidos de la búsqueda anterior.

Como se ha dicho anteriormente, se han seleccionado 4 fuentes: Science@Direct, ACM, Wiley y IEEE.

En primer lugar se han realizado búsquedas con Science@Direct ya que de las fuentes seleccionadas es la que menos documentos tiene registrados. Con esta fuente se han realizado todas las posibles combinaciones de cadenas de búsquedas, de todas ellas se han tomado aquellas cadenas de búsqueda que mejores resultados se han obtenido y se han aplicado a las fuentes restantes: ACM, Wiley y IEEE.

El criterio de selección se ha mantenido a lo largo de toda la revisión.

C.4. Conclusiones.

Evaluada la revisión sistemática, se llega a la conclusión de que la revisión ha sido satisfactoria para parte del trabajo del investigador en su informe técnico. El objetivo ha sido cubierto y la primera parte del informe técnico ha sido realizado con éxito.

Antes de realizar el informe técnico, el investigador tenía una cierta idea de su trabajo a desarrollar: definición de lenguajes de modelos dentro de MDA, en los que se tenía que hablar de los metamodelos UML 2.0 y MOF.

La revisión sistemática le ha servido de gran ayuda no sólo para responder a los puntos introductorios de definición de lenguajes de modelos dentro de MDA sino a extenderse en los apartados referentes a UML 2.0 y MOF. Además ha guiado notablemente al investigador a realizar un caso de uso para estos lenguajes.

En resumen, la revisión sistemática ha respondido a los siguientes apartados:

- Definiciones de los conceptos modelado y metamodelado.
- Definición de la arquitectura en niveles o capas de modelado utilizado en la OMG.
- Descripción de UML 2.0 y función dentro del modelado en MDA.
- Descripción de MOF y función dentro del modelado en MDA.
- Caso de estudio MOF-MDA, donde se han representado todas las capas de modelado mediante diagramas UML.

Con respecto a propuestas futuras, puesto que el informe técnico se trata de una comparación entre la definición de lenguajes de modelos dentro de MDA y usando DSL, el investigador propone una futura revisión para la definición de lenguajes de modelos usando DSLs, de esta forma el informe técnico estará completamente terminado y revisado y preparado para un futuro artículo o presentación de congreso.

En cuestión a la experiencia personal, el investigador se siente satisfecho con el trabajo realizado ya que la información proporcionada ha sido muy útil para el informe técnico que según los expertos, ha respondido los requisitos esperados.

En futuras revisiones sistemáticas se espera afianzar la metodología y sobre todo aprender de los errores y experiencias propias y ajenas.

D. Cuestionario del Caso de Estudio.

Valore del 1 (muy de acuerdo) al 5 (muy en desacuerdo) las siguientes preguntas referentes a la medición llevada a cabo con SMTool en el dominio de medición seleccionado.

Pregunta / Respuesta	Comentarios/Propuestas
¿Es adecuado el proceso de medición llevado a cabo para satisfacer las necesidades de medición realizada? 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ ¿Son suficientes las etapas definidas en el método de SMF para llevar a cabo las mediciones? 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐	
¿Se entiende el modelo de medición definido para el dominio de medición? 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ ¿Está de acuerdo que el modelo de medición gráfico facilita la definición del modelo de medición? 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ ¿Está de acuerdo que el modelo de medición gráfico permite ver toda la información de la medición que se va a ejecutar? 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐	
¿Se identifican claramente los resultados de la medición en el modelo? 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ ¿Está de acuerdo que el modelo de medición muestra los resultados de la medición de una manera entendible? 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐	

Pregunta / Respuesta	Comentarios/Propuestas
¿Es práctica y usable la herramienta SMTool para definir y ejecutar la medición? 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ ¿Está de acuerdo que la interfaz de SMTool resulta intuitiva para realizar mediciones? 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ ¿Está de acuerdo que la herramienta SMTool cubre las necesidades de medición de las herramientas existentes (Caliber, Together y Power Designer)? 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ ¿Está de acuerdo que SMTool facilita la definición y ejecución de mediciones de software)? 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐	
¿Es aplicable el modelo de medición en otros módulos de desarrollo? 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ ¿Se puede importar un modelo de medición para ejecutar una medición en el mismo dominio pero sobre un módulo de desarrollo distinto? 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ ¿Está de acuerdo que el marco de medición es genérico desde el punto de vista que se puede medir cualquier tipo de entidad (Requisito, Diagramas UML, etc.)? 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐	
En términos generales, ¿Considera que SMF es un marco de medición útil para llevar a cabo mediciones de software? 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ ¿SMF mejora la medición frente a la manera tradicional de medición llevada a cabo con herramientas específicas de cada dominio? 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐	

Apéndices.

- Lista de Acrónimos
- Referencias

1. Lista de Acrónimos.

CIM	*Computational Independent Model*
CMMI	*Capability Maturity Model Integration*
CQL	*Code Query Language*
CVS	*Concurrent Versions System*
CVS	*Comma-Separated Values*
CWM	*Common Warehouse Metamodel*
DAG	*Directed Aciclic Graph*
DDMM	*Domain Definition Metamodel*
DESERT	*Design Space Exploration tool suite*
DMM	*Dagstuhl Middle Metamodel*
DSL	*Domain Specific Language*
DSM	*Domain Specific Modeling*
DSML	*Domain Specific Modeling Language*
DSVL	*Domain Specific Visual Languages*
DTD	*Falta ponerlo*
DTD	*Document Type Definiton*
DVM	*Distributed Versioning Model*
EIS	*Entorno de Ingeniería del Software*
EMF	*Eclipse Modeling Framework*
FMESP	*Framework for the Modeling and Evaluation of Software Processes*
GEF	*Eclipse Graphical Editing Framework*
GME	*Generic Model Environment*
GMF	*Graphical Modeling Framework*
GPL	*General Purpose Language*
GQM	*Goal Question Metric*
GReAT	*Model Transformation tool suite*
GSL	*Graph Specification Language*
HMTL	*HyperText Markup Language*
ISIS	*Institute of Software Integrated Systems*
LMP	*Lenguaje de Modelado de Procesos*
LOC	*Lines Of Code*
LSF	*Language Specification File*
MDA	*Model Driven Development*
MDD	*Model Driven Development*
MDE	*Model Driven Engineering*

MDSD	*Model-Driven Software Development*
MDSL	*Measurement-Domain Specific Language*
MIC	*Model Integrated Computing*
MOF	*Meta Object Facility*
OCL	*Object Constrait Language*
ODEM	*Object-oriented DEsign Model*
OMG	*Object Management Group*
OO	*Object Oriented*
PIM	*Platform Independent Model*
PSM	*Practical Software Measurement*
PSM	*Platform Specific Models*
QVT	*Query/View/Transformation*
RML	*Relation Manipulation Language*
SaaS	*Software as a Service*
SDM	*Software Domain Models*
SEI	*Software Engineering Institute*
SMF	*Software Measurement Framework*
SMM	*Software Measurement Metamodel*
SMML	*Software Measurement Modelling Language*
SMO	*Software Measurement Ontology*
SMTool	*Software Measurement Tool*
SOC	*Service-Oriented Computing*
SPEM	*Software Process Engineering Metamodel Specification*
SPICE	*Software Process Improvement and Capability dEtermination*
SQL	*Structured Query Language*
SW-CMM	*Capability Maturity Model for Software*
UDM	*Model Management tool suit*
UoD	*Universe of Discourse*
WSA	*Web Services Architecture*
XMI	*XML Metadata Interchange*
XML	*Extensible Markup Language*

2. Referencias.

Abounader, J. y Lamb, D. (1997). *A data model for object-oriented design metrics*. Technical report, Queen's University, Kingston.

Ahmad, R. (1999). Visual Languages: A New Way of Programming. *Malaysian Journal of Computer Science* 12(1): 76-81, http://mjcs.fsktm.um.edu.my/document.aspx?FileName=67.pdf.

Alikacem, E. y Sahraoui, H. (2006). Generic metric extraction framework. *In Proceedings of IWSM/MetriKon*.

Arpaia, P., Buzio, M., Fiscarelli, L., Inglese, V., Commara, G. L. y Walckiers, L. (2009). Measurement-Domain Specific Language for Magnetic Test Specifications at CERN. *2009 IEEE Instrumentation and Measurement Technology Conference*, Singapore CERN/TE 2009-002: 1716 - 1720.

Atkinson, C. y Kühne, T. (2002). Model-Driven Development: A Metamodeling Foundation. *IEEE Software* 20(5): 36-41.

Auer, M., Graser, B. y Biffl, S. (2003). A Survey on the Fitness of Commercial Software Metric Tools for Service in Heterogeneous Environments: Common Pitfalls *Ninth IEEE International Software Metrics Symposium. IEEE METRICS*: 144-.

Avison, D., Lau, F., Myers, M. y Nielsen, A. (1999). Action Research. *Communications of the ACM* 42(1): 94-97.

Babbie, E. (1990). *Survey research methods*.

Balasubramanian, K., Gokhale, A., Karsai, G., Sztipanovits, J. y Neema, S. (2006). Developing Applications Using Model-driven Design Environments. *Computer* 39(2): 99-40.

Bansiya, J. (2000). Evaluating framework architecture structural stability. *ACM Comput. Survey* 32(1): 18.

Bansiya, J. y Davis, C. G. (2002). A hierarchical model for object-oriented design quality assessment. *IEEE Transactions on Software Engineering* 28: 4.

Baroni, A. L. (2005). Quantitative assessment of uml dynamic models. *Doctoral symposium at the 10th European Software Engineering Conference held jointly with 13th International Symposium on Foundations of Software Engineering (ESEC-FSE'05)*, ACM Press: 366-369.

Baroni, A. L., Braz, S. y Abreu, F. (2002). Using OCL to formalize object-oriented design metrics definitions. *ECOOP'02 Workshop on Quantitative Approaches in OO Software Engineering*.

Basili, V. (2000). Using Experiments to Build a Body of Knowledge. *7th European Workshop on Software Process Technology (EWSPT 2000)*, Salzburg (Austria): 265-282.

Basili, V. R., Shull, F. y Lanubile, F. (1999). Building Knowledge through Families of Experiments. *IEEE Transactions on Software Engineering* 25(4): 456-473.

Baskerville, R. (1999). Investigating Information Systems with Action Research. *Communications of the Association for Information Systems* 2(19), http://cis.gsu.edu/~rbaskerv/CAIS_2_19/index.html.

Berenbach, B. y Borotto, G. (2006). Metrics for model driven requirements development. *28th International Conference on Software Engineering (ICSE'06)*, ACM Press: 445-451.

Bernárdez, B., Durán, A. y Genero, M. (2004). Empirical Evaluation and Review of a Metrics-Based Approach for Use Case Verification. *Journal of Research and Practice in Information Technology* 36(4).

Beyer, D., Noack, A. y Lewerentz, C. (2005). Eficient relational calculation for software analysis. *IEEE Trans. Softw. Eng* 31(2): 137-149.

Bézivin, J. (2004). In search of a Basic Principle for Model-Driven Engineering. *Novatica Journal, Special Issue* 5: 21-24.

Bézivin, J., Jouault, F. y Touzet, D. (2005). Principles, standards and tools for model engineering. *10th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'2005)*: 28-29.

Biolchini, J., Mian, P. G., Natali, A. C. y Travassos, G. H. (2005). Systematic Review in Software Engineering.

Brereton, P., Kitchenham, B., Budgen, D. y Li, Z. (2008). Using a Protocol Template for Case Study Planning. *EASE: EBSE*.

Briand, L. C., Morasca, S. y Basili, V. R. (2002). An Operational Process for Goal-Driven Definition of Measures. *IEEE Trans. Softw. Eng* 28(12): 1106-1125, http://www.computer.org/tse/ts2002/e1106abs.htm.

Brown, M. y Dennis, G. (2004). Measurement and Analysis: What Can and Does Go Wrong? *10th IEEE International Symposium on Software Metrics (METRICS'04)*: 131-138.

Cabot, J., Mazón, J. N., Pardillo, J. y Trujillo, J. (2010). Specifying Aggregation Functions in Multidimensional Models with OCL ER 2010: 419-432.

Cabot, J. y Teniente, E. (2006). A metric for measuring the complexity of ocl expressions. *Model Size Metrics Workshop co-located with MODELS'06*.

Calero, C., Piattini, M. y Genero, M. (2001). Empirical validation of referential integrity metrics. *Information & Software Technology. Special Issue on Controlled Experiments in Software Technology* 43(15): 949-957.

Consens, M. P. y Mendelzon, A. O. (1990). GraphLog: a visual formalism for real life recursion. *ACM Symp. on Principles of Database Systems,*: 404-416.

Cook, S. (2004). Domain-Specific Modeling and Model Driven Architecture. *The MDA Journal: Model Driven Architecture Straight from the Masters*, David S. Frankel and John Parodi, Editors. 5: (1-10), http://www.bptrends.com/publicationfiles/01-04%20COL%20Dom%20Spec%20Modeling%20Frankel-Cook.pdf.

Cook, T. D. y Campbell, D. T. (1979). *Quasi-experimentation: design and analysis issues for field settings*, Boston, Houghton Mifflin Co.

Cruz-Lemus, J. A., Genero, M., Manso, M. E., Morasca, S. y Piattini, M. (2009). Assessing the Understandability of UML Statechart Diagrams with Composite States - A Family of Empirical Studies. *Empirical Software Engineering* 14(6): 685-719.

Cruz-Lemus, J. A., Maes, A., Genero, M., Poels, G. y Piattini, M. (2010). The impact of structural complexity on the understandability of UML statechart diagrams. *Information Sciences* 180(11): 2209-2220.

Czarnecki, K. y Eisenecker, U. (2000). *Generative Programming:Methods, Tools, and Applications*, Addison-Wesley.

Champeaux, D. (1997). *Object-oriented Development Process and Metrics*, Prentice- Hall.

Christensen, N. H. (2003). *Domain-Specific Languages in Software Development and the relation to partial evaluation*. Department. of Computer Science. Denmark, University of Copenhagen.

Deursen, A. v., Klint, P. y Visser, J. (2000). Domain-Specific Languages: An Annotated Bibliography. *SIGPLAN Notices* 35(6): 26-36.

Dumke, R. R. y Grigoleit, H. (1997). Efficiency of CAMEtools in software quality assurance. *Software Quality Journal* 6(2): 157-169.

Eclipse. (2007). *Eclipse Graphical Editing Framework (GEF) Main Page*. http://www.eclipse.org/gef/.

Eclipse. (2010a). *Eclipse Graphical Modeling Framework (GMF) Main Page*. http://www.eclipse.org/gmf/.

Eclipse. (2010b, septiembre 2010). *Eclipse Modelling Framework (EMF) Main Page*. http://www.eclipse.org/emf.

Eichberg, M., Germanus, D., Mezini, M., Mrokon, L. y Schäfer, T. (2006). Qscope: an open, extensible framework for measuring software projects. *10th European Conference on Software Maintenance and Reengineering (CSMR'06)*.

Eisenhardt, K. M. (1989). Building Theories from Case Study Research. *Academy of Management Review* 14(4): 532-550.

ElWakil, M., El-Bastawissi, A., Riad, M. B. y Fahmy, A. (2005). A novel approach to formalize and collect object-oriented design-metrics. *9th International Conference on Empirical Assessment in Software Engineering*.

Eysenck, M. W. y Keane, M. T. (2005). *Cognitive Psychology: A Student's Handbook*, Lawrence Erlbaum Associates: Mahwah NJ.

Feilkas, M. (2006). How to represent Models, Languages and Transformations? *Proceedings of the 6th OOPSLA Workshop on Domain-Specific Modeling (DSM'06)*: 204-213, www.dsmforum.org/events/DSM06/Papers/17-Feikas.pdf

Fenton, N. y Pfleeger, S. L. (1997). *Software Metrics: A Rigorous & Practical Approach, Second Edition*, PWS Publishing Company.

Ferreira, M., García, F., Ruiz, F., Bertoa, M. F., Calero, C., Vallecillo, A., Piattini, M. y Mora, B. (2006). *Medición del Software: Ontología y Metamodelo*, Department of Computer Science. University of Castilla - La Mancha. Informe Técnico UCLM-TSI-001, http://www.esc.uclm.es:8080/tsi/informes.

Florac, W. A., Carleton, A. D. y Barnard, J. (2000). Statistical Process Control: Analyzing a Space Shuttle Onboard Software Process. *IEEE Software* 17 (4)(4).

Fowler, M. (2006). *DSL Boundary*. http://martinfowler.com/bliki/DslBoundary.html.

French, W. y Bell, C. H. (1996). *Organizational Development: Behavioral Science Interventions for Organization Improvement*, Prentice Hall: London.

Furtado, A. W. B. (2006). *Sharpludus: improving game development experience through software factories and domain-specific languages*, Universidade Federal de Pernambuco (UFPE) Mestrado em Ciência da Computação centro de Informática (CIN).

García, F. (2004). *FMESP: Marco de Trabajo Integrado para el Modelado y la Medición de los Procesos Software*. PhD thesis, Computer Science Department. Ciudad Real, Spain, University of Castilla-La Mancha.

García, F., Bertoa, M. F., Calero, C., Vallecillo, A., Ruíz, F., Piattini, M. y Genero, M. (2006a). Towards a consistent terminology for software measurement. *Information & Software Technology* 48(8): 631-644

García, F., Piattini, M., Ruiz, F., Canfora, G. y Visaggio, C. A. (2006b). FMESP: Framework for the modeling and evaluation of software processes. *Journal of Systems Architecture - Agile Methodologies for Software Production* 52(11): 627-639.

García, F., Ruiz, F., Calero, C., Bertoa, M. F., Vallecillo, A., Mora, B. y Piattini, M. (2009). On the Effective Use of Ontologies in Software Measurement. *The Knowledge Engineering Review* 24(1): 23-40.

García, F., Serrano, M., Cruz-Lemus, J., Ruiz, F. y Piattini, M. (2007). Managing Software Process Measurement: A Metamodel-Based Approach. *Information Sciences* 177: 2570–2586.

Genero, M., Miranda, D. y Piattini, M. (2002). Defining and validating metrics for uml statechart diagrams. *Proceedings of QAOOSE'2002*.

Genero, M., Moody, D. y Piattini, M. (2005a). Assessing the capability of internal metrics as early indicators of maintenance effort through experimentation. *Journal of Software Maintenance* 17(3): 225-246.

Genero, M., Piattini, M. y Calero, C. (2005b). A Survey of Metrics for UML Class Diagrams. *Journal of Object Technology* 4(9): 59-92, http://www.jot.fm/issues/issue_2005_11/article1.

Gîrba, T., Lanza, M. y Ducasse, S. (2005). Characterizing the evolution of class hierarchies. *9th European Conference on Software Maintenance and Reengineering (CSMR'05)*, IEEE Computer Society: 2-11.

Greenfield, J. (2004). Software Factories: Assembling Applications with Patterns, Models, Frameworks and Tools. *Third International Conference, GPCE 2004*, Vancouver, Canada: 488.

Guerra, E., Lara, J. d. y Díaz, P. (2008). Visual specification of measurements and redesigns for domain specific visual languages. *Journal of Visual Languages and Computing* 19 (3): 399-425.

Harel, D. y Rumpe, B. (2004). Meaningful Modeling: What's the Semantics of "Semantics"? *IEEE Computer* 37(10): 64-72.

Harmer, T. J. y Wilki, F. G. (2002). An extensible metrics extraction environment for object-oriented programming languages. *International Conference on Software Maintenance*.

Harrison, W. (2004). A flexible method for maintaining software metrics data: a universal metrics repository. *Journal of Systems and Software* 72(2): 225-234

Ikv. (2010, septiembre 2010). *Medini QVT Home Page*. http://www.ikv.de.

ISIS. (2010). *Model Integrated Computing (MIC) main page*. http://www.isis.vanderbilt.edu/research/MIC.

ISO/IEC (2001). *ISO/IEC 9126-1: Software Engineering-Software product quality-Part 1 : Quality model. Geneva, Switzerland, International Organization for Standardization.*

ISO/IEC (2002). *ISO 15939: Software Engineering - Software Measurement Process.*

ISO/IEC (2004). Software and Systems Engineering - Guidelines for the application of ISO/IEC 9001:2000 to Computer Software. *International Standards Organization*.

Jedlitschka, A. y Ciolkowski, M. (2005). Reporting Guidelines for Controlled Experiments in Software Engineering. *ACM/IEEE International Symposium on Empirical Software Engineering* 95-195.

Jokikyyny, T. y Lassenius, C. (1999). Using the internet to comunicate software metrics in a large organization. *GlobeCom'99*.

Juristo, N. y Moreno, A. M. (2001). *Basics of Software Engineering Experimentation*, Kluwer Academic Publishers.

Kempkens, R., Rösch, P., Scott, L. y Zettel, J. (2000). Instrumenting Measurement Programs with Tools. *PROFES 2000*, Oulu, Finland.

Kitchenham, B. (2004). *Procedures for performing systematic reviews*. Software Engineerin Group, Department of Computer Science, Keele University and Empirical Software Engineering National ICT Australia Ltd.

Kitchenham, B. A., Pfleeger, S. L., Pickard, L. M., Jones, P. W., Hoaglin, D. C., Emam, K. E. y Rosenberg, J. (2002). Preliminary guidelines for empirical research in software engineering. *IEEE Transactions on Software Engineering* 28(8): 721 - 734

Kock, N. y Lau, F. (2001). Information Systems Action Research: Serving Two Demanding Masters. *Information Technology & People (special issue on Action Research in Information Systems)* 14(1): 6-11.

Kolovos, D. S., Paige, R. F., Kelly, T. y Polack, F. A. C. (2006). Requirements for Domain-Specific Languages. *First ECOOP Workshop on Domain-Specific Program Development (ECOOP'06)*, Nantes, France.

Komi-Sirviö, S., Parviainen, P. y Ronkainen, J. (2001). Measurement Automation: Methodological Background and Practical Solutions-A Multiple Case Study. *Seventh International Software Metrics Symposium (METRICS'01)*, London.

Kramler, G., Kappel, G., Reiter, T., Kapsammer, E., Retschitzegger, W. y Schwinger, W. (2006). Towards a Semantic Infrastructure Supporting Model-based Tool Integration. *international workshop on Global integrated model manage. GaMMa'06*: 43 - 46.

Kurtev, I., Bézivin, J., Jouault, F. y Valduriez, P. (2006). Model-based DSL Frameworks. *OOPSLA Companion 2006*: 602-616.

Lanza, M. y Ducasse, S. (2002). Beyond language independent object-oriented metrics: Model independent metrics. *Proceedings of the 6th International Workshop on Quantitative Approaches in Object-Oriented Software Engineering*: 77-84.

Lavazza, L. y Agostini, A. (2005). Automated Measurement of UML Models: an open toolset approach. *Object Technology* 4(4): 115-134.

Lott, C. M. y Rombach, H. D. (1996). Repeatable Software Engineering Experiments for Comparing Defect-Detection techniques. *Journal of Empirical Software Engineering* 1(3): 241-277.

Ma, H., Ji, Z., Shao, W. y Zhang, L. (2005). Towards the uml evaluation using taxonomic patterns on meta-classes. *Fifth International Conference on Quality Software (QSIC'05)* 0: 37.

Ma, H., Shao, W., Zhang, L., Ma, Z. y Jiang, Y. (2004). Applying OO metrics to assess UML meta-model. *MODELS/UML'2004, UML2004*.

MacDonell, S. G., Shepperd, M. J. y Sallis, P. J. (1997). Metrics for Database Systems: An Empirical Study. *4th IEEE International Software Metrics Symposium (METRICS 1997)*, Albuquerque, NM, USA, IEEE Computer Society: 99-107, http://citeseer.ist.psu.edu/macdonell97metrics.html.

Mahmood, S. y Lai, R. (2005). Measuring the complexity of a uml component specication. *Fifth International Conference on Quality Software (QSIC '05)*, IEEE Computer Society: 150-160.

Marcelo, C. y Parrilla, A. (1991). El estudio de caso: una estrategia para la formación del profesorado y la investigación didáctica. *El Estudio de caso en la formación del profesorado y la investigación didáctica*, Publicaciones de la Universidad de Sevilla: (9-71).

Marinescu, C., Marinescu, R. y Gîrba, T. (2005). Towards a simplified implementation of object-oriented design metrics. *In IEEE METRICS*: 11.

Mattsson, M. y Bosch, J. (1999). Characterizing stability in evolving frameworks. *Technology of Object-Oriented Languages and Systems (TOOLS '99)*, IEEE Computer Society: 118.

McGarry, J., Card, D., Jones, C., Layman, B., Clark, E., Dean, J. y Hall, F. (2002). *Practical Software Measurement. Objective Information for Decision Makers*, Addison-Wesley.

McQuillan, J. A. y Power, J. F. (2006). Experiences of using the dagstuhl middle metamodel for dening software metrics. *4th International Conference on Principles and Practices of Programming in Java*.

McTaggart, R. (1991). Principles for Participatory Action Research *Adult Education Quarterly* 41(3): 168-187.

Mens, T. y Lanza, M. (2002). A graph-based metamodel for object-oriented software metrics. *Electronic Notes in Theoretical Computer Science* 72: 57-68.

Mernik, M., Heering, J. y Sloane, A. M. (2005). When and how to develop domain-specific languages. *ACM Computing Surveys (CSUR)* Volume 37(4): 316-344.

Misic, V. B. y Moser, S. (1997). From formal metamodels to metrics: An object-oriented approach. *Technology of Object-Oriented Languages and Systems Conference (TOOLS'97)*

330.

Modelware (2006a). *D2.2 MDD Engineering Metrics Definition*. Framework Programme Information Society Technologies.

Modelware (2006b). *D2.5 MDD Engineering Metrics Baseline*. Framework Programme Information Society Technologies.

Modelware (2006c). *D2.7 MDD Business Metrics*. Framework Programme Information Society Technologies.

Monperrus, M., Jéezéquel, J.-M., Champeau, J., Brigitte y Hoeltzener (2008). A Model-Driven Measurement Approach. *Proceedings of the ACM/IEEE 11th International Conference on Model Driven Engineering Languages and Systems (MODELS'2008)*, Nantes (France) 5301/2008: 505-519.

Monperrus, M., Jézéquel, J.-M., Baudry, B., Champeau, J. y Hoeltzener, B. (2010). Model-driven Generative Development of Measurement Software. *Software and Systems Modeling (SoSyM), Springer* 9.

Moody, D. (2000). Building links between IS research and professional practice: improving the relevance

and impact of IS research. *Proceedings of the 21st International Conference on Information Systems*, Brisbane (Australia): 351-360.

Mora, B., García, F., Ruiz, F. y Piattini, M. (2008a). SMML: Software Measurement Modeling Language. *The 8th OOPSLA Workshop on Domain-Specific Modeling*, Nashville (Tennessee) USA, University of Alabama at Birmingham. Department of Computer and information Sciences: 52-59, http://www.dsmforum.org/events/DSM08/Papers/9-Mora.pdf.

Mora, B., García, F., Ruiz, F. y Piattini, M. (2008b). Supporting Software Process Measurement by Using Metamodels: A DSL and a Framework. *Proceedings of the Third International Conference on Software and Data Technologies ICSOFT 2008. Workshop on Metamodelling - Utilization in Software Engineering, MUSE 2008.*, Oporto (Portugal) SE/GSDCA/MUSE: 305-312.

Mora, B., García, F., Ruiz, F. y Piattini, M. (2009a). Model-Driven Software Measurement Framework: a case study. *The 9th International Conference on Quality Software, QSIC 2009*, Jeju, Korea., IEEE Reliability Society; Korea Advanced Institute of Science and Technology (KAIST): 239-248.

Mora, B., García, F., Ruiz, F. y Piattini, M. (2009b). SMML: Lenguaje para la Representación de Modelos de Medición del Software. *CIbSE*: 181-194.

Mora, B., García, F., Ruiz, F. y Piattini, M. (2010a). Graphical versus textual software measurement modelling: an empirical study. *Software Quality Journal* Published online, http://www.springerlink.com/content/h076841x37061004/fulltext.pdf.

Mora, B., García, F., Ruiz, F. y Piattini, M. (2010b). Lenguaje para la representación de Modelos de Medición del Software. *Calidad del Producto y proceso software*, Ra-Ma. 4: (91-115), http://www.dsmforum.org/events/DSM08/Papers/9-Mora.pdf.

Mora, B., García, F., Ruiz, F., Piattini, M., Boronat, A., Gómez, A., Carsí, J. Á. y Ramos, I. (2007a). Marco de Trabajo basado en MDA para la Medición Genérica del Software. *Actas de las XII Jornadas de Ingeniería del Software y Bases de Datos, JISBD'2007.*, Zaragoza: 211-220.

Mora, B., García, F., Ruiz, F., Piattini, M., Boronat, A., Gómez, A., Carsí, J. Á. y Ramos, I. (2008c). Marco de Trabajo basado en MDA para la Medición Genérica del Software. *IEEE América Latina* 6(4): 363-370, http://ewh.ieee.org/reg/9/etrans/ieee/issues/vol6/vol6issue4Aug.2008/09Mora.htm.

Mora, B., García, F., Ruiz, F., Piattini, M., Boronat, A., Gómez, A., Carsí, J. Á. y Ramos, I. (2008d). Software Measurement by using QVT Transformation in an MDA context. *10th International Conference on Enterprise Information Systems - ICEIS 2008*, Barcelona (Spain) 1: 117-124.

Mora, B., Ruiz, F., Garcia, F. y Piattini, M. (2006). *Definición de Lenguajes de Modelos MDA Vs DSL*, Department of Computer Science. University of Castilla - La Mancha, http://www.uclm.es/dep/tsi/pdf/.

Mora, B., Ruiz, F., Garcia, F. y Piattini, M. (2007b). Medición del Software mediante Transformaciones QVT. *DSDM 2007: Desarrollo de Software Dirigido por Modelos, MDA y Aplicaciones (JISBD'07)*, Zaragoza 1 (6): 119-135.

Mora, B., Ruiz, F., Garcia, F. y Piattini, M. (2008e). *SMML: Software Measurement Modeling Language*, Department of Computer Science. University of Castilla - La Mancha, http://www.uclm.es/dep/tsi/pdf/SMML_Software_Measurement_Modeling_Language.pdf.

Mora, B., Ruiz, F., García, F. y Piattini, M. (2007c). Experiencia en transformación de modelos de procesos de negocios desde BPMN a XPDL. *Actas de IDEAS 2007: X Workshop iberoamericano de Ingeniería de Requisitos y Ambientes de Software*: 165-178.

Morasca, S. (2001). *Software Measurement. In Handbook of Software Engineering and Knowledge Engineering*.

Myers, M. (1997). Qualitative Research in Information Systems. *MISQ Discovery* 21(2): 241-242.

OMG (2001). *UML Specification Version 1.4.*
, Object Management Group, http://www.omg.org/docs/formal/01-09-67.pdf.

OMG (2002a). *MOF 2.0 Query/Views/Transformations Request for Proposal*, Object Management Group.

OMG (2002b). *Object Constraint Language (OCL) Specification*, Object Management Group.

OMG (2002c). *XML Metadata Interchange (XMI) Specification*, Object Management Group, http://www.omg.org/formal/00-11-02.pdf.

OMG (2003a). *Common Warehouse Metamodel (CWM) Specification v 1.1*, http://www.omg.org/docs/formal/03-03-02.pdf.

OMG (2003b). *Meta Object Facility (MOF). Core Specification Version 2.0*, Object Management Group, http://www.omg.org/docs/ptc/03-10-04.pdf.
OMG (2003c). *Model-Driven Architecture (MDA) Guide Version 1.0.1*, Object Management Group, http://www.omg.org/mda/aspecs.htm.
OMG (2003d). *OCL 2.0 - OMG Final Adopted Specification*, Object Management Group.
OMG (2005a). *Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification*, Object Management Group.
OMG (2005b). *MOF 2: XMI - Mapping Specification, version 2.1*, Object Management Group, http://www.omg.org/docs/formal/05-09-01.pdf.
OMG (2007). *Architecture-Driven Modernization (ADM): Software Metrics Meta-Model (SMM). OMG Document: dmtf/2007-08-01*, Object Management Group.
Özgür, T. (2007). *Comparison of Microsoft DSL Tools and Eclipse Modeling Frameworks for Domain-Specific Modeling in the context of the Model Driven Development*. School of Engineering. Ronneby, Sweden, Blekinge Institute of Technology: 56.
Padak, N. y Padak, G. (1994). *Guidelines for Planning Action Research Projects. Ohio Literacy Resource Center*, http://archon.educ.kent.edu/Oasis/Pubs/0200-08.html.
Palza, E., Fuhrman, C. y Abran, A. (2003). Establishing a Generic and Multidimensional Measurement Repository in CMMI context *28th Annual NASA Goddard Software Engineering Workshop (SEW'03)*, Greenbelt (Maryland, USA): 12-22.
Patig, S. (2008). A practical guide to testing the understandability of notations. *Fifth Asia-Pacific Conference on Conceptual Modelling (APCCM 2008)*, Australia, January 2008: 49-55.
Pelechano, V., Albert, M., Javier, M. y Carlos, C. (2006). Building Tools for Model Driven Development comparing Microsoft DSL Tools and Eclipse Modeling Plug-ins. *Desarrollo de Software Dirigido por Modelos - DSDM'06*, Sitges (Barcelona) España.
Pfleeger, S. (1994). Experimental Design and Analysis in Software Engineering. *Software Engineering Notes* 1: 219-253.
Pfleeger, S. (1995). Experimental Design and Analysis in Software Engineering. *Software Engineering Notes* 1: 219-253.
Philips, P. A. (1998). Disseminating and Applying the Best Evidence. *Medical Journal of Australia* 168: 260-261.
Pressman, R. (2001). *Ingeniería del Software. Un enfoque práctico (5th ed.)*.
Reissing, R. (2001). Towards a model for object-oriented design measurement. *ECOOP'01 Workshop QAOOSE*.
Reynoso, L., Genero, M., Cruz-Lemuz, J. y Piattini, M. (2006). OCL 2: Using OCL in the Formal Definition of OCL Expression Measures. *1st Workshop on Quality in Modeling (QIM'2006)*.
Reynoso, L., Genero, M. y Piattini, M. (2003). Measuring ocl expressions: a "tracing"-based approach. *QAOOSE'2003*.
Robson, C. (1993). *Real world research: A resource for social scientists and practitioners-researchers.*
.
Saeki, M. y Kaiya, H. (2006). Model metrics and metrics of model transformation. *1st Workshop on Quality in Modeling*: 31-45.
Scotto, M., Sillitti, A., Succi, G. y Vernazza, T. (2004). A relational approach to software metrics. *Proceedings of the 2004 ACM symposium on Applied computing (SAC'2004)*, Nicosia, Cyprus: 1536-1540.
Schmidt, D. C. (2006). Model-Driven Engineering. *IEEE Computer Society*: 25-31.
SDMetrics. (2006). *The software design metrics tool for the UML*. http://www.sdmetrics.com.
Selic, B. (2006). Model-Driven Development: Its Essence and Opportunities. *Ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'06)*: 313-319.
Shull, F., Carver, J. C., Vegas, S. y Juzgado, N. J. (2008). The role of replications in Empirical Software Engineering. *Empirical Software Engineering* 13(2): 211-218.

Smacchia, S. A. (2006). *Code query language 1.5 specification*. Technical report, Smacchia S.A.

Stake, R. (1995). *The art of case study research*.

Stake, R. (2005). Qualitative case studies. *The Sage handbook of qualitative research*, In N. K. Denzin & Y. S. Lincoln (Eds.): (pp. 433-466).

Tang, M.-H. y Chen, M.-H. (2002). Measuring OO design metrics from UML. *MODELS/UML'2002* 2460/2002: 361-370.

Tichelaar, S. y Demeyer, S. (1998). An exchange model for reengineering tools. *Workshop on Object-Oriented Technology at ECOOP'98*: 82-84.

Tsalidis, C., Christodoulakis, D. y Maritsas, D. (1992). Athena: a software measurement and metrics environment. *Journal of Software Maintenance* 4(2): 61-81.

Vépa, É., Bézivin, J., Brunelière, H. y Jouault, F. (2006). Measuring Model Repositories. *Model Size Metrics Workshop at the MoDELS/UML 2006 conference*, Genoava, Italy, http://www.sciences.univ-nantes.fr/lina/atl/www/papers/MeasuringModelRepositories.pdf.

Völter, M. (2009). MD* Best Practices. *Journal of Object Technology* 8(6): 79-102.

Völter, M., Stahl, T., Bettin, J., Haase, A. y Helsen, S. (2006). *Model-Driven Software Development: Tecnology, Engineering, Management*, Wiley.

Wadsworth, Y. (1998). *What is Participatory Action Research?* Action Research International, paper 2, http://www.scu.edu.au/schools/gcm/ar/ari/p-ywadsworth98.html.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B. y Wesslén, A. (2000). *Experimentation in Software Engineering: An Introduction*, International Series in Software Engineering.

Wood-Harper, T. (1985). Research Methods in Information Systems: Using Action Research. *In Research Methods in Information Systems* Amsterdam. North-Holland: 169-191.

Yin, R. K. (2002). *Case Study Research: Design and Methods*, Beverly hills.

Zelkowitz, M. y Wallace, D. (1998). Experimental models for validating technology. *IEEE Computer* 31(5): 23-31.