UNIVERSIDAD DE LAS CIENCIAS INFORMATICAS

FACULTAD 4

Título: Procedimiento para la definición de modelos de dominio específico y lenguaje de dominio específico para sistemas de gestión del aprendizaje

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autores

Daylín Soto Zamora

Lilivett Ovalle Borrego

Tutores

Msc. Dainys Gainza Reyes

Ing. José Antonio Soto Pérez

La Habana, junio 2012

"Año 54 de la Revolución"

Declaración de autoría

Declaramos que somos los autores de este trabajo y autorizamos a la Facultad 4 de la Universidad de las Ciencias Informáticas, así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmamos la presente a los _____ días del mes de _____ del año 2012.

_____ _____

Autora **Autora**

_____ _____

Tutor **Tutora**

_____

Cotutor

Agradecimientos de "Lily"

A mi familia por haber confiado en mí, en especial a mi mamá por su amor y amistad incondicional y sobre todo porque sin su apoyo no hubiera sido posible alcanzar este sueño.

A mi hermana por su fidelidad y dulzura.

A mi papá por inspirarme espíritu luchador y trabajador.

A mis tutores por la dedicación que han brindado en la realización de esta tesis.

Agradecimientos de "Day"

A mis padres que siempre me apoyaron en los buenos y malos momentos, por darme ánimo cuando más lo necesité y por ser los mejores padres que cualquier niña puede desear.

A mi hermanito que ha tenido que soportar mis malcriadeces por más de 5 años.

A mi compañera de tesis que tanta paciencia tuvo conmigo.

A mi novio que siempre estuvo ahí cuando lo necesité.

A mis tutores por apoyarnos y confiar en nosotros.

A mis mejores amigas Aleyda (mi negra), Kity y Nune que siempre tendrán un lugar especial en mi corazón.

A mis compañeros de grupo y todos aquellos que de una forma u otra compartieron conmigo estos 5 años.

Dedicatoria

“A mis padres, a mi querida hermana y a mi abuelo que está siempre presente en cada paso que doy en la vida, gracias por todo…”

Lily

“A mis padres y a mi hermanito, gracias por todo yo sé que este es también su sueño…”

Day

Resumen

En este trabajo se realiza una propuesta para la correcta definición de Modelos de Dominio Específico y Lenguajes de Dominio Específico en los Sistemas de Gestión del Aprendizaje. Utilizando como problemática la necesidad de contar con una documentación técnica que guíe a los nuevos investigadores a través del proceso de definición de los dominios de trabajo es que tiene lugar la presente investigación.

Siguiendo la línea de la investigación se pasó a la elaboración de la propuesta. En la misma se realizó un estudio detallado de los principios básicos del Desarrollo de Software Dirigido por Modelos, unido a ello, las diferentes vertientes que abarcan esta técnica, haciendo énfasis en el Modelo Específico del Dominio, además de los principales conceptos de los Sistemas de Gestión de Aprendizaje.

Una vez realizado este estudio se llevó a cabo el procedimiento, en el cual fueron definidas 2 etapas, además de un conjunto de pasos complementarios.

Finalmente se retroalimentó y validó teóricamente la investigación mediante el uso del método Delphi, el cual a través del criterio de un grupo de expertos se pudo determinar el nivel de aceptación que tuvo el proceso de definición de Modelo de Dominio Específico y Lenguaje de Dominio Específico.

Palabras clave: investigación, procedimiento, lenguaje, dominio, modelo

Índice

Introducción

Desde hace algunas décadas la informática forma parte de la cotidianidad de las personas, porque con todos y cada uno de los programas que ofrece permite una comunicación mucho más avanzada que en los tiempos de nuestros antepasados. La necesidad de un avance continuo de la misma ha traído consigo el desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC), fenómeno que ha revolucionado al mundo en muchos aspectos teniendo especial impacto en la sociedad moderna. El desarrollo de estas tecnologías ha determinado en gran medida el avance de la humanidad, ya sean a niveles de investigación, ciencia, educación, comunicación, entre otros. En las entidades educativas ha jugado un papel fundamental para optimizar el proceso de enseñanza – aprendizaje, siendo necesario el desarrollo del software especializado. Es así como los Sistemas de Gestión del Aprendizaje (LMS, del inglés Learning Management System) se convierten en uno de los pilares del sistema educativo a distancia más importante, perfilándose como la herramienta base de las próximas generaciones de educandos.

La Universidad de las Ciencias Informáticas (UCI) nacida como un proyecto de la Revolución Cubana tiene como objetivos: formar profesionales altamente calificados en la rama de la Informática, producir aplicaciones y servicios informáticos, a partir de la vinculación estudio-trabajo como modelo de formación y servir de soporte a la industria cubana de la informática (1). Para dar cumplimiento a este último la UCI cuenta con una infraestructura productiva integrada por varios centros en los que se encuentran: FORTES (Centro de Tecnologías para la Formación), CESIM (Centro de Informática Médica), ISEC (Centro Informatización de la Seguridad Ciudadana), GEYSED (Centro de Desarrollo Geoinformática y Señales Digitales), entre otros. El centro FORTES se especializa en el desarrollo de tecnologías que permitan ofrecer servicios y productos para la implementación de soluciones de formación aplicando las TIC, a todo tipo de instituciones con diferentes modelos de formación y condiciones tecnológicas (2). Gran parte de los proyectos productivos de este centro están enfocados de cierta forma a los Sistemas de Gestión del Aprendizaje.

Entre los problemas principales a los que se enfrentan estos proyectos se encuentran: el atraso en los cronogramas para la entrega final del producto, el divorcio entre el trabajo de los analistas y de los programadores, la complejidad a la hora de gestionar los requisitos, entre otros (ver Anexo 1); todo esto se debe al desarrollo artesanal de los proyectos ya que no se aplican métodos industriales o procesos definidos de producción.

El uso de los modelos se ha hecho necesario en el desarrollo de los proyectos educativos, siendo empleado desde hace décadas como forma de documentación o de razonar sobre el código. Estos juegan un papel importante ya que especifican y validan el sistema (estructuras y comportamientos), permitiendo la detección de errores u omisiones en el diseño y constituyen una guía para el proceso de implementación. Con el propósito de suponer un avance significativo hacia la industrialización del software, o al menos proporcionar mejoras significativas en la productividad y calidad, emerge el paradigma del Desarrollo de Software Dirigido por Modelos (DSDM), término que engloba los diferentes paradigmas de desarrollo de software que tienen en común el uso de modelos para representar varios aspectos de un sistema software con el propósito final de elevar el nivel de abstracción y de automatización.

El DSDM está conformado por cuatro vertientes: Arquitectura Dirigida por Modelos (MDA, del inglés Model Driven Architecture), el enfoque de Factorías de Software, la Modernización Dirigida por Modelos y el Modelo Específico del Dominio (DSM, del inglés Domain Specific Modeling) aportando este último grandes beneficios al aumento de la productividad disminuyendo la incidencia de errores humanos y facilitando la captura de requerimientos.

El DSM plantea aumentar el nivel de abstracción centrándose en los conceptos y dejando a un segundo plano los aspectos o características de la implementación, siendo el código final generado automáticamente. Los expertos en un dominio crean los Lenguajes Específicos del Dominio (DSL, del inglés Domain Specific Language) y sus generadores de código, y los desarrolladores los usan para especificar una solución de alto nivel, de una forma más productiva que escribiendo el código en un lenguaje de programación.

En la Universidad de las Ciencias Informáticas solo un pequeño grupo de investigadores ha utilizado el paradigma DSM en el desarrollo de su investigación donde han tenido que enfrentarse a diversos problemas como: la falta de conocimientos, de orientación, la poca experiencia y la escasa documentación por lo novedoso del tema, siendo este último el más significativo ya que no cuentan con la documentación técnica necesaria que los guíe a través del proceso de definición de los dominios de trabajo y los elementos a tener en cuenta para una correcta definición de los mismos; así como para los lenguajes de dominio específicos.

Por lo antes expuesto el **problema científico de la investigación queda resumido en la siguiente interrogante**: ¿Cómo apoyar el proceso de definición de Modelos de Dominio Específico y Lenguaje de Dominio Específico para los Sistemas de Gestión del Aprendizaje?

El **objetivo general** se centra en elaborar un procedimiento para la definición de Modelos de Dominio Específico y Lenguaje de Dominio Específico para los Sistemas de Gestión del Aprendizaje.

El **objeto de estudio** de la investigación se centra en el Desarrollo de Software Dirigido por Modelos.

El **campo de acción** en el que está enmarcado el trabajo, es en las técnicas para la definición de Modelos de Dominio Específico y Lenguajes de Dominio Específico en los Sistemas de Gestión del Aprendizaje.

Del objetivo general se derivan los siguientes **objetivos específicos**.

- Identificar las metodologías y herramientas utilizadas en la definición y elaboración de los Modelos de Dominio Específico y Lenguajes de Dominio Específico en el desarrollo de los Sistemas de Gestión del Aprendizaje.
- Definir los elementos significativos del Modelo de Dominio Específico y Lenguaje de Dominio Específico en los Sistemas de Gestión del Aprendizaje.
- Elaborar el procedimiento para la definición de Modelos de Dominio Específico y Lenguaje de Dominio Específico en los Sistemas de Gestión del Aprendizaje.
- Validar el procedimiento para la definición de Modelos de Dominio Específico y Lenguaje de Dominio Específico en los Sistemas de Gestión del Aprendizaje.

Las **tareas de la investigación** a desarrollar para cumplir los objetivos trazados son:

- Recopilación de información acerca de los Sistemas de Gestión de Aprendizaje.
- Revisión de documentos relacionados con Modelos de Dominio Específico y Lenguaje de Dominio Específico.
- Búsquedas bibliográficas sobre las diferentes metodologías usadas para definir Lenguaje de Dominio Específico.
- Investigación sobre las diferentes herramientas usadas para Modelos de Dominio Específico y Lenguaje de Dominio Específico.
- Establecimiento de una comparación entre las herramientas que se usan para Modelos de Dominio Específico y Lenguaje de Dominio Específico.
- Organización de las etapas del procedimiento para la definición de Modelos de Dominio Específico y Lenguaje de Dominio Específico.
- Especificación de las actividades para llevar a cabo el procedimiento para la definición de Modelos de Dominio Específico y Lenguaje de Dominio Específico.

- Definición de los artefactos que se utilizarán en el procedimiento para la definición de Modelos de Dominio Específico y Lenguaje de Dominio Específico.
- Selección de los expertos para la aplicación del método Delphi.
- Elaboración y lanzamiento de los cuestionarios que serán aplicados a los expertos.
- Análisis del desarrollo práctico de la aplicación del método Delphi.

Por lo anteriormente puntualizado, surge la siguiente **idea a defender:** La elaboración de un procedimiento para la definición de Modelos de Dominio Específicos y Lenguajes de Dominio Específicos en los Sistemas de Gestión del Aprendizaje, permitirá guiar a los nuevos investigadores en el desarrollo de este proceso.

Métodos científicos de investigación:

En el desarrollo de la investigación se utilizan varios métodos científicos, como son:

Métodos teóricos:

- *Método Analítico - Sintético:* Es un método que consiste en la separación de las partes de un todo para estudiarlas en forma individual (Análisis), y la reunión racional de elementos dispersos para estudiarlos en su totalidad (Síntesis).

  En el método *Análisis* se separan los elementos relacionados con el Desarrollo de Software Dirigido por Modelos en sus partes constitutivas (Arquitectura Dirigida por Modelos, Factorías de Software, la Modernización Dirigida por Modelos y el Modelo Específico del Dominio) con el propósito de estudiar éstas por separado, así como las relaciones que las unen.

  El método *Síntesis* compone el Lenguaje de Dominio Específico a partir de la unión de los elementos principales que lo forman: la sintaxis abstracta, sintaxis concreta y la semántica del lenguaje.

- *De los Métodos Empíricos:*

  - ✓ Encuesta: Permite recopilar información sobre los principales problemas que presenta el centro FORTES. Se entrevista al 20 por ciento de la población y se utiliza dentro de las técnicas de muestreo la no probabilística, ya que se seleccionan solo algunos miembros de proyectos que cumplen con determinados criterios. De esta técnica se utiliza el muestreo intencional o de conveniencia ya que se selecciona directamente e intencionadamente los individuos de la población. También el método encuesta permite

conocer la opinión o criterios que dan un grupo de expertos para la aplicación del método Delphi.

El presente documento consta de tres capítulos:

Capítulo 1: Fundamento Teórico

Abarca todo el análisis bibliográfico realizado sobre los principales lenguajes y herramientas de metamodelado que se utilizan para definir los DSL, así como de investigaciones realizadas que se encuentran relacionadas con los Modelos de Dominio Específico y Lenguaje de Dominio Específicos para gestionar la investigación desde los Sistemas de Gestión de Aprendizaje.

Capítulo 2: Propuesta del Procedimiento

Se realiza un estudio detallado de las posibles etapas que pueden conformar el procedimiento. Se propone el procedimiento para definir el Lenguaje de Dominio Específico para los Sistemas de Gestión del Aprendizaje con el propósito de apoyar a los nuevos investigadores en el desarrollo de este proceso.

Capítulo 3: Validación del Procedimiento

Comprende el proceso de validación del procedimiento para definir el Lenguaje de Dominio Específico para los Sistemas de Gestión del Aprendizaje mediante el Método Delphi y el uso del criterio de un grupo de expertos pertenecientes a la universidad.

Capítulo 1: Fundamento Teórico

Introducción

"Investigar es ver lo que todo el mundo ha visto, y pensar lo que nadie más ha pensado"

Albert Szent-György (1893-1986)

El desarrollo de una investigación científica requiere de una base teórica que sustente y respalde el trabajo a desarrollar en el curso investigativo. En este capítulo se realiza un estudio detallado de los principios básicos del Desarrollo de Software Dirigido por Modelos, unido a ello, las diferentes vertientes que abarcan esta técnica, haciendo énfasis en el Modelo Específico del Dominio, además de los principales conceptos de los Sistemas de Gestión de Aprendizaje. Se analizan las herramientas y los diferentes lenguajes de metamodelado utilizados para definir Lenguajes de Dominio Específico.

1.1 Sistemas de Gestión de Aprendizaje

Un Sistema de Gestión del Aprendizaje (LMS, del inglés Learning Management System) es un software que automatiza la administración de acciones de formación. Registra usuarios, organiza los diferentes cursos en un catálogo, almacena datos sobre los usuarios, también provee informes para la gestión. Suministra al instructor un mecanismo para crear y distribuir contenido, monitorear la participación de los estudiantes y evaluar su desempeño. También ofrece a los estudiantes el uso de mecanismos de interacción como foros de discusión, videoconferencias o servicios de mensajería instantánea (3).

Los LMS en cierto sentido han sido la evolución de los Sistemas de Gestores de Contenidos (CMS, del inglés Content Management System) creados específicamente para la administración y control de información de usuarios a nivel corporativo. Desde su aparición ha sido producto de integración y creación de diferentes tipos de comités y grupos que a nivel mundial han permitido apoyar esta iniciativa para tratar de estandarizar los procesos de desarrollo de plataformas que se prestan para su uso en particular de formación (4).

Un LMS generalmente no incluye posibilidades de autoría (crear sus propios contenidos), se centra en gestionar contenidos creados por gran variedad de fuentes diferentes (3).

Es importante destacar que entre los LMS más conocidos de libre distribución se encuentran: SAKAI, ATUTOR, CLAROLINE, MOODLE y DotLRN (5).

1.2 Principios básicos de la tecnología DSDM

Aumentar el nivel de abstracción es uno de los avances claves en el progreso del desarrollo de software. Moverse a niveles más altos de abstracción brinda varios beneficios, incluyendo una mayor productividad y menos defectos. Este es el objetivo del Desarrollo de software Dirigido por Modelos (DSDM) (6).

El término *Desarrollo de Software Dirigido por Modelos* es utilizado para englobar a los diferentes paradigmas de desarrollo de software que tienen en común el uso de modelos para representar diferentes aspectos de un sistema de software con el propósito final de elevar el nivel de abstracción y de automatización. En el DSDM los modelos ya no son simples medios para describir software, sino la pieza fundamental de su desarrollo .Los modelos existentes en una fase (análisis, diseño) se transforman (automática, semiautomáticamente o manualmente) sucesivamente hasta derivar la aplicación (7). Las técnicas del DSDM no solo son útiles para crear nuevas aplicaciones sino que pueden ser aplicadas en la modernización o reingeniería del software. De esta manera, la información comúnmente manejada en tareas específicas de los procesos de evolución de software es representada mediante modelos y transformaciones de modelos, y aplicadas tanto para realizar la ingeniería inversa como la generación de los artefactos software del nuevo sistema (8).Los conceptos teóricos que respaldan esta nueva visión son: modelos, lenguajes específicos del dominio, metamodelado, transformaciones de modelos, entre otros.

Entre los paradigmas DSDM más destacados se encuentran Arquitectura Dirigida por Modelos, el Modelo Específico del Dominio, el enfoque de Factorías de Software y la Modernización Dirigida por Modelos.

- **Arquitectura Dirigida por Modelos**

La principal idea en la Arquitectura Dirigida por Modelos (MDA, del inglés Model Driven Architecture) es la separación de la especificación de la funcionalidad de una aplicación de su implementación sobre una plataforma concreta, con el objetivo de que el desarrollador solo se preocupe de la lógica del negocio y las herramientas específicas generen todo el código relacionado con las plataformas de implementación.

En este sentido, MDA propone que el desarrollador cree una descripción de la lógica del problema desde una perspectiva independiente de la computación (CIM, del inglés Computation Independent Model) que es donde mejor podrían encajar las reglas de modelado o metamodelo. Desde aquí se crean los modelos independientes de la plataforma (PIM, del

inglés Platform Independent Model). A partir de los modelos PIM se generan modelos específicos de una plataforma concreta (PSM, del inglés Platform Specific Model) y finalmente el código.

Cada modelo puede ser expresado en un lenguaje diferente y las trasformaciones CIM-PIM, PIM-PSM y PSM-código requieren de herramientas de transformación que reciben como entrada, además del modelo origen, una definición de transformación. Esta definición de transformación es expresada mediante un lenguaje de transformación (9).

- **Factorías de Software**

La combinación del desarrollo dirigido por modelos con las líneas de productos de software es uno de los enfoques más prometedores para lograr la aparición de una verdadera industria del software.

Las Factorías de Software representan una propuesta en este sentido, en el que modelos, DSL y transformaciones son usados para construir la infraestructura necesaria para crear familias de aplicaciones en vez de aplicaciones individuales (8). Estas se definen como una línea de producto software cuyo componentes de producción forman un entorno de desarrollo integrado (IDE, del inglés Integrated Development Environment) configurado para permitir el desarrollo rápido de los miembros de la familia de productos. Es decir, dado una línea de producto concreta, por ejemplo aplicaciones de comercio electrónico, una factoría de software está formada por un conjunto de elementos software (procesos, patrones, DSL y herramientas) que son instalados en un entorno de desarrollo para favorecer el desarrollo de aplicaciones concretas para ese dominio de forma productiva y generando código de calidad.

- **Modernización Dirigida por Modelos**

La Modernización de Software Dirigida por Modelos ha surgido como una nueva disciplina centrada en la utilización de técnicas de Desarrollo de Software Dirigido por Modelos en los procesos de evolución de software.

Las técnicas del DSDM no solo son útiles para crear nuevas aplicaciones sino que pueden ser aplicadas en la modernización o reingeniería del software. De este modo, la información comúnmente manejada en tareas propias de los procesos de evolución de software es representada mediante modelos y transformaciones de modelos, y aplicadas tanto para realizar la ingeniería inversa como la generación de los artefactos software del nuevo sistema.

- **Modelo Específico del Dominio**

El Modelo Específico del Dominio (DSM, del inglés Domain Specific Modeling) plantea elevar el nivel de abstracción por encima de los lenguajes de programación, de modo que sea posible especificar la solución mediante conceptos que realmente aparecen en el dominio (9) y esconde herramientas de programación innecesarias para describir la solución (accesos a memoria, punteros, variables, métodos); siendo el código final generado automáticamente a partir de esas especificaciones de alto nivel.

La automatización es factible debido a que tanto el lenguaje y los generadores se ajustan a los requisitos, es decir, se precisa el dominio o espacio de la solución al que se puede aplicar el lenguaje, a veces reduciéndose al ámbito de los productos de una empresa. Se ha hecho necesario el uso de este paradigma de desarrollo de software para disminuir la incidencia de errores humanos y aumentar la facilidad para captura de requerimientos.

Ventajas que ofrece el uso del paradigma DSM

Con el uso del DSM se obtiene un aumento significativo de la productividad en la tarea de programar ya que aporta beneficios como:

- Automatización: La aplicación del DSM a los diferentes lenguajes de programación puede permitir un nuevo aumento de productividad ya que un elemento en el modelo supondrá varias instrucciones en el lenguaje.
- Correspondencia Directa. Existe una correspondencia directa entre el modelo conceptual de la solución y el modelo DSM. Se facilita en gran medida la expresión de la solución al problema reduciendo el salto semántico entre el modelo conceptual que representa la solución para cierta tarea y el código o diagrama que la expresa en un formato ejecutable.
- Mantenimiento minimizado. Debido a la correspondencia directa resulta más fácil la modificación de un modelo cuando se producen cambios en los requisitos funcionales.
- Reducción del tiempo de prueba. Del modelo se generará el código que ejecute la solución tal y como se ha expresado. Deja de ser necesario por tanto el control de excepciones y de errores en la escritura del código, permitiendo al programador concentrarse únicamente en la solución a su problema.

Mientras que MDA, las Factorías de Software y el DSM son enfoques de creación de nuevas aplicaciones, la Modernización Dirigida por Modelos se centra principalmente en la reingeniería de software. Las Factorías como su propio nombre lo dice es la industria que se encarga de crear familias de aplicaciones en vez de aplicaciones individuales, además que se cuentan con

pocas herramientas para crear factorías de software. MDA al igual que DSM son las técnicas más estudiadas y completas, pero este primero no proporciona suficientes guías metodológicas, solo se define la estrategia general para la transformación PIM–PSM y tampoco garantiza la separación de conceptos (10), mientras que DSM aporta grandes beneficios como la creación de Lenguajes de Dominio Específico y sus generadores de código, que serán usados para especificar una solución de alto nivel, de forma más productiva que escribiendo el código en un lenguaje de programación.

1.3 Lenguaje de Dominio Específico o Lenguaje de Modelado

Un Lenguaje de Dominio Específico (DSL, del inglés Domain Specific Language) se diseña para suplir una necesidad muy específica dentro de un dominio de aplicación concreta. El objetivo es la creación de lenguajes especializados para poder, de forma más sencilla, modelar artefactos de software en el ámbito para el que fue prescrito el DSL, y sea capaz de generar código de estos modelos (gráficos o texto). El ámbito de aplicación determinará qué conceptos son aquellos que deberán aparecer en un DSL para permitir el modelado de dichos elementos.

Son estructuralmente similares a los de programación. Así como un lenguaje de programación da errores de compilación por no respetar su sintaxis, lo mismo pasará en el modelado si no se respetan las formas establecidas. La mayoría de los lenguajes de modelado no se ejecutan, pero es posible la generación de código a partir de modelos. Un lenguaje de modelado se define mediante un metamodelo o descripción de los elementos del lenguaje y de las relaciones existentes entre ellos.

1.3.1 Tipos de Lenguajes de Dominio Específico

Desde el punto de vista del formato del lenguaje:

Textuales*:* La mayoría de los lenguajes informáticos son textuales y están formados por un conjunto ordenado de sentencias. Un ejemplo muy conocido de DSL textual es SQL[1] utilizado para realizar consultas a una base de datos. Una forma de crear DSLs textuales es mediante la creación de una determinada gramática y posteriormente crear o utilizar un parser [2] para dicha gramática, para en etapas posteriores poder interpretar el DSL o generar código.

[1] SQL: El lenguaje de consulta estructurado o SQL (del inglés Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en estas.
[2] Parser: Un analizador sintáctico (en inglés parser) es una de las partes de un compilador que convierte el texto de entrada en otras estructuras (comúnmente árboles), que son más útiles para el posterior análisis y capturan la jerarquía implícita de la entrada.

Gráficos*:* En los últimos años están ganando gran aceptación los lenguajes gráficos, podrían citarse como ejemplo el Lenguaje Unificado de Modelado (UML, del inglés Unified Modeling Language). La creación de un lenguaje gráfico es similar a la de un lenguaje textual, la única diferencia es que en lugar de usar texto para representar los conceptos, se utilizan conectores y figuras simples.

Desde el punto de vista del dominio del problema:

Horizontales*:* Los DSL horizontales son aquellos en los que el cliente que utilizará el lenguaje no pertenece a ningún dominio específico. Un ejemplo son los editores visuales de entornos de desarrollo que permiten generar interfaces de usuario automáticamente (por ejemplo Windows Forms de Visual Studio[3]).

Vertical*:* A diferencia de los DSL horizontales, el cliente que utilizará el lenguaje pertenece al mismo dominio que el lenguaje en sí. Como en el ejemplo anterior para un lenguaje de definición de encuestas, los usuarios finales serían los expertos en estadística encargados de definir dichas encuestas (11).

Un DSL consta de tres elementos: la sintaxis abstracta que define los conceptos del lenguaje y las relaciones entre ellos, la sintaxis concreta que provee de una notación textual o gráfica, y la semántica del lenguaje que normalmente es definida a través de la traducción a conceptos de otro lenguaje destino cuya semántica se conoce.

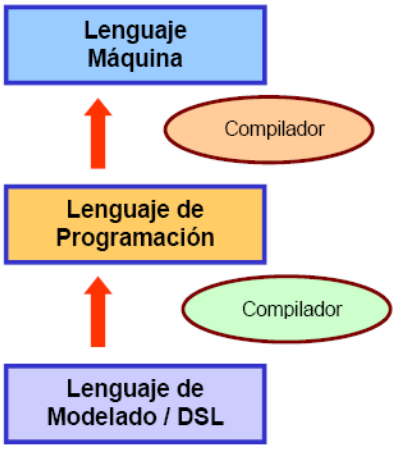

Figura 1: Un nuevo nivel de abstracción en los lenguajes (9)

[3]Visual Studio: Es un entorno de desarrollo integrado (IDE) para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic.

Como muestra la Figura 1, el uso de un DSL requiere, lógicamente, la existencia de un compilador que transforme el código expresado con un DSL en un lenguaje de programación (o en otro DSL, como paso previo a la generación de código).

Soluciones similares para definir Lenguaje de Dominio Específicos

Durante el estudio realizado fueron consultados algunos trabajos como la tesis doctoral "Modelado Específico de Dominio para la Construcción de Learning Objects Independientes de la Plataforma" (4) donde Carlos Enrique Montenegro a pesar de utilizar las herramientas y los lenguajes de metamodelado más propicios, realiza el modelado específico de dominio para la construcción de Módulos de LMSs independientes de la plataforma pero de manera muy general ya que no se especifica claramente la definición de los 3 elementos principales para la creación de los lenguajes de modelado.

Otro de los trabajos consultados fue "Aplicación de Técnicas de Ingeniería de Lenguajes al Campo del Modelado Educativo" (12) donde Iván Martínez Ortiz encuadra su tesis dentro del campo de la enseñanza apoyada por la tecnología, que globalmente se denomina por el término en inglés *e-learning*, con la peculiaridad del uso de técnicas y herramientas de Lenguajes de Dominio Específicos para el dominio educativo. En su tesis solamente se adoptan los principios básicos de las modernas tendencias de desarrollo dirigido por modelos pero no definen claramente el proceso para la definición de Lenguajes de Dominio Específico para las técnicas de los lenguajes de Modelado Educativo como los LMS.

Es importante mencionar que durante el estudio investigativo realizado no se encontró ninguna documentación donde se detallara alguna metodología o procedimiento que ayudara a una correcta definición del Modelo de Dominio Específico y Lenguaje de Dominio Específico, los trabajos consultados solo se guían por lo que plantea OMG[4] (del inglés Object Management Group) y se preocupan más por la aplicación que generan que por explicar el método para lograr su solución.

La creación de un DSL no es sencilla aunque se dispongan de las herramientas apropiadas. La teoría subyacente a la creación de lenguajes de modelado o DSL se denomina metamodelado y los diferentes aspectos que abarca son: lenguajes de metamodelado, sintaxis abstracta, sintaxis concreta, semántica y transformaciones.

[4] OMG (*Object Management Group*): Grupo de empresas dedicadas al cuidado y establecimiento de diversos estándares de tecnologías orientadas a objetos.

1.4 Metamodelo

Un metamodelo define la sintaxis abstracta de un lenguaje que establece los conceptos y relaciones entre ellos, e incluye las reglas que determinan qué es un modelo bien formado. Algunas de las principales definiciones de metamodelo son:

Seidewitz: Un metamodelo es un modelo de especificación para una clase de sistemas que cumplen que cada uno de ellos es en sí mismo un modelo válido en un cierto lenguaje de modelado (13).

Mellor: Un metamodelo es un modelo que define la estructura semántica y restricciones de una familia de modelos (14).

MOF v2.0: Un metamodelo es un modelo que define el lenguaje para definir un modelo (15).

A partir de estas definiciones se puede deducir que un metamodelo es una representación que describe un dominio de sistemas, limitando completamente los elementos que pueden formar parte de un modelo de un sistema en particular y sus relaciones. Es un modelo que describe un lenguaje de modelado con el que se describen otros modelos.

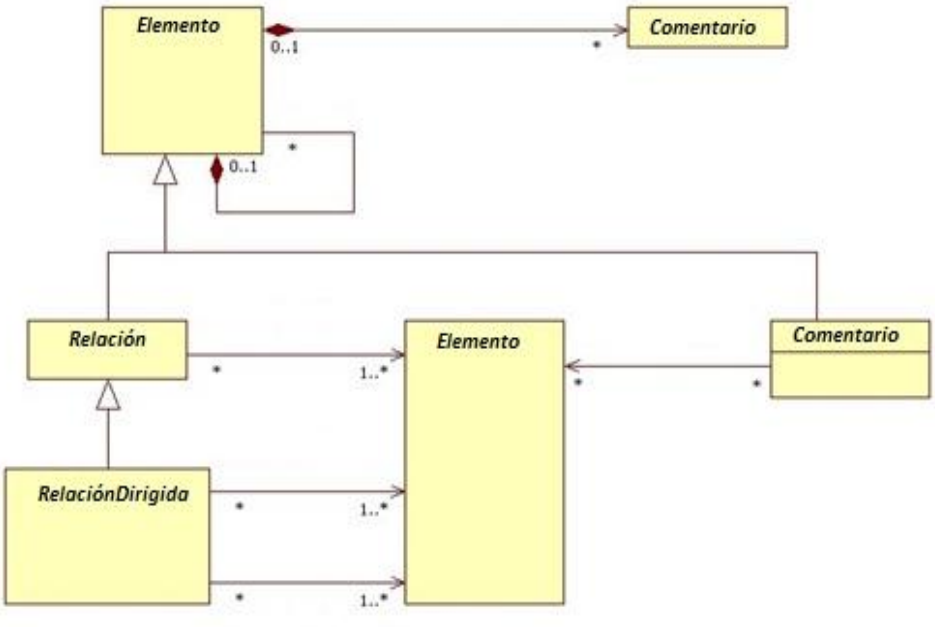

Figura 2: Un metamodelo simple "esquema relacional"

Como se muestra en la Figura 2 un metamodelo es un modelo de un modelo. Por tanto un modelo es una abstracción y un metamodelo es una abstracción mayor, donde el propósito es definir las propiedades de los modelos en sí mismo.

Usos de los metamodelos:

- Como esquema de datos semánticos que necesitan ser intercambiados o almacenados.
- Como un lenguaje que soporta un método en particular o proceso.
- Como un lenguaje para expresar semántica adicional de información existente. Hay cuatro diferentes enfoques para describir la semántica de un metamodelo:

✓ Traducción: Se traducen los conceptos en conceptos de otro lenguaje que tenga una semántica precisa.

✓ Operacional: Modelado del comportamiento operacional de los conceptos.

✓ Extensional: Se extiende la semántica de los conceptos de otro lenguaje.

✓ Denotacional: Asocia objetos matemáticos a cada objeto del lenguaje.

Un modelo es siempre conforme a su metamodelo, el metamodelo también debe tener su propio meta-metamodelo (que contiene los elementos básicos para describir metamodelos), y al que por supuesto tiene que ser conforme y así sucesivamente. La única forma para que el enfoque DSM pueda ser aplicada en la práctica pasa por romper esta relación recursiva en la definición de metamodelos (13).

1.4.1 Lenguaje de metamodelo

Lenguaje para describir otros lenguajes. Los lenguajes de metamodelo más extendidos son MOF (del inglés Meta-Object Facility) de OMG y Ecore para Eclipse. Una definición de un lenguaje a través de un lenguaje de metamodelo se denomina metamodelo. Los lenguajes de meta-modelado se definen con metamodelos.

Con el objetivo de entender más el metamodelado. La Figura 3 ilustra las relaciones entre un modelo, el DSL con el que está definido, y la de éste con su meta-metamodelo. Un lenguaje de metamodelado también se define mediante un meta-metamodelo creado con el propio lenguaje, de manera que un metamodelo también conforma a su meta-metamodelo.

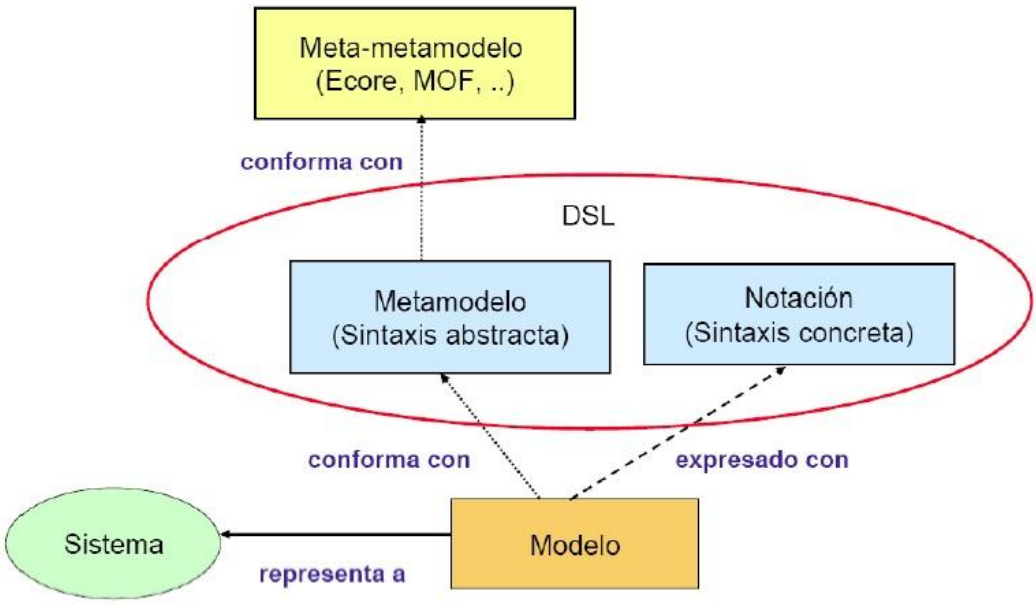

Figura 3: Relación entre modelo, DSL y meta-metamodelo (8)

El Meta-Object Facility (MOF)

MOF es un lenguaje para crear metamodelos. Según OMG proporciona el entorno en que los modelos pueden ser exportados e importados entre herramientas, almacenados y extraídos de repositorios, transformados entre distintos meta-modelos y usados para generar código.

Ha sido la herramienta creada para la construcción de metamodelos por parte de OMG (16). Es un lenguaje abstracto cuyo objetivo es especificar, construir, manejar y almacenar metamodelos, es decir MOF es un meta-metamodelo. Su uso facilita las labores de exportación e importación entre aplicaciones, transformaciones, entre otras cosas.

Según algunas bibliografías MOF es:

- Un estándar desarrollado por el OMG, cuyo objetivo inicial era proporcionar un metamodelo para describir UML y servir como pieza central de la iniciativa de MDA. En la arquitectura de 4 capas sintácticas definida por el OMG, MOF ocupa el nivel M3, el más alto de todos. Es la herramienta fundamental para el enfoque de DSM (17).
- El lenguaje de meta-modelado definido por OMG. MOF es un lenguaje para crear metamodelos (por ejemplo, el metamodelo de UML ha sido definido con MOF), y es por tanto un elemento básico de MDA. En realidad, la especificación de MOF distingue entre el lenguaje de EMOF (del inglés Essential MOF, el núcleo de MOF) y CMOF (del inglés Complete MOF). EMOF es un subconjunto de CMOF, que proporciona las primitivas

básicas de metamodelado, mientras que CMOF añade mecanismos avanzados de metamodelado, como por ejemplo extensibilidad y reflexión (9).

- Está diseñado dentro de una arquitectura de 4 niveles o metaniveles como se dice en algunas bibliografías M3, M2, M1, M0, en el ámbito de MDA (18).

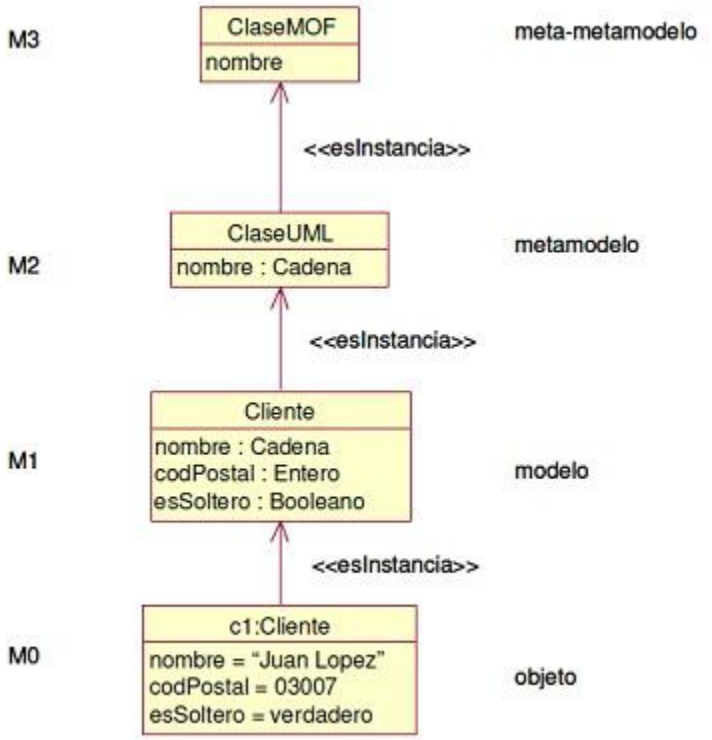

Figura 4: Arquitectura de las 4 capas de MOF (19)

Capa M0: **El objeto**. En esta capa se define el sistema en ejecución, en él vive un conjunto de objetos, datos o instancias reales. Por ejemplo si se especifica un sistema de comercio electrónico, entre otros conceptos existen los artículos del sistema. Una instancia mostrada en el ejemplo serían un cliente de nombre "Juan López", su código Postal es 03007 y es soltero.

Capa M1: **El modelo.** Esta capa contiene el modelo que representa un sistema de software concreto. El modelo puede tratarse de un modelo de clases UML, un modelo de navegación o un modelo que cumpla la condición de que se pueda meta-modelar cada uno de los elementos que lo componen. Siguiendo el ejemplo anterior de comercio electrónico, el modelo de dominio tiene que representar a la clase Cliente que contendría los atributos *nombre* de tipo cadena, *codPostal* de tipo entero y *esSoltero* de tipo booleano.

Capa M2: **El metamodelo.** Los modelos definidos en esta capa son denominados meta-modelos. La función de meta-modelo es modelar los elementos que constituyen los modelos

del sistema. Es decir, definir el lenguaje con el que se escriben los modelos de los sistemas, especificando las restricciones entre ellos y las posibles relaciones. Un ejemplo es el metamodelo definido por la superestructura de UML 2.0 (20), en la que mediante un modelo de clases se definen todos los elementos que definen los 11 modelos UML 2.0

Capa M3: **El meta-metamodelo**. La OMG propuso el estándar denominado MOF. En la Figura 4 se muestra como se define el concepto de Class MOF que define el Class del metamodelo de UML.

El interés actual por los lenguajes de transformación y las herramientas de transformación (o compiladores de modelos) recuerda a la situación vivida a principios de los sesenta con la aparición de los primeros lenguajes de programación y toda la teoría relacionada con los compiladores e intérpretes. Como se dijo anteriormente, son muchos los que consideran que el DSDM plantea un salto en el nivel de abstracción del mismo orden de magnitud que con los lenguajes de programación.

GOPRR

GOPRR (Grafo Objeto Propiedad Relación Rol) es un lenguaje de meta-modelado que permite la definición de metamodelos y modelos usando grafos en una notación visual. Los principales elementos de este lenguaje son:

- Grafo. Este elemento es el usado como raíz y contiene a todos los demás. Los grafos pueden conectarse con otros grafos a través de estructuras llamadas *Explosión*. Estos segundos grafos pueden ser usados para agrupar elementos. Los grafos tienen *Propiedades*.
- Objeto. Se usa para representar cualquier objeto genérico. Tiene varias *Propiedades*. Además los Objetos poseen herencia y pueden crear sus propias jerarquías.
- Relación. Sirve para relacionar cualquier número de Roles. Por tanto, representan relaciones n-arias. Las Relaciones también tienen Propiedades.
- Rol. Se usa para representar los extremos de las Relaciones. En M1, cada Rol conecta un Relación con un Objeto de una clase de objetos determinada. Los Roles también tienen *Propiedades*.
- Propiedad. Almacena un valor de tipo primitivo (5).

Ecore

Ecore es un lenguaje relativamente simple que tiene una gran semejanza con la definición de subconjunto de clase de UML. Permite definir metamodelos y es utilizado como elemento central del Eclipse Modeling Framework (EMF) que constituye una infraestructura básica para crear herramientas y soluciones DSDM. Realmente, Ecore es un subconjunto de MOF (específicamente de EMOF) que fue el lenguaje de metamodelado definido por OMG para definir formalmente UML (8).

Los metamodelos y modelos usados por EMF se representan con documentos XML y ofrecen serialización, notificación y reflexión a cualquier metamodelo.

Elementos de Ecore

EClass. Contiene EAttributes y EReferences. Las instancias de EClass pueden extender otras instancias de Eclass, heredando los atributos.

EAttribute. Representa atributos. Contiene valores de tipos primitivos (enteros, cadenas de caracteres, etc.).

EReference. Representa relaciones binarias entre dos instancias de EClass. La primera instancia de EClass contiene una instancia de EReference que señala a lo otra instacia de EClass. Si la instancia de EReference es múltiple, quiere decir que en el modelo se podrá instanciar varias veces en la EClass contenedora.

EPackage. Representa un paquete que agrupa elementos del metamodelo.

Existen herramientas que tratan automáticamente metamodelos y modelos. Por ejemplo, EMF genera código automáticamente para editores de modelos para un lenguaje de modelado, definido con un metamodelo en Ecore. Otros ejemplos de plugin de Eclipse[5] para transformar este tipo de modelos, son ATL (del inglés Atlas Transformation Language)[6] y GMF (del inglés Graphical Modeling Framework)[7] (21).

[5]Eclipse: Plataforma abierta y de libre distribución. En su desarrollo participan importantes empresas como Borland, IBM, Intel, Motorola, etc.

[6]ATL: Framework para la transformación de modelos. En un lenguaje mixto: imperativo-declarativo. Tiene un depurador de transformaciones y distingue de mayúsculas y minúsculas.

[7]GMF: Framework que permite generar editores visuales a partir de metamodelos ECore y otros modelos adicionales que describen aspectos de representación de las entidades.

1.5 Herramientas de Metamodelado

1.5.1 MetaEdit+

La herramienta MetaEdit+ ha sido desarrollada por la empresa Metacase[8] y destinada a la creación de DSL. Esta provee un entorno gráfico, amigable y fácil de usar, el cual integra diferentes editores, navegadores de datos y otras herramientas que inicialmente son suficientes para crear cualquier DSL, crear modelos expresados en ese DSL y generar código. El lenguaje de metamodelado usado por MetaEdit+ es GOPRR, estos son los meta-tipos que proporciona para describir los lenguajes de modelado. Éste proporciona un conjunto de herramientas potente y a la vez sencillo, de acuerdo con los conceptos del lenguaje de metamodelado GOPRR.

Dichas herramientas de desarrollo son:

• **Herramienta de objetos (***ObjectTool***)**: Especifica tipos de objetos que son componentes básicos de los metamodelos.

• **Herramienta de relaciones (***RelationshipTool***)**: Indica los conectores entre tipos de objetos.

• **Herramienta de roles (***Role Tool***)**: Indica los tipos de objetos que intervienen en una relación jugando un determinado rol.

• **Herramienta de puertos (***Port Tool***)**: Especifica semántica adicional respecto a cómo los tipos de roles se conectan con los tipos de objetos.

• **Herramienta de grafos (***GraphTool***)**: Una vez definidos los tipos de objetos, relaciones, roles y puertos, permite establecer las reglas para la conexión de todos estos elementos.

• **Herramienta de propiedades (***PropertyTool***)**: Permite modificar las propiedades de cualquier tipo de elemento, algo posible con las propias herramientas, y también permite crear nuevos tipos de datos (22).

Como en cualquier herramienta de esta naturaleza, existe un repositorio destinado a almacenar los elementos de los metamodelos y de los modelos, y que por tanto juega un papel esencial (9).

[8]Metacase: Empresa especializada en la gestión de proyectos que engloban la concepción, el desarrollo y la implantación de Plataformas direccionadas para la Gestión Global de todos los Flujos Financieros de grandes compañías.

Ventajas de MetaEdit+

MetaEdit+ proporciona un lenguaje de metamodelado propio (GOPRR) para expresar cómo se realiza la generación de código, informes de consistencia o cualquier otro tipo de salida a partir del modelo. MetaEdit+ es un framework[9] que ha sido íntegramente diseñado para el propósito que cumple, y constituye una herramienta autosuficiente, estable, práctica y madura, ya que cuenta con la experiencia de quince años en el mercado. Hay buena documentación sobre la herramienta y tiene baja necesidad de usar un lenguaje externo con cierta frecuencia.

Desventajas de MetaEdit+

A diferencia de otras herramientas, el entorno de trabajo que utiliza no diferencia claramente los conceptos de sintaxis abstracta y sintaxis concreta. De esta forma, cuando se define un nuevo elemento del metamodelo y sus propiedades, se diseña también el aspecto visual del mismo. Es una herramienta no gratuita, solo es posible descargar una versión de prueba que caduca en el plazo de un mes.

1.5.2 DSL Tools

DSL Tools es una propuesta de herramienta de metamodelado creada por la Microsoft[10] que permite crear DSL, integrándose dentro del entorno de desarrollo Visual Studio 2005 en sus diferentes distribuciones. Esto simplifica y reduce el tiempo requerido para tal tarea, al mismo tiempo que favorece el entendimiento del DSL en particular, obviando detalles de implementación que podrían dificultar el desarrollo del mismo. En particular esta tecnología, que se puede considerar como "herramientas para crear herramientas", facilita la tarea de definir un DSL como así también reducir el esfuerzo y los conocimientos necesarios para crear los editores gráficos y compiladores (23).

La implementación de DSL con la herramienta de metamodelado DSL Tools consiste, a grandes rasgos, en la definición de una sintaxis abstracta que modele todos los elementos del lenguaje, en base a la cual se define una sintaxis concreta. No obstante

Microsoft no utiliza esta denominación para estas sintaxis, sino que utiliza los conceptos *Domain Model* and *Designer*, lo que traducido sería Modelo de dominio y Diseñador, para referirse a las sintaxis abstracta y concreta respectivamente.

[9] Framework o infraestructura digital: Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos.
[10] Microsoft: Empresa multinacional dedicada al sector de la informática.

La perspectiva de Microsoft se basa en la construcción de DSL gráficos para la edición de modelos, en los que el desarrollador debe definir los siguientes elementos:

- *Modelo de dominio*

Se refiere al metamodelo o sintaxis abstracta del DSL definido con un lenguaje de metamodelado que se utiliza a través de un editor de metamodelos y que tiene una naturaleza similar a MOF y a otros lenguajes de metamodelado.

- *Figuras del diagrama (shapes) y conectores (connector) asociados al modelo de dominio*

Se corresponden con la representación gráfica de los artefactos modelados, es decir, con la sintaxis concreta del DSL. Cada figura agregada en un diagrama representa una instancia concreta de un concepto determinado dentro del dominio, por ejemplo un estado en una máquina de estados, una clase en un diagrama de clases, etc.

- *Reglas de validación*

Un DSL no consta únicamente de una representación gráfica que se corresponde con la parte léxica del mismo.

- *Plantillas de texto*

De la construcción de un modelo nos interesa exclusivamente la obtención de un código resultado interpretable por una máquina, ya que este es el verdadero objetivo de la generación automática de código. Las plantillas de texto serán la guía que defina el proceso de generación de código a partir de un modelo de entrada.

Con DSL Tools la creación de un nuevo lenguaje parte de la base de otros lenguajes plantilla que se incluyen, dando al programador la posibilidad de escoger aquel que considere más apropiado con el fin de aprovechar el mayor número de características posible.

Ventajas de DSL Tools

Otra posibilidad en torno a la idea de lenguajes que recurren a varios diagramas, y que no contempla las DSL Tools, es la de permitir que ciertos elementos de un diagrama tengan asociado otro diagrama. Un ejemplo típico lo encontramos en UML, que permite asociar un diagrama de estados para cada clase de un diagrama de clases.

El lenguaje de metamodelado de las DSL Tools es simple pero potente. Utiliza unos pocos conceptos básicos (clase, atributo, referencia, embebido, herencia) con los que construir los metamodelos, de modo que aunque se trate de un lenguaje propietario, no es difícil de

aprender. Además ayuda que utiliza conceptos que son familiares de lenguajes como MOF, lo que facilita un poco más su comprensión.

Desventajas de DSL Tools

Según las bibliografías consultadas delatan que la herramienta definitivamente no está finalizada, sino que está en fase de desarrollo. Presenta un nivel de madurez bajo y existe escasa documentación de la misma (22). El manejo del entorno de DSL Tools, al tratarse de una herramienta integrada en un entorno como el Visual Studio que dispone de numerosas opciones le añade complejidad a la herramienta.

1.5.3 Eclipse

Eclipse nace como un entorno de desarrollo para Java[11] y se ha convertido en una plataforma que permite el desarrollo y la integración de herramientas de desarrollo. En su desarrollo participan importantes empresas como Borland, Intel y Motorola.

La principal ventaja de Eclipse radica en que su arquitectura está diseñada de forma que la mayoría de la funcionalidad proporcionada está localizada en plugin o en un conjunto de plugins relacionados. Esto hace a Eclipse una herramienta extensible.

Ventajas de Eclipse

Es de libre distribución, ampliable y configurable gracias a un diseño modular, fácil de instalar y existen versiones para casi cualquier Sistema Operativo. Sirve de plataforma de desarrollo para, numerosos lenguajes de programación como: C, C++, PHP, Ruby entre otros. La ventaja principal se halla en la gran variedad de plugins que se le pueden añadir, desde plugin para crear interfaces gráficas hasta descompiladores.

Eclipse cuenta con una comunidad muy activa, que es capaz de resolver prácticamente cualquier dificultad, además de brindar un servicio de actualizaciones y parches que soluciona todos aquellos problemas de incompatibilidades y de integración que se pudiera presentar.

Desventajas de Eclipse

El principal inconveniente, común a otros IDE en mayor o menor medida, es el consumo de recursos del sistema por lo que tiende a demorar en cargar. Otro inconveniente es la compatibilidad de versiones, muchos plugins requieren versiones específicas para funcionar

[11] Java: Lenguaje de programación. Fue la primera plataforma informática creada por Sun Microsystems en 1995. Es la tecnología subyacente que permite el uso de programas punteros, como herramientas, juegos y aplicaciones de negocios.

correctamente, siendo un poco complicado en proyectos grandes poder migrar de una versión vieja a una más nueva.

Luego de realizar un estudio sobre las herramientas existentes para el metamodelado se concluye que:

Existe una escasa variedad de herramientas para el metamodelado y todas con diversas características. En cuanto a DSL Tools la herramienta no está finalizada, presenta un nivel de madurez bajo y existe escasa documentación de la misma. Otras como Metaedit+ que son autosuficientes, estables y maduras, no diferencia claramente los conceptos de sintaxis abstracta y sintaxis concreta y además es una herramienta no gratuita. Eclipse cuenta con las funcionalidades requeridas para realizar correctamente el propósito para el cual fue diseñado ya que a diferencia de las demás herramienta, es de libre distribución y se le pueden añadir una gran variedad de plugins. DSLTools y MetaEdit+ para generar el código a partir de un modelo lo realizan mediante informes o plantillas de texto, lo que hace más engorrosa la transformación, mientras que Eclipse utiliza un framework para simplificar el trabajo.

1.6 Tipos de Transformaciones

Una transformación está asociada a un metamodelo de entrada y a un metamodelo de salida. Recibe como entrada un modelo conforme a un metamodelo de entrada y produce como salida otro modelo conforme al metamodelo de salida. Para la mejor comprensión se pueden definir 2 tipos de transformaciones (24):

1- Transformación Modelo a Modelo (M2M): Convierten la información de un modelo origen (o un conjunto de modelos) a un modelo destino (o un conjunto de modelos).

Básicamente existen dos lenguajes para las transformaciones entre modelos: QVT que ha sido creado por la OMG y ATL que se enmarca dentro de EMF.

QVT (del inglés Query View Transformation): Es un estándar propuesto por la OMG para resolver el problema de la transformación de modelos. Está compuesto de un conjunto de lenguajes estándar de dominio específico (DSL) que propone la OMG para la transformación de modelos. Las transformaciones que se definen están basadas en metamodelos MOF. Cabe destacar que QVT se centra en transformaciones Modelo a Modelo quedando las Modelo a Texto fuera del estándar (16).

ATL (del inglés Atlas Tranformation Language): Es un lenguaje para transformaciones de modelos, especificando un metamodelo y una sintaxis concreta de tipo textual. En el contexto

de DSM, ATL proporciona una forma de especificar como producir un conjunto de modelos destino a partir de un conjunto de modelos origen. Es un lenguaje que posibilita el expresar mapeos entre los elementos del modelo destino y los del modelo origen de forma sencilla. Una transformación ATL se compone de reglas que definen sobre qué elementos del modelo origen se aplican y que elementos del modelo destino se producen (11).

2- Transformación Modelo a Texto (M2T): Convierten cada elemento del modelo origen en definiciones de texto.

Java Emitter Template (JET): JET es la herramienta que provee por defecto EMF para la generación código en forma automática. Requiere que se especifiquen plantillas, como entrada para el proceso de traducción, indicando el formato que tendrá el código generado. Estas plantillas se escriben con una sintaxis muy parecida a la de JSP[12], lo que hace que sean fáciles de comprender para el común de los desarrolladores Java. De esta manera, JET puede utilizarse para generar cualquier tipo de archivo de texto, como SQL, XML, código Java y cualquier otro tipo de representaciones textuales

MOFScript: El lenguaje de transformación MOFScript, fue enviado al pedido de propuestas de lenguajes de transformación de modelo a texto lanzado por el OMG, pero que no resultó seleccionado. MOFScript está basado en QVT, es un refinamiento del lenguaje operacional de QVT. Es un lenguaje textual, basado en objetos y usa OCL para la navegación de los elementos del metamodelo de entrada. Además, presenta algunas características avanzadas, como la jerarquía de transformaciones y mecanismos de rastreo.

Algunas características de MOFScript son:

- ✓ El lenguaje permite especificar mecanismos de control básicos como iteraciones y sentencias condicionales.
- ✓ La herramienta tienen la habilidad de generar texto a partir de cualquier modelo basado en un metamodelo definido en MOF, como por ejemplo, modelos UML o cualquier otro metamodelo.
- ✓ La herramienta permite especificar el archivo salida para la generación del texto.
- ✓ La herramienta mantiene la trazabilidad entre los modelos y el texto generado.
- ✓ El editor de scripts posee ayuda para completar el código.
- ✓ La ingeniería inversa todavía no es parte de la herramienta.

[12] Java Server Pages (JSP): Es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo.

Otro de los lenguajes de transformación modelo a texto es el lenguaje de plantilla, este se realiza en dos pasos. En primer lugar, el motor crea una clase temporal, que se conoce como clase de transformación generada. Esta clase contiene el código que generan las directivas y los bloques de control. Después, el motor compila y ejecuta la clase de transformación generada para producir el archivo de salida.

Este proceso toma un archivo de plantilla de texto como entrada y genera un nuevo archivo de texto como salida. Por ejemplo, puede utilizar plantillas de texto para generar código de Visual Basic[13] o C#[14] o bien para generar un informe HTML[15] (del inglés HyperText Markup Language) (25).

Conclusiones parciales

En este capítulo se abordaron los principios básicos del paradigma de Desarrollo de Software Dirigido por Modelos, sus conceptos y elementos principales, además de las técnicas para la definición de DSM y DSL en los Sistemas de Gestión del Aprendizaje. Se concluyó que de las cuatro vertientes principales del DSDM se utilizó para el desarrollo de la propuesta DSM por los grandes beneficios que aporta al aumento de la productividad disminuyendo la incidencia de errores humanos y facilitando la captura de requerimientos. Durante este estudio investigativo no se encontró ninguna documentación donde se detallara alguna metodología o procedimiento que ayudara a una correcta definición de DSM y su correspondiente DSL, los trabajos consultados solo se guían por lo que plantea OMG y se preocupan más por la aplicación que generan que por explicar el método para lograr su solución.

[13] Visual Basic: Lenguaje de programación dirigido por eventos, desarrollado por Alan Cooper para Microsoft.
[14] C#: Lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET
[15] HTML (HyperText Markup Language): Lenguaje de marcado predominante para la elaboración de páginas web.

Capítulo 2: Propuesta del Procedimiento

Introducción

En este capítulo se presenta la propuesta de solución del problema planteado en la Introducción del presente trabajo investigativo, que radica en la creación de un procedimiento para la definición de Modelo de Dominio Específico y Lenguaje de Dominio Específico para los Sistemas de Gestión del Aprendizaje que permitirá guiar a los nuevos investigadores en el desarrollo de este proceso. Se especificarán detalladamente las fases por las que estará regido el procedimiento y una argumentada descripción de los diferentes elementos que formarían parte de la propuesta final de la investigación.

2.1 Consideraciones generales

Sobre la base de los elementos abordados en el capítulo 1, el apoyo de algunos trabajos como los tratados en el epígrafe de soluciones similares y teniendo en cuenta las características fundamentales de los Sistemas de Gestión del Aprendizaje, el Modelo de Dominio Específico y Lenguaje de Dominio Específico, se realizará la propuesta para llevar a cabo un proceso de definición de los mismos. Aunque en el procedimiento que se dará a continuación no necesariamente se tendrá en cuenta la creación de la ontología[16], se recomienda que en futuros trabajos se pueda utilizar la ontología para crear el metamodelo, una vez alcanzada la madurez necesaria en el paradigma de Desarrollo Dirigido por Modelos.

Para comenzar la construcción de un LMS es necesario tener presente algunas buenas prácticas como:

1. El sistema debe ser de código abierto.

2. Debe ser accesible a través de un navegador Web estándar.

3. Las opciones de autoría (si las posee) así como el resto de funciones del sistema deben poder ser utilizadas sin la necesidad de comprar ningún plugin o visualizador adicional.

4. Debe existir funciones básicas para la administración de usuarios.

5. El sistema debe ofrecer una función de autenticación.

6. El sistema debe ofrecer gestión de permisos.

[16] Ontología: Formulación de un exhaustivo y riguroso esquema conceptual dentro de uno o varios dominios dados.

7. El sistema debe estar abierto a la localización.

8. El alumno debe poder interaccionar a través del navegador con el profesor, el sistema y otros alumnos. La comunicación debe poder ser electrónica.

9. Debe existir funciones básicas para la evaluación y progreso de los alumnos y funciones básicas para al menos la autoría de test y evaluaciones.

10. Debe existir funciones para la gestión de cursos.

11. Debe existir funciones para la gestión de contenidos.

2.2 Alcance

La propuesta va dirigida al grupo de investigadores que está utilizando el paradigma DSM para el desarrollo de sus investigaciones. Es aplicable a todos los proyectos productivos que deseen comenzar el desarrollo de un Sistema de Gestión del Aprendizaje.

2.3 Estructura general del procedimiento

El procedimiento que se propone consta de 2 etapas claves. Cada una de estas etapas encierra una serie de pasos complementarios. A continuación se muestran las etapas del procedimiento para realizar el proceso de definición de Lenguaje de Dominio Específico y además una representación gráfica de la propuesta:

1. **Definición del Modelo de Dominio Específico**
   - 1.1 Definición de los conceptos principales de los Sistemas de Gestión del Aprendizaje
2. **Creación de un Lenguaje de Dominio Específico**
   - 2.1 Proceso para modelar la sintaxis abstracta
     - 2.1.1 Creación del metamodelo para Sistemas de Gestión del Aprendizaje
   - 2.2 Proceso para definir la sintaxis concreta
     - 2.2.1 Editor para el Lenguaje de Dominio Específico
     - 2.2.2 Proceso para definir la semántica
     - 2.3.1 Generación de código o transformación de DSL- código

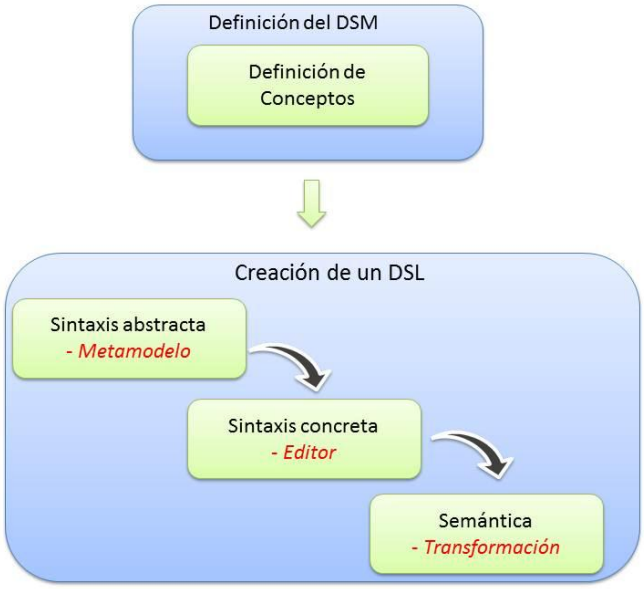

Figura 5: Propuesta gráfica general del procedimiento

2.4 Definición del Modelo de Dominio Específico

En la práctica, cada solución DSM diseña un lenguaje y un generador de código dentro de una organización en particular. Se enfocan en dominios pequeños, porque son más fáciles de definir y permiten mejores posibilidades para su automatización. El desafío de los desarrolladores y empresas se centra en la definición de los lenguajes de modelado, la creación de los generadores de código y la implementación de framework específicos del dominio; elementos claves de una solución DSM.

En la actualidad, existe una familia importante de herramientas para crear soluciones en DSM que permiten diseñar un lenguaje específico de dominio, para luego construir herramientas de modelado y generadores de código para ese lenguaje. Algunas de estas fueron mencionadas en el capítulo anterior, ellas son: Microsoft con las "Factorías de Software". Una Factoría de Software es una línea de producción de software que combina herramientas de desarrollo extensible, tales como Visual Studio Team System, con contenido empaquetado como DSLs, patrones y frameworks, basados en recetas para construir tipos específicos de aplicaciones. Metacase con su producto MetaEdit+ y Eclipse con su proyecto llamado Eclipse Modeling Project; proyecto compuesto por otros subproyectos código abierto para el desarrollo de lenguajes de modelado y generación de código (26).

El DSM propone comenzar el desarrollo con una definición formal de los conceptos más utilizados en los LMS y la representación de alto nivel mediante modelos, los cuales serán el artefacto principal de la especificación del software.

2.4.1 Definición de los conceptos principales de los Sistemas de Gestión del Aprendizaje

En esta etapa se deben estudiar las características de los LMS, los elementos y conceptos principales. Para ello se debe trabajar con cada una de las plataformas (Sakai, Atutor, Claroline, Moodle, Dotlrn, entre otras) para poder clasificar sus módulos por ejemplo: *Documentos, Ayuda, Gestión de curso, Cursos, Usuarios, Sistemas de evaluación, Foros, Trabajos, Administración de la plataforma*. Estos módulos se comparan basándose en la experiencia y el conocimiento que haya proporcionado el haber utilizado los mismos en modo profesor y finalmente se genera una aproximación de módulos comunes.

La diagramación se pudiera llevar a cabo sin ayuda de medios digitales o mediante alguna herramienta que provea facilidades de representación gráfica como los mapas mentales o de conocimiento. Se recomiendan las herramientas FreeMind y CMaptools ya que son de libre distribución.

- **FreeMind**: Herramienta para organizar y estructurar las ideas, los conceptos, su relación entre ellos y su evolución. Puede ser utilizada en cualquier área del ámbito educativo, como mecanismo o forma de plasmar tormentas de ideas de todo tipo para su posterior reutilización. Es un software ligero, sencillo de instalar, configurar y utilizar está disponible en Windows y GNU/Linux.
- **CMapTools**: Es un software multiplataforma para crear mapas conceptuales, por medio de unas aplicaciones escritas en Java. Permite tanto el trabajo local individual, como en red, ya sea local, o en internet, con lo que facilita el trabajo en grupo o colaborativo. Posibilita la navegación por los mapas realizados, lo que los convierte en interactivos. Se pueden enlazar e indexar prácticamente todo tipo de archivos, con la posibilidad de añadir información contextual a cada uno de los conceptos o nodos del mapa (27).

Después de haber analizado algunas de las características de estas herramientas se puede utilizar cualquiera de las dos u otras que usted considere propicias para la diagramación, aunque es más aconsejable CMapTools ya que ofrece más facilidad para el trabajo colaborativo Web y posee diversidad de formatos para la exportación del mapa.

Una vez seleccionada la herramienta de diagramación se crea el mapa de conocimiento por cada sistema de gestión del aprendizaje con el fin de conocer la estructura genérica[17] de cada LMS y se lleva a cabo un proceso de comparación entre cada uno de los módulos[18] de los LMS generando una aproximación de los posibles módulos compatibles entre los diferentes LMS. Estos módulos modelados serán utilizados en la creación de Lenguaje de Dominio Específico.

2.5 Creación de un Lenguaje de Dominio Específico

Un DSL se diseña para suplir unas necesidades muy específicas dentro de un dominio de aplicación concreto, contrastando así con los lenguajes de propósito general.

El desarrollo de un lenguaje es considerado una tarea difícil, ya que requiere un profundo conocimiento del dominio y mucha experiencia en el tema. Pocas personas tienen ambas habilidades, y debido a eso, la decisión de desarrollar un DSL es pospuesta muchas veces indefinidamente.

El objetivo de los mismos como se había mencionado en el capítulo anterior es la creación de lenguajes especializados para poder, de forma más sencilla, modelar artefactos de software en el ámbito para el que fue prescrito el DSL, y sea capaz de generar código de estos modelos (gráficos o texto).

En la actualidad se están dedicando importantes esfuerzos al desarrollo de herramientas que permitan la definición de lenguajes de modelado o DSL, estas herramientas incorporan mecanismos para: crear metamodelos; asociar una notación cualquiera a los elementos de un metamodelo; expresar transformaciones para la generación de modelos o código a partir de un modelo conforme a cierto metamodelo, y realizar consultas sobre los metamodelos creados (9).

La definición del DSL consta de 2 partes:
Modelo de dominio: Es un modelo de los conceptos descriptos por el lenguaje. EL modelo de dominio define la sintaxis abstracta del DSL.
Notación: Los conceptos del dominio son mapeados a figuras con el fin de poder representarlos en diagramas. Esta notación define la sintaxis concreta del DSL (28).

Para definir un nuevo lenguaje es necesario definir su sintaxis abstracta, su sintaxis concreta y su semántica.

[17] Genérica: Que es común o se refiere a un conjunto de elementos del mismo género.
[18] Módulo: Es una porción de un software de computadoras. Un módulo realiza comúnmente una o varias tareas que necesita el programa para cumplir su función u objetivo.

Sintaxis abstracta

La sintaxis abstracta de un lenguaje describe su vocabulario, es decir, los conceptos provistos por el lenguaje y como estos conceptos se pueden combinar para crear modelos. Es importante enfatizar en este punto que la sintaxis abstracta del lenguaje es independiente de su sintaxis concreta y de la semántica asociada. La sintaxis abstracta define la forma y la estructura de los conceptos del lenguaje, sin considerar su presentación o su significado.

Sintaxis concreta

Todos los lenguajes proveen una notación que facilita la representación y la construcción de modelos o programas escritos en dicho lenguaje. Esta notación también se la conoce como sintaxis concreta. Las clases de sintaxis concretas se pueden dividir en dos tipos principales: sintaxis textual y sintaxis visual. Una sintaxis textual permite escribir modelos o programas de manera textual (29). Puede tener varias formas, pero típicamente consiste de una sección de declaraciones, donde se declaran los objetos y variables que van a estar disponibles dentro del programa y de un conjunto de sentencias asociadas. El código Java siguiente muestra una sintaxis textual donde se declara una clase con una variable local y los correspondientes métodos para accederla.

```
public abstract class Person {

private String name;

public String getName(){

return name;

}

public String setName(String newName){

name:= newName;

}

}
```

La ventaja de las sintaxis textuales es la posibilidad de modelar o escribir expresiones complejas. Sin embargo, más allá de cierta cantidad de líneas se vuelve muy difícil de comprender y manejar.

Una sintaxis visual presenta un modelo o programa en forma de diagramas. Consiste de un número de íconos gráficos que representan vistas de un modelo subyacente. Se recomienda la utilización del UML para la representación de la sintaxis visual ya que es utilizado para modelar sistemas complejos, tanto en el diseño de los sistemas software como para la arquitectura hardware. UML es además un método formal de modelado y aporta las siguientes ventajas:

- ✓ Mayor rigor en la especificación.
- ✓ Permite realizar una verificación y validación del modelo realizado.
- ✓ Se pueden automatizar determinados procesos y permite generar código a partir de los modelos y a la inversa (a partir del código fuente generar los modelos). Esto permite que el modelo y el código estén actualizados, con lo que siempre se puede mantener la visión en el diseño, de más alto nivel, de la estructura de un proyecto (30).

El principal beneficio de una sintaxis visual es la habilidad de expresar mucha información de una manera intuitiva y entendible. La desventaja es obvia: solamente ciertos detalles pueden ser expresados en un gráfico ya que si se quisiera expresar todo, el mismo se volvería demasiado complejo e incomprensible.

En la práctica se utiliza una mezcla de ambas, ya que se gana el beneficio de las dos. Así, los lenguajes frecuentemente usan las notaciones visuales para representar las vistas de alto nivel de un modelo, mientras que las sintaxis textuales se usan para capturar los detalles.

Semántica

La sintaxis abstracta no define que significan los conceptos dentro de un lenguaje. Por lo tanto se precisa información adicional para capturar la semántica de dicho lenguaje. La semántica de un lenguaje describe el significado de los conceptos del mismo. Por esa razón cuando se utiliza un lenguaje, es necesario asignar un significado a sus conceptos para lograr entender cómo usarlo (31).

2.5.1 Proceso para modelar la sintaxis abstracta

El primer paso en el diseño de un lenguaje de modelado es construir su sintaxis abstracta. Este paso es tan importante como construir un modelo del sistema en un diseño orientado a objetos. La sintaxis abstracta describe los conceptos y las relaciones existentes entre ellos. Estos conceptos, relaciones y reglas identificados en este paso proveen el vocabulario y la gramática para construir los modelos usando este lenguaje. Así, esta sintaxis constituye la base sobre la cual los otros artefactos del lenguaje se van a definir.

La identificación de conceptos y el modelado de los mismos son pasos importantes en el desarrollo del modelo de una sintaxis abstracta. Estos pasos se describen a continuación:

- **La identificación de los conceptos**

El primer paso en el modelado de la sintaxis abstracta de un lenguaje es utilizar cualquier información disponible que ayude a identificar los conceptos del lenguaje, y cualquier regla obvia que posibilite validar o invalidar esos modelos.

La identificación de los conceptos relevantes para el lenguaje depende en gran medida de las ideas creativas y de la experiencia en el dominio. Siempre que sea posible se debe consultar a personas con más experiencia en el tema, por ejemplo los expertos en el dominio, arquitectos, y líderes de los equipos de desarrollo. Sus opiniones y puntos de vista son la principal fuente para la creación de la solución.

Fuentes típicas para encontrar conceptos candidatos:

- ✓ La arquitectura: Una descripción de la arquitectura es una buena fuente de información ya que aquí se opera con conceptos del dominio. Usualmente es la parte más estable y revela abstracciones importantes acerca de los elementos de la aplicación y su comportamiento
- ✓ Productos existentes, aplicaciones y correspondientes manuales: Estos capturan la estructura, el comportamiento y la semántica, pero desafortunadamente hacer un análisis completo no siempre es práctico, ya que podría consumir demasiado tiempo examinar exhaustivamente todos esos productos existentes. Sería más conveniente entonces seleccionar un conjunto de aplicaciones representativo para inspeccionarlas con mayor detalle. Naturalmente en ese conjunto seleccionado deberían encontrarse las que tengan una relación más estrecha con la que se quiere desarrollar.
- ✓ Especificaciones disponibles: Analizando descripciones existentes de las características, aplicaciones o productos se puede comprender la estructura del dominio y traducirla a un esquema conceptual. Documentos de requisitos son especialmente buenos para alcanzar el nivel de abstracción ya que usualmente se enfocan en el dominio del problema más que en su implementación. Los requisitos también involucran la terminología del cliente, ampliando así el uso del DSL: Los clientes podrán entonces leer y chequear sus modelos. Las especificaciones no necesitan ser formales o estar basadas en algún lenguaje de especificación. De hecho, los modelos realizados sin ninguna restricción permite a los modeladores capturar el dominio de una manera más efectiva.

A veces, estas fuentes de conceptos candidatos para el lenguaje no están disponibles, ya sea porque el dominio es nuevo y no hay experiencias pasadas disponibles o porque no existen especificaciones. En este caso, el conocimiento del dominio está disperso en la organización y se encuentra solo en la mente de los individuos. Lo ideal entonces es crear ejemplos concretos de formas alternativas para especificar el problema. Ejemplos múltiples con sus respectivos prototipos ayudarán a identificar abstracciones y pueden ser usados luego para probar la solución.

Algunas características que deben tener los conceptos del dominio del problema para poder ser seleccionados a la hora de definir el lenguaje son:

- ✓ Frecuentemente ya tienen establecida una semántica: el lenguaje se puede definir entonces en base a estas definiciones existentes. Por ejemplo, en el dominio de las aplicaciones para dispositivos móviles, se conoce el concepto de soft key: son dos o más teclas cuya función cambia basado en el contexto de la aplicación. De la misma manera, en todos los dominios existen conceptos conocidos inherentes a ellos. Si la semántica de un concepto en particular no puede ser definida, lo más probable es que no sea un concepto relevante para el lenguaje.
- ✓ Son conocidos y usados: se usan frecuentemente mientras se habla entre los desarrolladores. Esto hace que no sea necesario introducir nuevos conceptos en el lenguaje, ya que se construye con esos existentes, usados y aceptados. Incluso la gente se siente más cómoda con esos conceptos ya que no tienen que invertir tiempo en aprenderlos.
- ✓ Establecen un mapeo natural con el problema que se quiere resolver con el lenguaje: hace que la creación del modelo sea más fácil y hace posible operaciones como la reutilización de elementos del modelo en el dominio. Una estrecha relación con el dominio hace que los modelos sean fáciles de leer, entender, recordar, chequear y comunicar a los demás integrantes del equipo de desarrollo.

Casos de uso

Una técnica útil para identificar los conceptos del lenguaje de modelado es considerar diferentes casos de uso asociados al uso del lenguaje. Por ejemplo, una colección de operaciones que deberían ser usadas cuando se interactúa con el metamodelo. Algunos casos de uso típicos para esto incluyen crear y eliminar los elementos del modelo, pero también deberían incluir operaciones ricas semánticamente como transformar o ejecutar el modelo. Las

descripciones detalladas que resultan sobre el análisis de los casos de uso son una gran fuente de conceptos para el lenguaje (26).

Al modelar un lenguaje deberían ser modelados solamente los conceptos que permiten describir alguna variación y no modelar algo que siempre permanece igual, si eso fuera así debería proveerse mediante librerías[19] o componentes o ser producido directamente por el generador.

Los conceptos que pueden ser identificados a partir de una combinación de otros existentes deben ser excluidos ya que se debe mantener el lenguaje lo más acotado posible, pues se vuelve más fácil de aprender y usar. Además se debe tener en cuenta que durante este proceso se distinga entre la sintaxis concreta del lenguaje y la sintaxis abstracta. Es muy común que la estructura de la sintaxis abstracta refleje su sintaxis concreta. Por ejemplo, considerando un lenguaje que provea múltiples vistas superpuestas de un pequeño número de conceptos del modelo. En este caso, la sintaxis abstracta debería modelar solo los conceptos del modelo y no su sintaxis concreta.

En general, los mejores modelos de sintaxis abstracta son los más simples. Cualquier complejidad debida a una representación en diagramas debe ser deferida a los modelos de sintaxis concreta.

- **Modelado de los conceptos**

Tan pronto como se hayan identificado las partes relevantes del lenguaje de modelado, se deben formalizar. La mejor forma de hacerlo es creando un metamodelo.

Creación del metamodelo para Sistemas de Gestión del Aprendizaje

Usando un lenguaje de modelado, se pueden crear modelos; un modelo especifica que elementos pueden existir en un sistema. Por ejemplo si se define la clase Persona en un modelo, se pueden tener instancias de Persona como *Juan*, *Pedro*. Por otro lado, la definición de un lenguaje de modelado especifica que elementos pueden existir en un modelo. Por ejemplo, el lenguaje UML especifica que dentro de un modelo se pueden usar los conceptos *Clase, Estado, Paquete*. Debido a esta similitud, se puede describir un lenguaje por medio de un modelo, usualmente llamado metamodelo. El metamodelo de un lenguaje describe que elementos pueden ser usados en el lenguaje.

[19] Librería: Conjunto de subprogramas utilizados para desarrollar software. Las librerías contienen código y datos, proporcionando servicios a programas independientes.

Como se mencionó en el capítulo 1 un metamodelo es un modelo, generalmente de la forma de un diagrama de clases, que describe los modelos que son parte del lenguaje. Metamodelar simplemente significa modelar el lenguaje: mapear los conceptos del dominio a los elementos del lenguaje, como los objetos, las propiedades de esos objetos, y las relaciones que existan entre ellos. El proceso de metamodelar debe ser soportado por un lenguaje de metamodelado que es a su vez, un lenguaje específico de dominio para especificar lenguajes de modelado. El lenguaje estándar para metamodelar es MOF definido por la OMG (26), aunque se pueden utilizar otros lenguajes de metamodelado dependiendo de la herramienta que se seleccione.

Apoyándose en la arquitectura de las 4 capas de modelado definida por OMG (18) se puede decir que análogamente a como ocurre con las capas M0 (objeto) y M1 (modelo), los elementos del nivel M1 son a su vez instancias del nivel M2 (metamodelo).

Para un mejor entendimiento de cómo definir correctamente el metamodelo la Figura 6 muestra una parte del metamodelo UML. En este nivel aparecen conceptos tales como *Clase, Atributo o Asociación.*

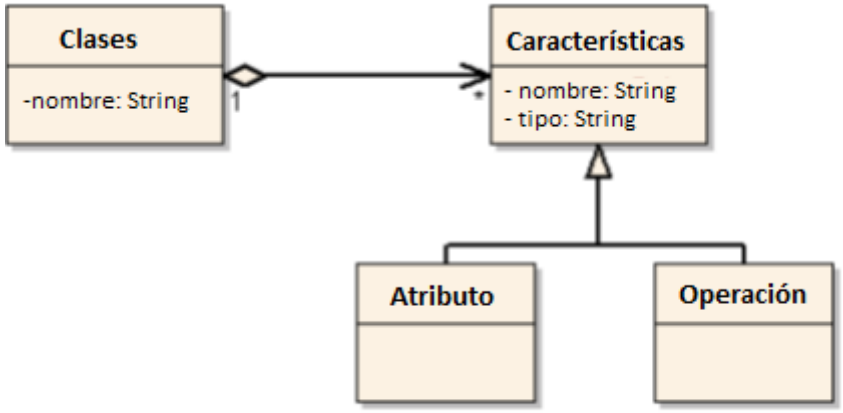

Figura 6: Parte del metamodelo UML (26)

Como se muestra a continuación, la entidad *Cliente* será una instancia de la metaclase *Clases* del metamodelo UML. Sus *atributos, nombre y dirección* serán instancias de la metaclase *Atributo*.

La Figura 7 muestra la relación entre los elementos del nivel M1 con los elementos del nivel M2.

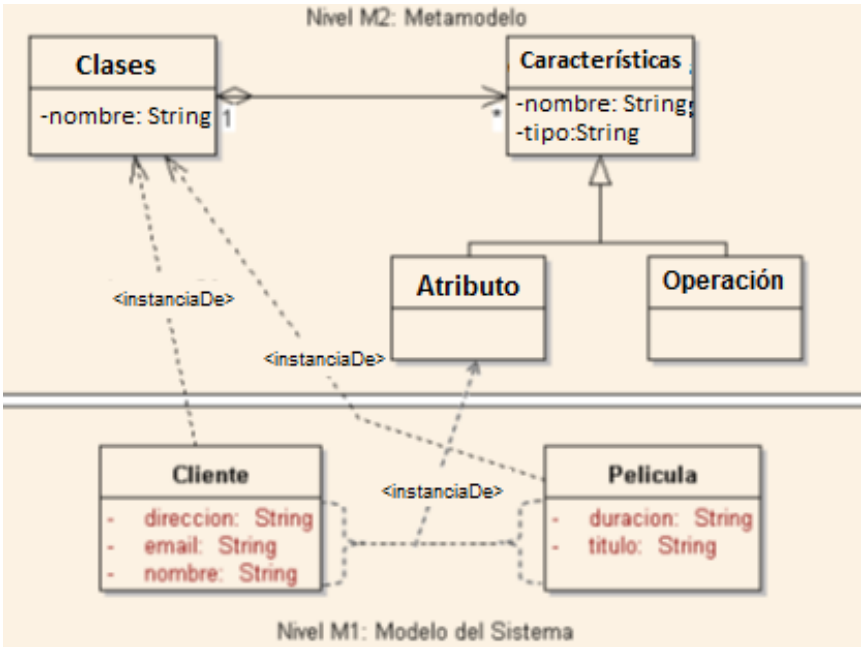

Figura 7: Relación entre el nivel M1 y el nivel M2 (26)

Utilizando la herramienta DSL Tools el metamodelo se construye de forma visual mediante un editor gráfico, el DomainModel.dsldm (Modelo de Dominio según Microsoft) (22). Mientras que MetaEdit+ como utiliza el lenguaje GOPRR (Grafo Objeto Propiedad Relación Rol), se entremezclan los conceptos de sintaxis abstracta y del metamodelo. En cuanto a Eclipse es necesario tener instalado el plugin de EMF (del inglés Eclipse Modeling Framework) para la utilización de Ecore, su lenguaje de metamodelado, el cual se utiliza para definir la sintaxis abstracta (4).

Como se había mencionado anteriormente la generación de código es posible a partir de la especificación de un modelo. De esta manera, permite que el desarrollador se concentre en el modelo y delegue en el framework los detalles de la implementación.

2.5.2 Proceso para definir la sintaxis concreta

La sintaxis abstracta describe el vocabulario y la gramática del lenguaje, pero no define como se mostrará ese lenguaje al usuario. Estas vistas son definidas por la sintaxis concreta, y pueden ser en forma de diagrama o en forma de texto. Muchos de los lenguajes de modelado usan sintaxis concreta en forma de diagramas, como máquinas de estado o los diagramas de clases; aunque frecuentemente se ven limitados por el tamaño del diagrama.

El proceso de definir la sintaxis concreta tiene dos pasos: el primero es interpretar la sintaxis y asegurar que sea válida. El segundo es usar la sintaxis concreta para instanciar la sintaxis

abstracta. Estos pasos son igualmente aplicables si la sintaxis es en forma de texto o de diagrama, aunque existe una diferencia importante en la manera en que las sintaxis concretas se construyen por el usuario final.

De la misma manera que la sintaxis abstracta juega un rol central en la descripción del lenguaje, la sintaxis concreta es crucial al diseñarlo. Además se puede definir como un elemento separado en la descripción del lenguaje.

El rol de la sintaxis concreta es representar un programa a los sentidos del usuario. Usualmente se hace por medio de una herramienta, siendo la más importante de ellas el editor, el cual se utiliza para crear o cambiar el programa. El tipo de editor usado influye en el modo en que se piensa en la sintaxis concreta y su mapeo a la sintaxis abstracta. También hay que considerar si el lenguaje definido es textual o gráfico.

Editor para el Lenguaje de Dominio Específico

Los editores son el principal medio para crear, mostrar y editar la información de los modelos. Generalmente cada herramienta de metamodelado incluye sus propios editores como por ejemplo Metaedit+, que presenta editores de aspecto visual y editores de los modelos como el *editor de diagrama*, el *editor de matriz*, y el *editor de tabla*. DSL Tools por su parte expresa la sintaxis concreta mediante un fichero XML, el Designer.dsldd (Diseñador según Microsoft) que contiene los elementos o figuras y los conectores que aparecen en el lenguaje, así como la relación con los conceptos del modelo de dominio (22). En caso de que sea utilizado Eclipse como herramienta de metamodelado se debe emplear GMF Tooling (del inglés Graphical Modeling Framework Tooling) que hace parte del proyecto GMP[20] (del inglés Graphical Modeling Project), este primero es un framework de código abierto, que permite construir editores gráficos (32). Cada editor es conforme con el metamodelo elegido y muestra los modelos diseñados (9).

2.5.3 Proceso para definir la semántica

Una semántica tiene como objetivo esencial comunicar el significado de los modelos o programas, entre los miembros del equipo de desarrollo de un sistema. La semántica tiene un rol central ya que define las capacidades del lenguaje como su ejecución, análisis y transformación.

[20] GMP: Es un proyecto Eclipse de modelado gráfico que proporciona un conjunto de componentes de generadores y de las infraestructuras en tiempo de ejecución para el desarrollo de editores gráfico.

Una estrategia es describir la semántica usando metamodelos, lo cual brinda algunos beneficios importantes. Primero, la semántica del lenguaje está completamente integrada a la definición del mismo. Esto significa que está incluida dentro de los otros elementos del lenguaje, como la sintaxis concreta y sintaxis abstracta. Segundo, como se usa el mismo lenguaje de metamodelado para las definiciones de todos los lenguajes, las definiciones semánticas se convierten en aserciones reusables que pueden ser integradas y extendidas con relativa facilidad. Finalmente y lo más importante es que las definiciones semánticas son independientes de plataforma, se pueden intercambiar y, si son entendidas por la herramienta que las importa, las puede usar para conducir la forma en que esta interactúa con el lenguaje.

Es importante notar que un metamodelo de la semántica es bastante diferente del modelo de sintaxis abstracta de un lenguaje, el cual define la estructura del mismo. Sin embargo, un modelo de sintaxis abstracta es prerrequisito para la definición de la semántica, ya que la semántica agrega una capa al significado de los conceptos definidos por la sintaxis abstracta. La semántica en este sentido debería ser distinguida de la semántica abstracta, la cual expresa las reglas que indican si una expresión del lenguaje está bien formada. Las reglas de la semántica estática son usadas por las herramientas como verificadores de tipos.

Generación de código o transformación de DSL- código (modelo a código)

La transformación de modelos juega un papel clave en el desarrollo dirigido por modelos, puesto que serán un conjunto de transformaciones las que, partiendo de un conjunto de modelos que especifican un sistema, permitan conseguir el software ejecutable sobre una plataforma concreta (33).

Por último, una vez definidos todos los modelos y relacionados a través del archivo de mapeo, se pasa a generar el código de los mismos. Para ello es necesario tener en cuenta los lenguajes de transformación que fueron mencionados anteriormente en el capítulo 1, haciendo énfasis en los de tipo Transformación Modelo a Texto (M2T). Uno de ellos fue Java Emitter Template (JET) que está enfocado en la traducción de modelos a su representación textual, la generación de código con JET consta de dos pasos: traducción y generación.

En el primero se traduce el template[21] a una representación en código Java; y en el segundo se realiza la generación del archivo de texto utilizando las clases creadas en el template (26).

[21] Template: Un template de JET es un archivo cuya extensión termina en "jet". Por convención de EMF, la primer parte de la extensión del archivo corresponde al tipo del resultado de la traducción.

La herramienta MOFScript (34) permite la transformación de cualquier modelo MOF a texto. Por ejemplo, permite la generación de código Java, EJB, JSP, C#, SQL Scripts, HTML o documentación a partir de los modelos. La herramienta como está desarrollada como un plugin de Eclipse, soporta el parseo, chequeo y ejecución de scripts escritos en MOFScript. La herramienta asume que la extensión ".m2t" refiere a archivos de MOFScript. Estos archivos pueden ser ubicados dentro del Eclipse para que luego puedan ser compilados y ejecutados.

Metaedit+ proporciona un lenguaje propio para expresar cómo se realiza la generación de código, informes de consistencia o cualquier otro tipo de salida a partir del modelo. Con el término *informe* se hace referencia a aquellos programas escritos en el lenguaje de informes propietario de Metaedit+, y que sirve para generar una salida en base a un modelo. Con todo ello se permite construir un modelo gráfico a partir del cual se pueden crear instancias de las que se puede generar distintos ficheros de código (22). DSL Tools introduce el concepto de las plantillas de texto para generar código. Las plantillas de texto serán la guía que defina el proceso de generación de código a partir de un modelo de entrada. Se pueden considerar como las unidades de generación de código o documentación, de forma que, generalmente la ejecución de una plantilla de texto genera una unidad de documentación o código. Mientras que con la herramienta Eclipse se integran varias tecnologías como son JET (35) y MOFScript (36), ambas emplean el mismo principio, la creación de reglas de transformación basándose en un metamodelo. Estas reglas serán aplicadas al modelo, para generar código en el lenguaje deseado.

Conclusiones parciales

En el presente capítulo se realizó un estudio de las fases que conforman el procedimiento. Se llevó a cabo la propuesta del procedimiento para definir el Modelo de Dominio Específico y Lenguaje de Dominio Específico para los Sistemas de Gestión del Aprendizaje. Fueron definidas 2 etapas, además de un conjunto de pasos complementarios para la correcta elaboración del procedimiento con el propósito de apoyar a los nuevos investigadores en el desarrollo de este proceso.

Capítulo 3: Validación del Procedimiento

Introducción

Para validar el modelo propuesto, es necesario contar con el criterio de personas expertas en el tema que posean el conocimiento necesario para saber si lo investigado, está realmente cercano a alcanzar la calidad que se espera obtener en los resultados. Para realizarle la validación a la propuesta del capítulo anterior, se tomó como criterio las opiniones de un conjunto de especialistas. La validación se realizará mediante el método Delphi, el cual a través del criterio de los expertos se puede determinar el nivel de aceptación que tendrá el proceso de definición de Modelo de Dominio Específico y Lenguaje de Dominio Específico.

3.1 Métodos de científicos existentes

Los métodos científicos son un proceso mediante el cual los científicos construyen y estudian una representación confiable, consistente y no arbitraria de situaciones reales que representan un problema que se quiere resolver. Ellos disponen de diferentes metodologías y técnicas de investigación para la construcción de ciencia y conocimiento. Estas se pueden clasificar en cuantitativas y cualitativas. Cada una de ellas dispone de diferentes herramientas que, a su vez, tienen aplicaciones en diferentes circunstancias, entornos, sectores, casuísticas.

Los métodos cuantitativos, se precisan como aquellos que permiten la obtención de datos e informaciones sobre los fenómenos y procesos de la realidad, donde utilizan muestras grandes y representativas de sujetos, pero que manifiestan una relación distante (vertical) entre el investigador, los sujetos y objetos investigados, así como que en el análisis de datos obtenidos utilizan la estadística, sin una perspectiva valorativa ni reflexión crítica. Se usan principalmente en situaciones en las que existen o se pueden conseguir datos previos confiables sobre un fenómeno de análisis.

Mientras que los métodos cualitativos, son considerados como aquellos que permiten profundizar en la comprensión e interpretación de los procesos captando la realidad en toda su riqueza, totalidad y cotidianidad, desde los criterios de una muestra reducida de sujetos, seleccionados por métodos generalmente no probabilísticos.

Existen varias técnicas de investigación cualitativa donde participan expertos como son las entrevistas en profundidad, las entrevistas estructuradas, entre otras (37). Los métodos de investigación orientados a la prospectiva, se pueden agrupar en tres tipos: Métodos de expertos (basado en las opiniones de conocedores del problema que se quiere analizar);

métodos extrapolativos (basado en datos históricos que se pueden extrapolar al futuro) y métodos de correlación (basados en la identificación de factores relevantes y su evolución hacia el futuro). El método *Delphi* es un método de expertos definido como "un proceso sistemático e iterativo encaminado a la obtención de las opiniones, y si es posible el consenso, de un grupo de expertos" (37).

Como el resultado que se desea obtener del proceso de validación del procedimiento, es a partir de la búsqueda progresiva del consenso en los criterios y valoraciones expuestas por los expertos o especialistas consultados; de ahí que, además de la aplicación de los métodos empíricos, especialmente el criterio de experto, se sostiene la necesidad de seleccionar y emplear un método que garantice o se acerque a tal propósito. Por tanto se utilizará el método Delphi ya que está considerado como uno de los métodos de pronósticos más confiables, para establecer un cuadro de la evolución estadística de las opiniones de expertos en un tema, fundamentadas en su análisis lógico y experiencia intuitiva (38).

3.2 Métodos Delphi

El método Delphi[22] se basa en el principio de la inteligencia colectiva, que trata de lograr un consenso de opiniones expresadas de forma individual por un grupo de personas seleccionadas cuidadosamente como expertos calificados en torno al tema, por medio de la iteración sucesiva de un cuestionario retroalimentado de los resultados promedio de la ronda anterior, con la aplicación de cálculos estadísticos (37). En los cuestionarios se analizan aspectos sobre el nombre, la definición y las actividades que componen la intervención. Con este se pretende extraer y maximizar las ventajas mediante el criterio emitido por un conjunto de especialistas en el tema. La calidad de los resultados depende, sobre todo del cuidado que se tenga en la elaboración del cuestionario y en la elección de los especialistas consultados (38).

En la siguiente imagen muestra el modo de funcionamiento de este método:

[22] Delphi*:* El nombre de este método de proviene del oráculo de *Delphos*, que se encontraba en la antigua Grecia, al que se acudía para hacer preguntas al dios a través de una sacerdotisa. El oráculo de *Delphos* poseía gran reputación por la certeza de sus predicciones.

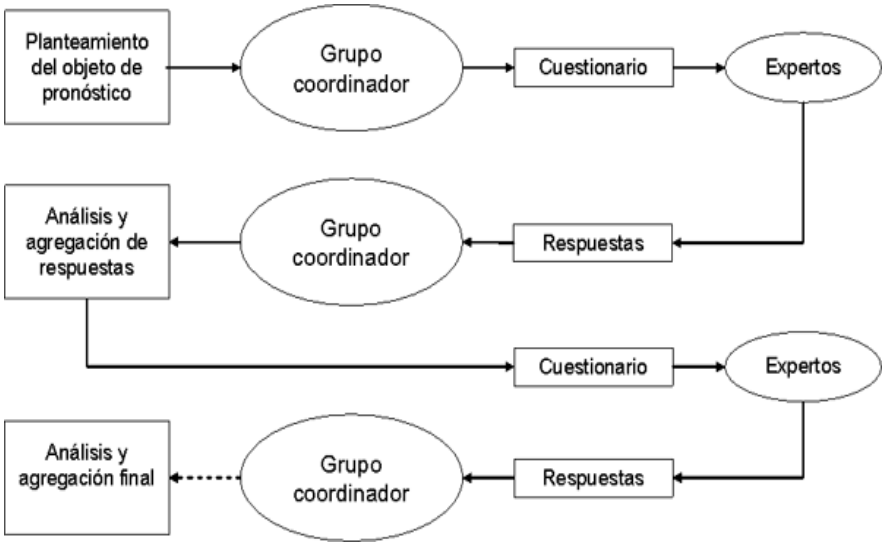

Figura 8: Modo de Funcionamiento del Método Delphi (38)

Como técnica empleada, la inteligencia colectiva es el principio por el cual se rige este método. Se logra además un grado de conformidad con las opiniones expresadas individualmente por las personas que se seleccionaron, clasificándose estos como expertos en el tema.

El método presenta 4 características principales:

- **Anonimato**: Ningún experto conoce la identidad de los otros que componen el grupo de debate.
- **Iteración y retroalimentación controlada**: La iteración se consigue al presentar varias veces el mismo cuestionario. Como se van presentando los resultados obtenidos de los cuestionarios anteriores, se consigue que los expertos vayan conociendo los distintos puntos de vista y puedan ir modificando su opinión si los argumentos presentados les parecen más apropiados que los suyos.
- **Respuesta del grupo en forma estadística**: La información que se presenta a los expertos no es solo el punto de vista de la mayoría, sino que se presentan todas las opiniones indicando el grado de acuerdo que se ha obtenido.
- **Heterogeneidad**: Pueden participar expertos de determinadas ramas sobre las mismas bases (39).

Para la realización de este método se elegirán un conjunto de expertos que serán los encargados de llevar a cabo la validación de la propuesta. Durante este período ningún experto conocerá la identidad del resto de sus compañeros, permitiendo así que pueda defender sus criterios u opiniones aun siendo estos erróneos. Además, se evita que un miembro sea

influenciado por la opinión de la mayoría o por la reputación de uno de sus compañeros. Un aspecto importante es la correcta selección de los expertos, puesto que ofrece no solo la certeza de un resultado correcto sino también un alto grado de confiabilidad y credibilidad.

3.3 Aplicación del método

Para la aplicación del método se siguieron las etapas que se mencionan a continuación:

- Elección de los expertos.
- Elaboración de cuestionarios para la validación de la propuesta.
- Determinación de la concordancia de los expertos.
- Desarrollo práctico de los resultados.

3.3.1 Elección de los expertos

Se dice que un experto es aquella persona, grupo de personas u organización que posee conocimientos amplios o aptitudes en un área particular del conocimiento, capaces de valorar, formular conclusiones objetivas y dar recomendaciones acerca del problema mostrado (40).

Los expertos se seleccionaron teniendo en cuenta que cumplieran con los criterios siguientes:

- Graduado de Nivel Superior.
- Un año de experiencia como mínimo trabajando temas relacionados al Desarrollo de Software Dirigido por Modelos.
- Vinculación al desarrollo de productos informáticos.
- Dominio y conocimiento sobre el tema.
- Capacidad de análisis y pensamiento lógico.
- Disposición a participar en la entrevista.
- Honestidad.

Para poner en práctica el método se seleccionaron 9 expertos dentro de la Universidad de las Ciencias Informáticas a los cuales se les presentó la propuesta de participación , de ellos 8 respondieron afirmativamente brindando su colaboración en la investigación y formando parte del equipo de validación.

Para la selección de los expertos finales se hace obligatorio conocer el grado de conocimiento del experto en cuestión, la misma se realiza con la ayuda del Coeficiente de Competencia. Este coeficiente se determina mediante la fórmula: K= ½ (Kc + Ka), donde:

Kc: es el Coeficiente de Conocimientos del experto sobre el tema.

Ka: es el Coeficiente de Argumentación del experto sobre el tema.

Kc se obtiene de la siguiente tabla que recoge una autoevaluación del posible experto.

1	2	3	4	5	6	7	8	9	10
								x	

Tabla 1: Autovaloración del Coeficiente de Conocimientos (Kc)

En esta tabla el experto debe marcar según el grado de conocimiento que tenga sobre el tema que se ha puesto a su consideración en una escala del 1 al 10, luego para ajustarla a la teoría de las probabilidades se multiplica por 0.1. De esta forma, si selecciona el 9 en la **Tabla 1**, al multiplicarlo por 0.1 se considerada en la tabla, **Kc** = 0.9. Una evaluación de 1 quiere decir que el experto no posee conocimiento alguno sobre el tema y una evaluación de 10 significa que domina el tema perfectamente. (41)

Para calcular el coeficiente de argumentación (Ka), el experto debe marcar según su consideración cuáles fueron las fuentes de las que obtuvo sus conocimientos que le permite argumentar su evaluación del nivel de conocimiento que especificó en la tabla anterior. Las marcas de los expertos se traducen a puntos según la siguiente tabla patrón:

No	Fuentes de argumentación	Grado de influencia		
		Alto	Medio	Bajo
1	Análisis teóricos realizado por Ud.	0.3	0.2	0.1
2	Su experiencia obtenida	0.5	0.4	0.2
3	Trabajos de autores nacionales.	0.05	0.05	0.05
4	Trabajos de autores extranjeros.	0.05	0.05	0.05
5	Su propio conocimiento del tema.	0.05	0.05	0.05
6	Su intuición.	0.05	0.05	0.05
	Total	1.0	0.8	0.5

Tabla 2: Escala de puntos para la determinación del coeficiente de argumentación (41)

Con estos elementos es suficiente para obtener el Coeficiente de Competencia (K) a través de la siguiente fórmula: K= ½ (Kc + Ka)

El resultado obtenido se interpreta en una escala que se plantea a continuación según el método Delphi:

Si 0.8 < **K** < 1.0, el Coeficiente de Competencia es alto y confiable.

Si 0.5 < ***K*** < 0.8, el Coeficiente de Competencia es medio.

Si **K** < 0.5 el Coeficiente de Competencia es bajo. (41)

Los expertos seleccionados para formar parte del grupo de validación fueron aquellos cuyos resultados del coeficiente de competencia fueron Alto y Medio. De los 8 expertos iniciales a los que se les aplicó la Encuesta de Autovaloración, solo 7 resultaron seleccionados para continuar con la ejecución del método, los resultados se muestran a continuación:

Expertos	Coeficiente de conocimiento (Kc)	Coeficiente de argumentación (Ka)	Coeficiente de competencia (K)	Nivel
E1	0.9	0.9	0.9	ALTO
E2	0.5	0.8	0.65	MEDIO
E3	0.5	0.6	0.55	MEDIO
E4	0.8	0.6	0.7	MEDIO
E5	0.6	0.8	0.7	MEDIO
E6	0.3	0.6	0.45	BAJO
E7	0.8	0.6	0.7	MEDIO
E8	0.9	0.9	0.9	ALTO

Tabla 3: Coeficiente de Competencia de los Expertos

Un aspecto importante en la selección de los expertos es el número de expertos que debe tener el grupo. No existe una norma generalizada para determinar el número óptimo de expertos, sin embargo, hasta siete expertos el error disminuye exponencialmente, después de treinta, aunque el error disminuye lo hace de manera poco significativa y no compensa el incremento de costos y esfuerzo, por lo que se sugiere utilizar un número de expertos en el intervalo de siete a treinta (42).

El experto 6 deja de formar parte del grupo de validación de la propuesta, debido a que su Coeficiente de Competencia es Bajo. En la Figura 9 se representa el resultado de acuerdo al coeficiente de competencia del grupo resultante para la validación de la propuesta.

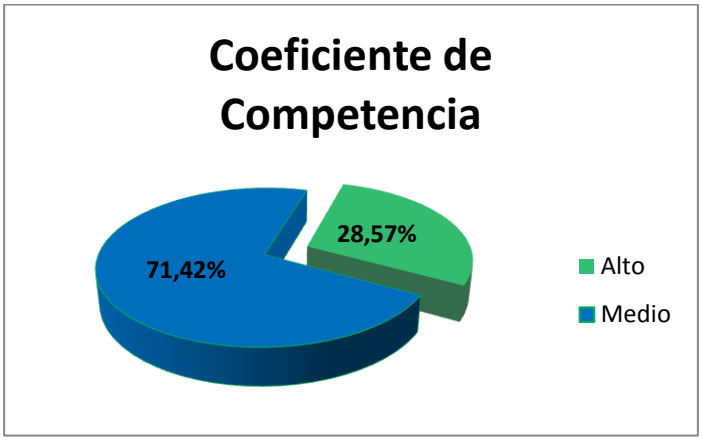

Figura 9: Gráfica. Coeficiente de Competencia

Al tener el número total de expertos (7) que se utilizarán para la validación, se formulan las preguntas que no deben ser demasiadas, pero si sobre cuestiones medulares referentes a la investigación que se realizó para buscar los criterios relativos a la temática sometida a consideración.

3.3.2 Elaboración de cuestionarios para la validación de la propuesta

Una vez seleccionados los expertos, se prosigue con la elaboración de la encuesta de validación, por lo que se hace necesario elaborar un cuestionario de forma tal que se adapten a las condiciones de los expertos. El cuestionario consta de 8 preguntas y fue conformado de forma tal que las respuestas fueran categorizadas en Muy Adecuada (C1), Bastante Adecuada (C2), Adecuada (C3), Poco Adecuada (C4) y No Adecuada (C5).

Finalmente en la última pregunta se solicita un comentario general donde se aporten mejoras al proceso. Los especialistas tuvieron a su disposición la documentación de la guía propuesta y se les requirió un tiempo determinado para las respuestas o hacer las preguntas pertinentes que les hubiesen surgido al presentar el documento. Para conocer en detalle el cuestionario de validación, ver Anexo 2.

3.3.3 Determinación de la concordancia de los expertos

Para darle mayor validez a la propuesta se necesita calcular el Coeficiente de Concordancia de Kendall, el cual permite comprobar el grado de coincidencia de las valoraciones realizadas por los expertos.

Para la aplicación del Coeficiente de Concordancia de Kendall se construye una tabla que contiene los aspectos evaluados en la encuesta contra los Expertos a los que se le realizó la misma, en esta tabla se sitúan los rangos de valoración en términos numéricos del uno al cinco, tomando el valor más alto (5) como C1 (Muy Adecuado) y así respectivamente. Estos datos son tomados a partir de la encuesta de validación realizada a los expertos. Para acceder a esta tabla consultar Anexo 3

Después de confeccionar la tabla se realiza:

- La suma de los valores numéricos asignados a cada valor que se evalúa, según el criterio emitido por cada uno de los expertos (Rj).
- El valor medio ($\overline{Rj}$), dado por la sumatoria de las Rj entre N, siendo esta última el total de aspectos a evaluar (los aspectos serán el número de preguntas del cuestionario, en este caso N= 8).
- La desviación media, dada por la diferencia entre cada Rj y el valor de la media.
- La suma de los cuadrados de las desviaciones medias, S.
- El cuadrado del número total de expertos, K. En este caso K=7.
- El cubo del número total de aspectos a evaluar, N.
- La diferencia entre el cubo de N y N y su multiplicación por el cuadrado de K.

Una vez que se tienen todos estos datos es posible calcular el Coeficiente de Kendall (W) a través de la fórmula siguiente:

$$W = \frac{12 * S}{k^2(N^3 - N)}$$

El coeficiente de Kendall (W) brinda el valor que permite determinar el nivel de concordancia entre los expertos. Este valor W siempre es positivo y va a oscilar entre 0 y 1, además con él se puede calcular el Chi Cuadrado real, precisamente para observar si existe o no concordancia entre los expertos y se calcula mediante la fórmula siguiente:

$$X^2 = K(N-1)W$$

Después de calcular el Chi Cuadrado se procede a comparar el valor con el de las tablas estadísticas. Si se cumple que $x^2 real < x^2\ (\infty, N-1)$ entonces quiere decir que existe concordancia entre los expertos. Teniendo en cuenta la probabilidad de error de 0.05 y después de realizar los cálculos se concluye que $x^2 real$ = 0.51190476 y $x^2\ (0.05, 7)$ =

14,0671404, lo cual afirma el cumplimiento de la comparación y por tanto la concordancia entre los expertos. Para acceder a los cálculos realizados, consultar Anexo 3.

3.3.4 Desarrollo práctico de los resultados

Los expertos que formaron parte del panel recibieron el cuestionario a responder con un total de 8 preguntas, los cuales fueron enviados vía e-mail garantizando el anonimato de los mismos, debido a que fueron enviados a cada uno por separado, evitando que se supiera el nombre del resto de los miembros del panel.

Se confeccionaron tablas para ir recogiendo los resultados aportados por los expertos. Para ello se utilizó el programa Excel 2010 y dichos resultados se recogieron en una tabla como la que sigue:

Tabla de Frecuencias Absolutas							
No	**Preguntas**	**Criterio de Expertos**					
		C1(MA)	**C2(BA)**	**C3(A)**	**C4(PA)**	**C5 (NA)**	**Total**
1	Pregunta 1	3	3	1	0	0	7
2	Pregunta 2	4	3	0	0	0	7
3	Pregunta 3	5	2	0	0	0	7
4	Pregunta 4	2	4	1	0	0	7
5	Pregunta 5	3	4	0	0	0	7
6	Pregunta 6	4	2	1	0	0	7
7	Pregunta 7	5	2	0	0	0	7
8	Pregunta 8	2	5	0	0	0	7

Tabla 4: Frecuencias Absolutas

La Tabla 4 queda representada gráficamente como se muestra en la Figura 10

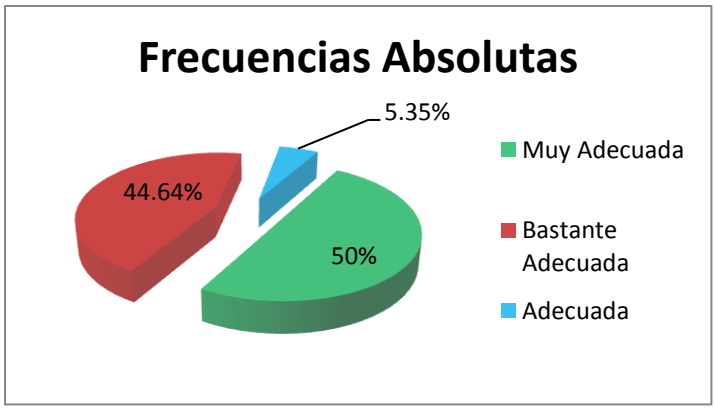

Figura 10: Gráfica. Frecuencias Absolutas

Una vez tabulados todos los datos, se realizan los siguientes pasos para obtener los resultados deseados:

- **PRIMER PASO**: Se construye una tabla de Frecuencias Acumuladas donde cada número en la fila (excepto el primero) se obtiene sumándole el anterior.

Tabla de Frecuencias Absolutas Acumuladas						
No	**Preguntas**	**Criterio de Expertos**				
		C1(MA)	C2(BA)	C3(A)	C4(PA)	C5 (NA)
1	Pregunta 1	3	6	7	7	7
2	Pregunta 2	4	7	7	7	7
3	Pregunta 3	5	7	7	7	7
4	Pregunta 4	2	6	7	7	7
5	Pregunta 5	3	7	7	7	7
6	Pregunta 6	4	6	7	7	7
7	Pregunta 7	5	7	7	7	7
8	Pregunta 8	2	7	7	7	7

Tabla 5**:** Frecuencias Absolutas Acumuladas

Observación: En la tabla de Frecuencias Absolutas Acumuladas la última columna desaparece

- **SEGUNDO PASO:** Se construye la tabla de Frecuencias Relativas Acumuladas.

Para construir esta tabla, se divide el valor de cada celda de la tabla anterior entre el número de expertos consultados, en este caso 7. El cociente de esa división debe aproximarse hasta la diez-milésima. Queda eliminada una columna pues hay 5 categorías y solo se necesitan 4 puntos de corte (con cuatro puntos se obtienen 5 intervalos).

Tabla de Frecuencias Relativas Acumuladas					
No	**Preguntas**	**Criterio de Expertos**			
		C1(MA)	**C2(BA)**	**C3(A)**	**C4(PA)**
1	Pregunta 1	0.42857143	0.8571429	0,9999	0,9999
2	Pregunta 2	0.57142857	0.9999	0,9999	0,9999
3	Pregunta 3	0.57142857	0.9999	0,9999	0,9999

4	Pregunta 4	0.28571429	0.8571429	0,9999	0,9999
5	Pregunta 5	0.28571429	0.9999	0,9999	0,9999
6	Pregunta 6	0.28571429	0.8571429	0,9999	0,9999
7	Pregunta 7	0.71428571	0.9999	0,9999	0,9999
8	Pregunta 8	0.14285714	0.9999	0,9999	0,9999

Tabla 6: Frecuencias relativas acumuladas

- **TERCER PASO:** Se buscan las imágenes de los elementos de la tabla anterior por medio de la función (Distribución Normal. Standard Invertida). A la misma tabla se le adicionan tres columnas y una fila para colocar los resultados que se mencionan a continuación:

✓ Suma: Sumatoria de cada fila y de cada columna según sea el caso.

✓ P: Promedio de la suma de cada fila.

✓ N: División de la sumatoria de las sumas de las filas entre el resultado de multiplicar el número de categorías (5) por el número de preguntas (8).

✓ N-P: Es entonces el valor promedio que le otorgan los expertos consultados a cada pregunta o afirmación sobre el proceso.

✓ Punto de corte: Promedio de la suma de cada columna.

La tabla 7 resume lo dicho en los puntos anteriores:

Puntos de Corte								**N=**	**1.98245847**
No	**Preguntas**	**C1**	**C2**	**C3**	**C4**	**Suma**	**P**	**N-P**	
1	Pregunta 1	-0.18001237	1.06757071	3.72	3.72	8.32755835	2.0818896	-0.09943112	Bastante adecuado
2	Pregunta 2	0.180012366	3.71901649	3.72	3.72	11.3390289	2.8347572	-0.85229874	Muy adecuado
3	Pregunta 3	0.180012366	3.71901649	3.72	3.72	11.3390289	2.8347572	-0.85229874	Muy adecuado
4	Pregunta 4	-0.56594881	1.06757071	3.72	3.72	7.9416219	1.9854055	-0.00294701	Bastante adecuado
5	Pregunta 5	-0.56594881	3.71901649	3.72	3.72	10.5930677	2.6482669	-0.66580845	Muy adecuado
6	Pregunta 6	-0.56594881	1.06757071	3.72	3.72	7.9416219	1.9854055	-0.00294701	Bastante adecuado
7	Pregunta 7	0.565948809	3.71901649	3.72	3.72	11.7249653	2.9312413	-0.94878285	Muy adecuado
8	Pregunta 8	-1.06757054	3.71901649	3.72	3.72	10.0914459	2.5228615	-0.54040302	Muy adecuado
Suma		-2.01945579	21.7977946	29.76	29.76	158.596678			
Puntos de Corte		-0.25243197	2.72472432	3.72	3.72				

Tabla 7: Puntos de Corte

Las sumas obtenidas en las cuatro primeras columnas dan los puntos de corte. Estos se utilizan para determinar el grado de adecuación de los indicadores según los criterios de los expertos seleccionados. Para ello se opera del modo siguiente:

Muy Adecuado	Bastante Adecuado	Adecuado	Poco Adecuado	No Adecuado
N-P<= -0.25243197	> N-P < = 2.72472432	> N-P < =3.72	> N-P <=3,72	-------

Tabla 8: Grados de adecuación

En la Figura 11 se presenta un resumen gráfico con los resultados obtenidos de acuerdo a los puntos de corte. En ella se evidencia en qué rango se encuentra la pregunta (valor de N-P) con respecto al punto de corte para determinar si la misma es muy adecuada, bastante adecuada, adecuada o poco adecuada.

Figura 11: Gráfica. Puntos de Corte

A continuación se recogen las recomendaciones dadas por los expertos al proceso propuesto, que no divergen de la propuesta pero si aportan ideas sobre cómo escalarlo.

- ✓ Se propone especificar qué acciones correctivas se podrían realizar para garantizar una capacitación sobre el tema antes de poner en práctica dicho procedimiento, para una mejor comprensión y aplicación del tema.
- ✓ Se debe tener en consideración las etapas del desarrollo de software en la definición del modelo.

- ✓ Se propone enriquecer el procedimiento con las herramientas (o artefactos) que deben usar las personas que sigan este proceso (plantillas, formularios, listas de chequeo, matrices).

Todos los aspectos a validar del proceso fueron considerados por los expertos de Muy Adecuado y Bastante Adecuado, obteniéndose el 63 % y 37 % respectivamente. El último valor muestra un nivel de significancia elevado con respecto al de Muy Adecuado ya que los expertos no discreparon de la propuesta, solo aportaron recomendaciones sobre cómo mejorarla. En la Figura 12 se presenta un resumen gráfico con los resultados obtenidos.

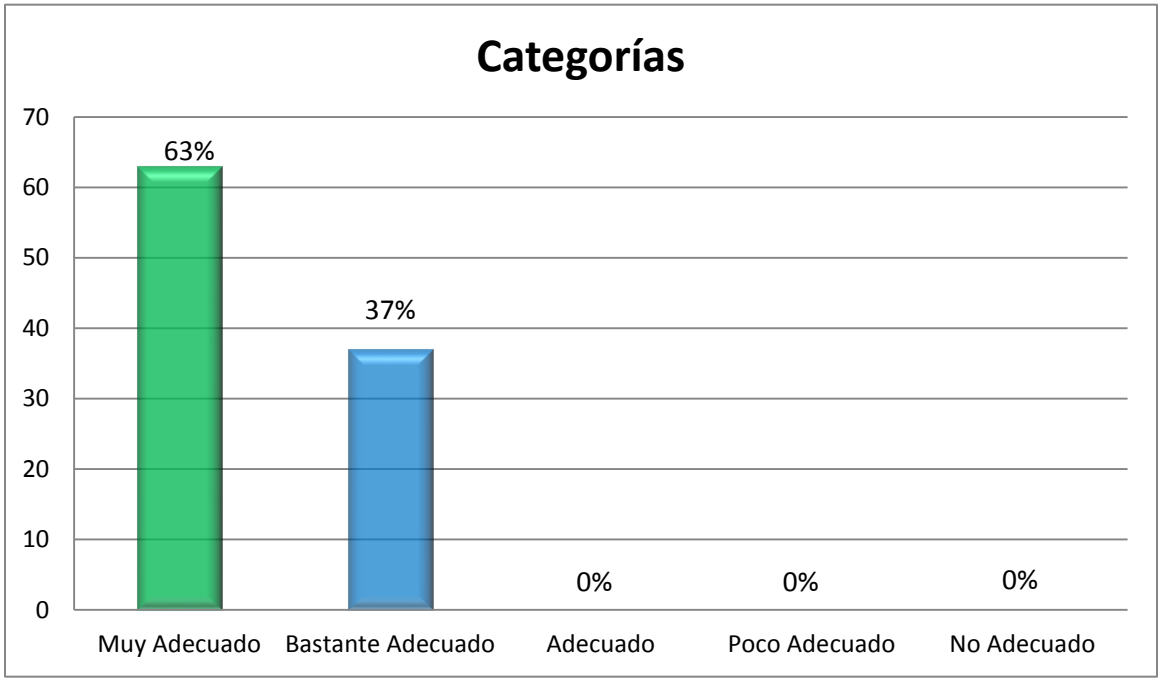

Figura 12: Gráfica. Resultados generales de la encuesta

De acuerdo a los resultados obtenidos se puede resumir que la validación resulta Muy Adecuada en cuanto a su elaboración teórica, demostrando utilidad, adecuación y la validez de los objetivos propuestos. Los resultados arrojados fueron satisfactorios y se consideró suficiente una sola ronda de preguntas.

Conclusiones parciales

En este capítulo se validó la solución propuesta para la definición de Modelo de Dominio Específico y Lenguaje de Dominio Específico para los Sistemas de Gestión del Aprendizaje, utilizando el método de validación Delphi. Se seleccionaron 7 expertos que dieron su criterio respecto a 8 preguntas relacionadas con el procedimiento planteado, los resultados obtenidos fueron satisfactoriamente evaluados de Bastante Adecuado y Muy Adecuado, por lo que se concluye que el modelo es Muy Adecuado según la opinión de los expertos encuestados.

Conclusiones

Una vez realizada esta investigación se concluye que:

- ✓ Durante el estudio investigativo se analizaron las cuatro vertientes principales del Desarrollo de Software Dirigido por Modelos utilizando para el desarrollo de la propuesta el Modelo de Dominio Específico por los grandes beneficios que aporta a la productividad disminuyendo la incidencia de errores humanos y facilitando la captura de requerimientos.
- ✓ De la documentación consultada no se encontró ninguna que detallara alguna metodología o procedimiento que ayudara a una correcta definición del Modelo de Dominio Específico y Lenguaje de Dominio Específico, los trabajos consultados solo se guían por lo que plantea OMG (del inglés Object Management Group) y se preocupan más por la aplicación que generan que por explicar el método para lograr su solución.
- ✓ Se realizó un procedimiento para definir el Modelo de Dominio Específico y Lenguaje de Dominio Específico para los Sistemas de Gestión del Aprendizaje, donde se definieron 2 etapas, además de un conjunto de pasos complementarios que apoyarán a los nuevos investigadores en el desarrollo de este proceso.
- ✓ Finalmente, se logró la validación de la propuesta mediante el método Delphi donde participaron 7 expertos, y mediante los criterios aportados se obtuvieron resultados satisfactorios, catalogando al 63 % de los aspectos como Muy Adecuados y al restante 37% como Bastante Adecuado, quedando así reflejado en las tabulaciones realizadas en el capítulo 3.

Recomendaciones

Se recomienda:

- ✓ Utilizar la herramienta de metamodelado Eclipse ya que es de libre distribución, y multiplataforma, además cuenta con las funcionalidades requeridas para realizar correctamente la definición de Modelo de Dominio Específico y Lenguaje de Dominio Específico.
- ✓ Enmarcar el procedimiento en una de las etapas definidas por las metodologías de desarrollo de software.
- ✓ Especificar qué acciones correctivas se podrían realizar para garantizar una capacitación sobre el tema antes de poner en práctica dicho procedimiento, para una mejor comprensión y aplicación del tema.
- ✓ Hacer uso del procedimiento propuesto para definir el Lenguaje de Dominio Específico y Modelo de Dominio Específico de una herramienta que permita crear un Sistema de Gestión del Aprendizaje con un enfoque al Desarrollo de Software Dirigido por Modelos.

Glosario de Términos

Constitutivas: Que constituye una cosa en el ser de tal y la distingue claramente de otras. Que forma parte fundamental de una cosa y la distingue de las demás.

Estándar: Herramienta base de la interoperabilidad informática. Permite definir cómo interactúan los componentes informáticos existentes (43).

Instancias: Es una copia de una versión ejecutable del programa que ha sido escrito en la memoria del computador. Es creada típicamente por el click de usuario en un ícono de una interfaz Gráfica para usuarios o por la entrada de un comando en una interfaz de línea de comandos.

Mapear: Trasladar a un mapa sistemas o estructuras conceptuales.

Método: Es un modo ordenado y sistemático de proceder para alcanzar un objetivo determinado. Es un procedimiento o conjunto de procedimientos que se utilizan de instrumentos para lograr los objetivos de la investigación.

Metodología: Es la teoría acerca del método o del conjunto de métodos. La metodología es normativa ya que valora y organiza un conjunto de comportamientos y de reglas técnicas donde prescribe como dirigir los mayores problemas del desarrollo (43), es descriptiva (expone) o comparativa (analiza). La metodología es el estudio analítico y crítico de los métodos de investigación**.**

Modelo: Un modelo como representación simplificada de una realidad que se quiere plasmar o manipular, es uno de los conceptos fundamentales en el que se basa el enfoque de Modelo de Dominio Especifico. Entre las definiciones de modelo se destacan: Descripción o especificación de un sistema y su adaptación a un propósito determinado. Comúnmente se presenta a un modelo como la combinación de texto y dibujos. El texto puede estar en un lenguaje de modelado o en lenguaje natural. Representa una abstracción del sistema que permite concentrarse en los detalles más interesantes dejando a un lado otros aspectos secundarios.

Paradigma: Define a un modelo o patrón en cualquier disciplina científica o contexto epistemológico .Es el conjunto de prácticas que definen una disciplina científica durante un período específico de tiempo.

Plugin: Es un programa que incrementa o aumenta las funcionalidades de un programa principal. Módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande.

Proceso: Conjunto de actividades o eventos que se realizan o suceden con un determinado fin. Cada proceso tiene entradas, funciones y salidas.

Prospectivo: Expresa el grado de acercamiento de los elementos constitutivos del perfil profesional a los cambios más trascendentales que se avizoran como resultado del desarrollo científico-técnico, tecnológico, medioambiental, social y laboral relacionado con la profesión y el profesional, a corto, mediano y largo plazo.(44)

Semántica: Se refiere al significado de los conceptos y de las relaciones en el lenguaje lo cual es necesario para comprender el lenguaje.

Software: Conjunto de programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un Sistema de Computación.

Referencias Bibliográficas

1. **UCI.** *Universidad de las Ciencias Informáticas.* [En línea] 2012. http://www.uci.cu/mision.

2. **Fortes.** [En línea] 2011. http://portal.fortes.prod.uci.cu.

3. **KME.** *KME, Knowledge Management Environment.* [En línea] 2007. http://www.nuevosmedios.ws/ayudakme/index.html?lms.html.

4. **Montenegro Marín, Carlos Enrique .** *Modelado Específico de Dominio para la Construcción de Learning Objects Independientes de la Plataforma.* Universidad de Oviedo : s.n., 2011.

5. **Sigossee.** Evaluación de las plataformas LMS. [En línea] 2007. http://www.guidance-research.org/sigossee/join/sp.

6. **Aprea.** *Un Enfoque de Fábrica de Software a Soluciones de HL7 Versión 3. 18.* 2006.

7. **Fuentes , Lidia y Sánchez, Pablo.** *Desarrollo de software con aspectos dirigido por modelos.* Universidad de Málaga, España : s.n., 2006.

8. **García Molina, Jesus y Lasheras Velasco, Joaquín.** *Una experiencia en transferencia de tecnología de Desarrollo de Software Dirigido por Modelos.* Universidad de Murcia : s.n., 2007.

9. **Moñino Martínez, José Antonio .** *Aplicación del Modelado Específico de Dominio a las Redes de Sensores Inalámbricos.* Universidad Politécnica de Cartagena : s.n., 2007.

10. **Mora, Beatriz y Ruiz, Francisco.** *Definición de Lenguajes de Modelos MDA vs DSL.* 2006.

11. **Montenegro Marín, Carlos Enrique.** *Aplicación de Ingeniería Dirigida por Modelos (MDA), para la construcción de una herramienta de Modelado de Dominio Específico (DSM) y la creación de módulos en Sistemas de Gestión de Aprendizaje (LMS) independientes de la plataforma.* 2011.

12. **Martínez Ortiz, Iván .** *Aplicación de Técnicas de Ingeniería de Lenguajes al Campo del Modelado Educativo.* Universidad Complutense de Madrid, España : s.n., 2011.

13. **Seidewitz, Ed.** *What Models Mean.* s.l. : IEEE Computer Society, 2006.

14. **Mellor, Stephen.** *Executable UML: A Foundation for Model-Driven Architectures.* 2007.

15. **OMG.** *Meta Object Facility (MOF) 2.0 Core Specification.* 2007.

16. **OMG**. *OMG's MetaObject Facility.* 2011.

17. **Alonso Cáceres, Diego .** *Desarrollo de Software para Robots de Servicio: Un enfoque dirigido por modelos y orientado a componentes.* Universidad Politécnica de Cartagena : s.n., 2008.

18. **Begoña Pelayo, Cristina .** *Desarrollo ágil de Software con Arquitecturas Dirigidas por Modelos.* Universidad de Oviedo : s.n., 2007.

19. **Meliá, Santiago.** *WebSA: Un método de Desarrollo Dirigido por Modelos de Arquitectura para Aplicaciones Web.* Universidad de Alicante : s.n., 2007.

20. **OMG.** *UML 2.0 Superestructure Specification.* 2010.

21. **García Magariño, Iván .** *Representación de las Relaciones en los Metamodelos con el Lenguaje Ecore.* 2007.

22. **García Molina, Jesús Joaquín.** *Herramientas de Metamodelado: Microsoft DSL Tools vs MetaEdit+.* Universidad de Murcia : s.n., 2007.

23. **García, Di Lorenzo.** *Herramientas de soporte al proceso de desarrollo dirigido por modelos y su implementación con DSL Tools.* Universidad Tecnológica Nacional, Argentina : s.n., 2007.

24. **Cueva, Díaz.** *Towards the systematic measurement of ATL transformation models.* 2010.

25. **Microsoft.** *El proceso de transformación de las plantillas de texto.* 2010.

26. **Andino, Luciano Omar y Esteban Ruiz, German.** *Análisis y uso de los frameworks de Eclipse para la definición de DSLs.* 2009.

27. **Pinilla, Felipe.** *CMapTools.* 2010.

28. **Pons, Juan , Pons, Claudia y Vaquero, Marcelo.** *Low cost wireless computer system for quality evaluation of grains stored in Silobags.* Universidad Nacional de La Plata : s.n., 2010.

29. **Heidenreich.** *Derivation and Refinement of Textual Syntax for Models.* 2009.

30. **Orallo Hernández, Enrique.** *El Lenguaje Unificado de Modelado (UML).* [En línea] [Citado: Enero 12, 2012.] http://www.disca.upv.es/enheror/pdf/ActaUML.PDF.

31. **Esteban, Omar.** *Análisis y uso de los frameworks de Eclipse para la definición de DSLs.* 2009.

32. **GMP.** *Graphical Modeling Project (GMP).* [En línea] 2011. http://www.eclipse.org/modeling/gmp.

33. **Hebach.** *MDA with QVT.* 2006.

34. **Oldevik, Jon.** *MOFScript* User Guide. *version 0.6 (MOFScript v 1.1.11).* [En línea] Noviembre 2, 2006. http://www.eclipse.org/gmt/mofscript/doc/MOFScript-User-Guide.pdf.

35. **JET, M2T-.** *M2T- JET*. [En línea] 2010. http://wiki.eclipse.org/M2T-JET.

36. **MOFScript.** *MOFScript.* [En línea] 2010. http://www.eclipse.org/gmt/mofscript/.

37. **Mohedano, Félix Ortega.** *El método Delphi, prospectiva en Ciencias Sociales a través del análisis de un caso práctico.* Universidad EAN : Revista EAN, 2008.

38. **Castellanos, Orvelis Alba.** Manual de validación de perfiles profesionales y estándares de competencia. *Manual de validación de perfiles profesionales y estándares de competencia.* [En línea] mayo 2011. http://www.eumed.net/rev/ced/27/oac.htm.

39. **Jaimes, Marisol Carreño.** *El método Delphi: cuando dos cabezas piensan más que una en el desarrollo de guías de práctica clínica.* Universidad Javeriana, Bogotá, Colombia : s.n., 2009.

40. **Cedeño Hernández, Yudeisy y Díaz Macias, Dayana.** *Guía para elevar el nivel de mantenibilidad en bases de datos Oracle en el desarrollo de aplicaciones informáticas.* Universidad de las Ciencias Informáticas : s.n., 2010.

41. **Rosales Pérez, Dianella y Marquez Cabrera, Raul .** *Propuesta de Modelo de Estructura y Evolución para un Centro de Excelencia SOA.* Universidad de las Ciencias Informáticas : s.n., 2010.

42. **Durand, R.** *El método Delphi y la perspectiva del hidrógeno.* España : s.n., 2010.

43. **EVA.** *EVA.* [En línea] 2011. [Citado: noviembre 2, 2011.] http://eva.uci.cu/mod/resource/view.php?id=14131.

44. **Opentia.** *Estudio sobre estándares informáticos, tipos y caracterizaciones.* 2007.

Bibliografía

Alonso Cáceres, Diego . 2008. *Desarrollo de Software para Robots de Servicio: Un enfoque dirigido por modelos y orientado a componentes.* Universidad Politécnica de Cartagena : s.n., 2008.

Andino, Luciano Omar y Esteban Ruiz, German. 2009. *Análisis y uso de los frameworks de Eclipse para la definición de DSLs.* 2009.

Cedeño Hernández, Yudeisy y Díaz Macias, Dayana. 2010. *Guía para elevar el nivel de mantenibilidad en bases de datos Oracle en el desarrollo de aplicaciones informáticas.* Universidad de las Ciencias Informáticas : s.n., 2010.

Ecured. 2011. [En línea] 2011. http://www.ecured.cu/index.php/Universidad_de_las_Ciencias_Informaticas.

FORTES. 2011 . [En línea] 2011. http://portal.fortes.prod.uci.cu.

Fernández Rubén, Laguna Miguel A. 2009. *Un Lenguaje Específico de Dominio para Líneas de Productos Software***.** Universidad de Valladolid, España: s.n., 2009

Fuentes , Lidia y Sánchez, Pablo. 2006. *Desarrollo de software con aspectos dirigido por modelos.* Universidad de Málaga, España : s.n., 2006.

García Magariño, Iván . 2007. *Representación de las Relaciones en los Metamodelos con el Lenguaje Ecore.* 2007.

García Molina, Jesús Joaquín. 2007. *Herramientas de Metamodelado: Microsoft DSL Tools vs MetaEdit+.* Universidad de Murcia : s.n., 2007.

GMP. 2011. *Graphical Modeling Project (GMP).* [En línea] 2011. http://www.eclipse.org/modeling/gmp.

Jaramillo Pérez, Carlos Mario. [En línea] 2008. [Citado: mayo 9, 2010.] www.escuelagobierno.org/v1/archivos.php?descargar=78..

JET, M2T-. 2010. *M2T- JET*. [En línea] 2010. http://wiki.eclipse.org/M2T-JET

KME. *KME, Knowledge Management Environment*. [En línea] 2007. http://www.nuevosmedios.ws/ayudakme/index.html?lms.html.

Martínez Ortiz, Iván . 2011. *Aplicación de Técnicas de Ingeniería de Lenguajes al Campo del Modelado Educativo.* Universidad Complutense de Madrid, España : s.n., 2011.

Meliá, Santiago. 2007. *WebSA: Un método de Desarrollo Dirigido por Modelos de Arquitectura para Aplicaciones Web.* Universidad de Alicante : s.n., 2007.

OMG. 2007. *Meta Object Facility (MOF) 2.0 Core Specification.* 2007.

OMG. 2011. *OMG's MetaObject Facility.* 2011.

OMG. 2010. *UML 2.0 Superestructure Specification.* 2010.

Papatsoutsos, Dimitrios. *Information Systems Development Methodologies in the Age of Digital Economy.* Universidad de Atenas : s.n., 2007.

Pons, Juan , Pons, Claudia y Vaquero, Marcelo. 2010. *Low cost wireless computer system for quality evaluation of grains stored in Silobags.* Universidad Nacional de La Plata : s.n., 2010.

Seidewitz, Ed. 2006. *What Models Mean.* s.l. : IEEE Computer Society, 2006.

Anexos

Anexo 1: Encuesta realizada a proyectos del Centro FORTES

Nombre del proyecto: _____

Rol que desempeña: _____

¿Cuál de los siguientes riesgos inciden con mayor intensidad en su proyecto?

Riesgos	Marca (x)	Cant. de Incidencias	%
Retraso en el cronograma de entrega		13	76.5
Retraso en la realización de tareas asignadas en el proyecto		4	23.5
Poca comunicación entre el cliente y el equipo de desarrollo		4	23.5
Inestabilidad o baja del personal		4	23.5
Divorcio entre analistas y programadores		11	64.7
Complejidad a la hora de gestionar los requisitos		12	70.6

Otros riesgos: _____

Nota: La encuesta se le realizó a 17 miembros de diferentes proyectos que de cierta forma se encuentran enfocados a los Sistemas de Gestión del Aprendizaje.

Anexo 2: Cuestionario de evaluación de la Propuesta

Datos

Nombre y apellidos: _____

Años de experiencia: _____

Categoría docente: _____

Usted ha sido seleccionado para participar en una encuesta con el fin de validar la Propuesta para definir Modelo de Dominio Específico y Lenguaje de Dominio Específico para Sistemas de Gestión del Aprendizaje. Esta validación se lleva a cabo por el método Delphi. Para responder el cuestionario que aparece a continuación primero debe haber leído la investigación; de antemano se le asegura que nadie podrá saber quién es el encuestado y además se garantiza que sus opiniones se tendrán en cuenta para la posterior aplicación.

Valore el grado de factibilidad de cada pregunta de acuerdo con la siguiente escala:

Muy Adecuada (C1), Bastante Adecuada (C2), Adecuada (C3), Poco Adecuada (C4) y No Adecuada (C5).

No	Preguntas	Criterio de Expertos				
		C1 (MA)	C2 (BA)	C3 (A)	C4 (PA)	C5 (NA)
1	¿La propuesta de las dos etapas y sus pasos complementarios para lograr la correcta definición de Modelo de Dominio Específico y Lenguaje de Dominio Específico, UD. las considera?					
2	¿La propuesta de realizar un estudio de los conceptos y elementos principales de los Sistemas de Gestión del Aprendizaje, UD. la considera?					
3	¿Considera correctamente definidos los 3 elementos principales (sintaxis abstracta, sintaxis concreta y semántica) del Lenguaje de Dominio Específico?					
4	En el procedimiento se diferencian claramente los conceptos y elementos principales de la sintaxis abstracta con respecto a la sintaxis concreta. ¿Cómo considera usted esta afirmación?					
5	¿Considera correcto el orden de pasos a seguir para definir el Lenguaje de Dominio Específico?					
6	¿Cuán adecuado es el procedimiento para definir Modelo de Dominio Específico y Lenguaje de Dominio Específico?					
7	La propuesta es útil y servirá para guiar a los nuevos investigadores en el desarrollo del proceso de definición de Modelo de Dominio Específico y Lenguaje de Dominio Específico. ¿Cómo considera usted esta afirmación?					
8	¿Considera que la aplicación de este procedimiento tendrá éxito en los proyectos que utilizan Sistemas de Gestión del Aprendizaje?					

Exprese otros criterios o recomendaciones que pudieran servir para perfeccionar la propuesta. Puede especificar si considera qué se debe adicionar, suprimir o modificar algún elemento en concreto: _____

Anexo 3: Cálculo del coeficiente de Kendall

K es el número de expertos que intervienen en el proceso de validación, por lo que toma el valor de 7. N cantidad de aspectos a validar, en este caso N = 8. Rj es la suma de los rangos asignados a cada pregunta por parte de los expertos.

$\overline{Rj}$ es la media de los rangos y se determina mediante la fórmula :

$$\overline{Rj} = \frac{\sum_{j=i}^{n} Rj}{N}$$

Preguntas	Experto 1	Experto 2	Experto 3	Experto 4	Experto 5	Experto 6	Experto 7	Rj
Pregunta 1	4	5	4	4	5	4	4	30
Pregunta 2	5	4	4	5	5	4	5	32
Pregunta 3	4	5	5	5	4	5	5	33
Pregunta 4	4	4	4	5	4	4	4	29
Pregunta 5	5	4	5	4	4	5	4	31
Pregunta 6	4	4	4	4	4	4	5	29
Pregunta 7	5	5	5	4	5	5	4	33
Pregunta 8	4	4	4	4	4	4	5	29

Obteniendo el valor de $\overline{RJ} = \frac{30+31+32+2*33+3*29}{8} = 30.75$

S es la suma de los cuadrados de las desviaciones y se calcula de la siguiente forma:

$$\sum_{j-1}^{n} \left(R_j - \overline{Rj}\right)^2$$

Donde S = 21.5

W es el coeficiente de Kendall y se calcula mediante la fórmula siguiente:

$$W = \frac{12 * S}{k^2(N^3 - N)}$$

Sustituyendo los valores obtenidos en la ecuación:

$W = \frac{12 * 21.5}{49(512-8)} = 0{,}01$

Luego se procede con el cálculo del Chi-Cuadrado para poder ver si existe concordancia entre los expertos: x^2= K(N-1) W= 7 (8-1) 0.01= 0.51190476

Este Chi-Cuadrado se compara con el de la tabla inversa de la función de distribución de la variable Chi-Cuadrado con una probabilidad de error de 0,05.

Si el Chi-Cuadrado real ($x^2\ real$) es menor que el Chi Cuadrado de la tabla (x^2 (∞, N-1)) entonces hay concordancia entre los expertos:

$$x^2 real < x^2 \ (\infty, \text{N-1})$$

$$x^2 real < x^2 \ (0{,}05, 7)$$

$$0.51190476 < 14{,}0671404$$