**Universidad de las Ciencias Informáticas**

**Facultad 10**


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

---

# Título: Validación de herramientas de metamodelado.

**Autores: Aimé Ramírez Benitez**

**Lissette Fonseca Suárez**

**Tutores: MSc. David Leyva Leyva**

**Ing. Susel Cañete Pollán**

Ciudad de La Habana. Junio, 2009

*Todos y cada uno de nosotros paga puntualmente su cuota de sacrificio consciente de recibir el premio en la satisfacción del deber cumplido, conscientes de avanzar con todos hacia el Hombre Nuevo que se vislumbra en el horizonte.*

*Ernesto Che Guevara*

# DATOS DE CONTACTO

**Tutor: MSc. David Leyva Leyva:** Jefe de Departamento de Técnicas de Programación, Facultad 10, Universidad de Ciencias Informáticas. Teléfono: (53) (07) 837 2511. davidl@uci.cu Licenciado en Cibernética-Matemática, Universidad de Las Villas (UCLV), 1988. Máster en Computación Aplicada, Universidad de Las Villas (UCLV), 1995.

Profesor de la Universidad de Holguín desde 1990. Jefe de Departamento de Informática (2003-2005). Participó en un Proyecto de Educación a Distancia en la Universidad del 2001 hasta el 2005, año en el que comenzó a trabajar en la Universidad de Ciencias Informáticas. Ha participado en un gran número de eventos nacionales e internacionales. Le fueron concedidas sendas becas Intercampus en La Universidad de Castilla-La Mancha (1998) y La Universidad Autónoma de Madrid (2000). Trabajó como profesor invitado en la University of Belize (Belice, de 2001 a 2003) y de la Universidad Nacional de Ingeniería (Managua, Nicaragua, 2005).

Actualmente es miembro de un grupo de investigación y desarrollo orientado a la extensión de la plataforma Moodle y el desarrollo de otras herramientas para la teleformación.

**Tutor: Ing. Susel Cañete Pollán**: Profesora de Práctica Profesional V. Facultad 10, Universidad de Ciencias Informáticas. Carretera a San Antonio de los Baños Km. 2 ½, Torrens, Boyeros, Ciudad de La Habana. Cuba. Teléfono: (53) (07) 837 2543. scanete@uci.cu Ingeniera en Ciencias Informáticas, Universidad de Ciencias Informáticas, 2007.

Alumna Ayudante durante su carrera. Desarrolló su Trabajo de Diploma en la temática de las Técnicas Formales asociadas a la fiabilidad del software.

# AGRADECIMIENTOS

*A mi mami Digna por ser madre y abuela, por cuidarme y educarme siempre, por darme su apoyo y comprensión.*

*A mi papá Arnulfo y su esposa Julia por apoyarme incondicionalmente en esta etapa tan importante.*

*A mi esposo Ismel (Michi) por su paciencia, comprensión y amor, por su gran ayuda en la realización de este trabajo. Te adoro.*

*A mi hermanita Niurka que aunque estos 5 años nos han distanciado un poco siempre la tengo en mi corazón.*

*A Chacho, mi tío Juan Miguel, mi abuela Kuka y en general a toda mi familia por quererme así, tal como soy.*

*A Leticia (Kuky) mi gran amiga y confidente durante estos 5 años, y Nidia por estar siempre presente.*

*A mis nuevos amigos Israel, Gilberto y Reinier.*

*Y a los no tan nuevos pero muy importantes Lissetica, Chuchy, Yoana, Jorgito, Oscarito y Henry que hacen de mi estancia en Aguas Claras algo maravilloso.*

*A mi compañera de cuarto y tesis Lissette.*

*A los tutores Susel y David, por su ayuda incondicional sin la cual este sueño no hubiese sido posible.*

*A todos los que de una forma u otra han contribuido durante estos 5 años en mi formación tanto personal, como laboral.*

*Aimé Ramírez Benítez*

# AGRADECIMIENTOS

*Agradezco a toda mi familia que me ayudó y apoyó desde que estaba pequeña para que pudiese llegar a ser quien soy hoy.*

*A mi mamá Estela, por haberme educado y formado como lo hizo con mucho cariño y amor, por ser la madre que es y siempre será.*

*A mi papá Eliesel y su esposa Mirza por haber sido los que me mostraron la universidad e incentivarme a tomar la decisión de estudiar en la UCI, por ayudarme tanto en todos estos años.*

*Al esposo de mi mamá, Calderón, que se esforzó y se sacrificó mucho para que pudiera lograr mis objetivos.*

*A mis abuelos Kaki y Cirilo por quererme, cuidarme y ayudarme, como si fueran mis padres.*

*A mi abuela "mima" por dejarme ser una de sus nietas favoritas aunque tiene muchas.*

*A todos mis hermanos, Pancho por ayudarme mucho, como pudo, y quererme. Por ser mi hermano mayor aunque parezcas el menor y por regalarme esos sobrinos tan lindos que quiero tanto. A Elie y Elier, mis hermanitos chiquitos, por quererme y ser tan cariñosos.*

*A mi novio querido, Israel, que me ha ayudado y soportado desde los primeros años de universidad y que se ha esforzado en esta investigación tanto como yo, por él he podido llegar aquí con los resultados que tengo los cuales son también de él, muchas gracias mi amor.*

*A la Gordi (Yuleimi) por ser como una hermana para mi y ayudarme en todo lo que me hizo falta.*

*Al Yumbo (Ariel) por ser como otro padre, cariñoso y generoso.*

*A todos los amigos que hice aquí en la universidad, a Gilber.*

*A mi compañera de cuarto y tesis Aimé.*

*A los tutores Susel y David, por habernos ayudado en la realización de este trabajo, guiándonos y apoyándonos en los momentos mas difíciles.*

*Lissette Fonseca Suárez*

# DEDICATORIA

*A mi mami Ángela del Carmen, por quererme y enseñarme todo cuanto sé, por darme las fuerzas para seguir adelante.*

*Para ti este trabajo que significa la culminación de mis estudios y el comienzo de mi vida laboral.*

*Aunque ya no estés, trato de ser mejor cada día para ti.*

*Te quise y te quiero mucho. Te extraño.*

*Aimé Ramírez Benítez*

# DEDICATORIA

*A mi mamá que es la persona más importante para mi, y es la que me ha inspirado para llegar a ser quien soy, todo se lo debo a ella.*

*Y a mi papá por ayudarme tanto, que yo se que me quiere mucho y que le hacia mucha ilusión que yo me graduara.*

*Para los dos este trabajo, que es el resultados de 18 años de estudios y de la educación, el cariño y el esfuerzo de ustedes.*

*Los quiero mucho.*

*Lissette Fonseca Suárez*

# RESUMEN

En los últimos años el desarrollo de software dirigido por modelos se ha convertido en un tema de gran interés a nivel mundial. El metamodelado es una actividad que ha venido aumentando su radio de acción en cuanto a su participación en proyectos informáticos, permitiendo entre otras ventajas, la reducción en tiempo a los desarrolladores y la minimización de la introducción de errores en los programas. El presente trabajo se realiza como continuidad a la investigación *"Metamodelado para la construcción de software en entornos libres"* (Isla y Llanes, 2008), donde se proponen las herramientas de metamodelado AndroMDA y AToM3. El mismo tiene como objetivo validar las herramientas propuestas en proyectos de la Facultad 10, con el fin de demostrar su utilidad en proyectos reales y con ello mejorar el desarrollo del software tanto en la Universidad de las Ciencias Informáticas (UCI) como en el resto del país.

**Palabras Claves:** AndroMDA, AToM3, herramientas de metamodelado.

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

# ÍNDICE

# INTRODUCCIÓN

El desarrollo de software en la actualidad es el tema fundamental por el cual muchos estudiosos de la materia se apasionan. Con el paso de los años el desafío se vuelve cada vez más difícil. Nuevos paradigmas surgen por la necesidad de perfeccionar y ampliar los conocimientos asociados al tema. Uno de ellos es el denominado Desarrollo del Software Dirigido por Modelos (MDSD, por sus siglas en inglés, *Model Driven Software Development*). Dentro de éste se encuentra la Arquitectura Dirigida por Modelos (MDA, por sus siglas en inglés, *Model-Driven Architecture)* muy importante a destacar en este trabajo, ya que constituye una aproximación al desarrollo de sistemas software con calidad; permitiendo elevar el nivel de abstracción, y dándole una mayor importancia al modelado conceptual, al papel de los modelos y las transformaciones entre ellos.

MDSD y MDA cubren un amplio espectro de áreas de investigación, como son la definición de lenguajes de dominios específicos para expresar los modelos que van a capturar cada uno de los aspectos involucrados en el sistema; la definición de lenguajes de transformación entre modelos; la especificación de los metamodelos para dichos lenguajes; la construcción de herramientas para manejar los modelos y las transformaciones entre ellos; la programación a nivel de metamodelos (metamodelado). (DSDM, 2008)

El metamodelado constituye una técnica dentro de este paradigma que juega un papel importante y aporta un gran beneficio al equipo de implementadores. Es un tema fundamental en el campo de las bases de datos y desarrollo de software. Este término nace con la misma filosofía que en su tiempo motivó a desarrollar computadores de propósito general en lugar de seguir con el esquema de construir un ordenador para cada aplicación en particular. Al aplicarlo en el desarrollo de aplicaciones informáticas permite diseñar un modelo de datos (el metamodelo) con la capacidad de almacenar otros modelos de información. Esta tendencia genera importantes ahorros en tiempo y recursos destinados al mantenimiento de software, desarrollo de nuevas aplicaciones y en el desarrollo de nuevas características al software existente, esto es posibilitado por las herramientas de metamodelado, las cuales permiten modelar modelos y sistemas en su conjunto. (Quasar Tecnología, 2007)

# Introducción

El presente trabajo está basado en la investigación *"Metamodelado para la construcción de software en entornos libres"* (Isla y Llanes., 2008) donde se propusieron dos herramientas de metamodelado: AToM3 y AndroMDA. La primera permite la generación rápida de modelos a partir de metamodelos previamente definidos. La segunda, por su parte, fue escogida como propuesta debido a que se considera un motor de transformación de modelos a código fuente, éstos deben ser proporcionados por una herramienta de modelado.

El metamodelado es una alternativa que está cobrando fuerza a la hora de desarrollar software; éste se utiliza en el mundo hace varios años. Sin embargo en el contexto de la Universidad de las Ciencias Informáticas (UCI) aún no se ha empleado. El uso de herramientas de metamodelado en cualquiera de los proyectos de la Universidad posibilitaría al desarrollador abstraerse de los detalles de implementación mediante el trabajo a nivel de modelos y metamodelos, enfocándose directamente en la problemática que el software intenta resolver e independientemente de la plataforma de desarrollo en que se necesite trabajar. Todo esto traería consigo un alto nivel de organización.

Teniendo en cuenta lo analizado anteriormente se determina el siguiente **problema científico**: ¿Cómo demostrar que las herramientas de metamodelado AToM3 y AndroMDA son útiles para el desarrollo de software en los proyectos de la Universidad de las Ciencias Informáticas?

Se define como **objetivo general:** validar las herramientas de metamodelado AToM3 y AndroMDA para el desarrollo de software. La investigación enmarca su **objeto de estudio** en las herramientas de metamodelado y el **campo de acción** está centrado en las herramientas AToM3 y AndroMDA.

Como **idea a defender** se plantea la siguiente: La validación en proyectos reales de herramientas de metamodelado propuestas puede demostrar su utilidad para el desarrollo de software.

Como **tareas de la investigación** se proponen las siguientes:

- Profundización en el estudio de la bibliografía referente al metamodelado.

- Búsqueda de posibles instituciones que utilicen el metamodelado, tanto a nivel nacional como internacional.
- Caracterización de las herramientas AToM3 y AndroMDA.
- Diseño de la validación.
- Validación de las herramientas propuestas.
- Evaluación de los resultados de la validación.

Entre los **métodos científicos teóricos** utilizados en la investigación se encuentran: el **analítico-sintético,** para analizar toda la información relacionada con el metamodelado y determinar los aspectos más importantes asociados al mismo. El **histórico-lógico** fue utilizado para precisar cómo surgieron y evolucionaron términos como metamodelo y metamodelado. Por otra parte, se utilizó el **método empírico** de la **observación** para obtener información primaria acerca de las herramientas investigadas y por tanto como punto de partida para la posterior utilización de la **encuesta** y a partir de ella procesar la información recibida permitiendo arribar a conclusiones.

El documento está estructurado en tres capítulos además de las secciones de Glosario de Términos, Referencias Bibliográficas, Bibliografía y Anexos.

En el **Capítulo I** se realiza un estudio del estado del arte referido al metamodelado, el paradigma de desarrollo MDSD y la arquitectura MDA. Se hace un estudio de las herramientas AToM3, AndroMDA y ArgoUML que exponiendo sus principales características.

En el **Capítulo II** se exponen los elementos fundamentales para la instalación de las herramientas, un análisis de las versiones a utilizar y la propuesta de validación incluyendo los aspectos a tener en cuenta para llevar a cabo la misma.

En el **Capítulo III** se analizan los resultados obtenidos de la validación.

# CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En el presente capítulo, se abordan distintos aspectos que servirán de base para el resto del documento. Se exponen, los conceptos asociados al metamodelado, información referente a MDSD, la arquitectura MDA y los lenguajes de metamodelado. Posteriormente se explican las características de las herramientas de metamodelado centrándose en AToM3 y AndroMDA, también se muestran aspectos significativos de la herramienta de modelado ArgoUML. Se realiza además un estudio sobre la situación actual del metamodelado teniendo en cuenta el mundo, Cuba y la UCI.

## 1.1 Conceptos básicos

A continuación se abordan conceptos y definiciones importantes para la comprensión de toda la investigación.

### 1.1.1 Modelo

Puede considerarse que los campos de la ingeniería desarrollan sus aplicaciones basados en modelos, al igual que muchas acciones de la vida diaria. Los modelos se usan para explicar y controlar fenómenos a nuestro alrededor y pueden predecir eventos que están por ocurrir. Se define un modelo como "*un ente que representa de forma precisa algo que será realizado o que ya existe"*. (Sánchez, 2008)

En general el modelo debe expresar de manera clara y correcta lo que se quiere obtener, sin sobrecargar de información al mismo, éste debe ser completo. Los modelos son utilizados en la ingeniería para varios propósitos, dentro de ellos se puede mencionar que son utilizados para captar y enumerar exhaustivamente los requisitos y el dominio de conocimiento, de forma que todos los implicados puedan entenderlos y estar de acuerdo con ellos, ayudar a pensar el diseño de un sistema, para capturar decisiones del diseño de manera tal que puedan ser cambiados a partir de los requisitos. Otros de sus usos es el de generar productos aprovechables para el trabajo y facilitar el manejo de los sistemas complejos. (Isla y Llanes., 2008)

El modelo es una instancia del metamodelo. Cuando se crea un modelo se está definiendo un lenguaje para describir el área que se está analizando o el sistema que se está diseñando. (González, 1998)

Uno de los elementos más utilizados en el desarrollo de software es el modelo, pues es necesaria una visión diferente del sistema a desarrollar. La construcción de un sistema se

realiza a través de éstos, utilizándolos para describir todas las perspectivas del sistema. Este proceso se conoce como modelado.

El modelado es un ejercicio de abstracción, entendida esta como una simplificación y generalización de la realidad. El uso de la abstracción permite describir, representar, manejar y resolver problemas complejos, pudiendo después aplicar los resultados a casos concretos. Normalmente, se construyen jerarquías de abstracción, en las que cada nivel de abstracción se apoya en los inferiores.

Niveles de abstracción en el modelado (Isla y Llanes., 2008):

0. **Nivel de Información:** agregación informal de datos a manejar en un entorno concreto (aplicación). Se separa un subconjunto de datos con características comunes o interesantes desde un determinado punto de vista.
1. **Nivel de Modelo:** agregación informal de metadatos (datos sobre los datos) que describen una información concreta. Se describen las características comunes de los datos dando lugar a metadatos que se agrupan, describiéndose las relaciones entre ellos, para formar un modelo.
2. **Nivel de Metamodelo:** agregación de modelos o meta-metadatos (descripción de metadatos). Descripciones que definen la estructura y la semántica de los metadatos. Se describen las características comunes de subconjuntos de metadatos dando lugar a meta-metadatos o modelos que se agrupan, describiéndose las relaciones entre ellos, para formar un metamodelo.
3. **Nivel de Meta-metamodelo:** definición de la estructura y la semántica de los meta-metadatos. Se capturan las características comunes de subconjuntos de modelos dando lugar a metamodelos que se agrupan, describiéndose las relaciones entre ellos, para formar un meta-metamodelo.

### 1.1.2 Metamodelo

Según (Quasar Tecnología, 2007), "*el metamodelo es un modelo con la capacidad de almacenar diferentes modelos"*. Partiendo de esta definición se puede pensar en la posibilidad de construir un producto de software con la capacidad para registrar información en el metamodelo y otro producto con la capacidad de realizar consultas sobre el mismo. El metamodelo está dividido en dos áreas:

- ✓ Área de Metadatos

Esta área almacena el metamodelo en sí y en un conjunto de tablas relacionales se guarda una descripción detallada del modelo de los datos a ser almacenados en el área de datos. Este modelo parte del principio que toda información para ser registrada en un sistema de información debe ser estructurada en uno o varios formatos.

- ✓ Área de Datos

Esta área almacena la información acorde a la estructura definida en el área de metadatos. Es aquí donde se registra la información proveniente del mundo real, estructurada de acuerdo a los formatos que forman parte del modelo almacenado en el área de metadatos.

El metamodelo es la capa donde se define el lenguaje que sirve para especificar los modelos que serán creados. En otras palabras, sirve para describir los elementos que van a componer los diagramas. (González, 1998)

Se puede considerar que un metamodelo es una definición precisa de las reglas necesarias para definir la semántica de los modelos. Otra definición a tener en cuenta es la que se plantea en (Reina, Torres y Toro, 2006) en la que se considera que el metamodelado puede verse como una actividad que está tomando auge en los últimos años y que sirve para organizar los modelos en diferentes niveles, de tal modo que un modelo se describe por otro modelo que está situado en un nivel superior, su metamodelo.

Finalmente las autoras definen un metamodelo como un modelo que almacena información referente a otros modelos, describiendo los elementos que conforman los mismos.

### 1.1.3 Metamodelado

En (De Lara, Vangheluwe y Alfonseca, 2003) se define el metamodelado como el "*proceso de modelado de formalismos*", el cual se suele realizar para determinar si una instancia de un modelo particular es consistente con su especificación en forma de metamodelo. La técnica de metamodelado permite definir lenguajes de modelado, a través de un metamodelo. De acuerdo con lo antes planteado se puede resumir que el metamodelado no es más que una técnica que permite la creación de lenguajes de modelado, mediante la realización de metamodelos, ya que éstos como bien se dice anteriormente, no son más que un modelo de un lenguaje de modelado.

El metamodelado se comprende a partir de una estructura de cuatro niveles M0, M1, M2, M3 (Isla y Llanes, 2008), los cuales se describen en la Tabla 1 que se explica a continuación. En esta estructura, el nivel M3 establece las primitivas de metamodelado, esto es, el lenguaje de metamodelado. En el nivel M2, se definen los metamodelos en sí con las primitivas de M3. En M1 se aplican los metamodelos para generar instancias, que se denominan modelos. Por último, en M0 se realiza la instanciación de los modelos donde se tiene la información que se manejará en el sistema.

| Nivel | Descripción |
|---|---|
| **M3 (Meta-metamodelo)** | Define un lenguaje para especificar metamodelos. |
| **M2 (Metamodelo)** | Define un lenguaje para especificar modelos<br>Cada elemento es una instancia del meta-metamodelo. |
| **M1 (Modelo)** | Cada elemento es una instancia de un metamodelo. |
| **M0 (Instancia)** | Instancias de elementos definidos en un modelo. |

**Tabla 1. Arquitectura en cuatro niveles.**

### 1.1.4 Meta-metamodelo

Un meta-metamodelo está definido como una especificación de la estructura de un lenguaje que sirve para crear lenguajes de modelado. La idea detrás de esto es bastante simple: proveer de un lenguaje que permita a cada grupo de desarrolladores definir sus propios lenguajes de modelado.

Teniendo en cuenta la estructura de cuatro niveles M0, M1, M2, M3 en la cual está comprendido el metamodelado, se plantea que el meta-metamodelo, definido en la capa del metamodelo M3, es un metamodelo que describe el contenido de los metamodelos, es decir, los tipos de entidades que son compartidas a través de los diferentes sistemas de información. Como ejemplo de meta-metamodelo se tiene Meta Object Facility (MOF),

estándar creado por la Object Management Group (OMG) que extiende UML (Unified Modeling Language) para que éste sea aplicado en el modelado de diferentes sistemas de información. El mismo conceptualiza diversos metamodelos, esencialmente abstrayendo la forma y la estructura que describen estos. Define los elementos esenciales, sintaxis y estructuras de metamodelos que son utilizados para construir modelos orientados a objetos de sistemas.

## 1.2 Desarrollo del Software Dirigido por Modelos y Arquitectura Dirigida por Modelos

El Desarrollo de Software Dirigido por Modelos es una propuesta para el desarrollo de software en la que se les atribuye a los modelos el papel principal de todo el proceso, frente a las propuestas tradicionales basadas en lenguajes de programación y plataformas de objetos y componentes software. MDSD persigue elevar el nivel de abstracción en el desarrollo de software, convirtiendo a los modelos y a las transformaciones entre ellos en los principales artefactos de todas las fases del proceso de desarrollo de software: captura y gestión de los requisitos, diseño, análisis, implementación, despliegue, configuración, mantenimiento y evolución. (Estévez, Pelechano y Vallecillo, 2007)

El éxito de esta iniciativa ha originado la evolución de distintos paradigmas englobados dentro del contexto del MDSD, como por ejemplo la iniciativa MDA de la OMG, las técnicas de Agile Model-Driven Development (AMDD), las propuestas de Domain Specific Modeling (DSM), las estrategias de Domain-Oriented Programming (DOP) o las Fábricas de Software (SF, por sus siglas en inglés, *Software Factories*). Cada una de ellas aborda el proceso de MDSD de diferente forma, y con distintos mecanismos. (Estévez, Pelechano y Vallecillo, 2007)

En particular, la iniciativa MDA ha cobrado fuerza en el desarrollo de software, donde la idea central es definir la estructura y comportamiento del sistema utilizando lenguajes de modelado, para luego, utilizando herramientas de software especializadas, transformar dichos modelos en una implementación en el lenguaje de programación requerido como se plantea en (Tasof, 2005).

La arquitectura dirigida por modelos es un acercamiento al diseño de software. MDA se ha concebido para dar soporte a la ingeniería dirigida a modelos de los sistemas software, es una arquitectura que proporciona un conjunto de guías para estructurar especificaciones expresadas como modelos. La misma está relacionada con múltiples

normas, incluyendo UML, MOF, Extensible Markup Language (XML)[1], Metadata Interchange (XMI)[2], Enterprise Distributed Object Computing (EDOC), el Software Process Engineering Metamodel (SPEM) y el Common Warehouse Metamodel (CWM).

Uno de los principales objetivos de MDA es separar el diseño de la arquitectura y de las tecnologías de construcción, facilitando que el diseño y la arquitectura puedan ser alterados independientemente. El diseño alberga los requerimientos funcionales, mientras que la arquitectura proporciona la infraestructura a través de la cual se hacen efectivos requerimientos no funcionales como la escalabilidad, fiabilidad o rendimiento.

## 1.3 Lenguajes de metamodelado

La mayoría de los informáticos, una vez que empiezan a trabajar en proyectos reales, suponen que código y programación son la única y exclusiva forma de desarrollar software.

Los lenguajes de programación de alto nivel son los que están en auge actualmente para desarrollar software, sin embargo, el nivel de complejidad que las plataformas han alcanzado hoy en día, unido a la necesidad de adaptar el software a requerimientos en constante cambio, han creado la necesidad de simplificar aún más el desarrollo. Ahora es necesario aumentar el nivel de abstracción, ocultar más los detalles de implementación, para así poder reducir la complejidad que el ingeniero de software debe enfrentar cuando va a crear un sistema. Es por ello que nacen los lenguajes de modelado, notaciones en su mayoría visuales, que intentan representar un sistema de software a un nivel mucho más alto que los lenguajes de programación, representándolo en forma más intuitiva para personas sin especialización en informática. (Tasof, 2005)

Los lenguajes de metamodelado más extendidos son MOF de OMG, Ecore y el lenguaje de modelado UML 2.0, los cuales serán descritos a continuación.

### - MOF

Meta Object Facility es el lenguaje de metamodelado propuesto por el Object Management Group. Es utilizado para crear metamodelos (por ejemplo, el metamodelo de UML ha sido definido con MOF), y es, por tanto, un elemento básico de MDA. Permite

[1] XML: es un metalenguaje para la definición de documentos estructurados mediante marcas o etiquetas.

[2] XMI: XML de Intercambio de metadatos.

expresar metadatos (igual que XML), es independiente de la plataforma, y está descrito con la notación UML y Object Constraint Language (OCL)[3]. Cada elemento del lenguaje se representa mediante una clase y sus propiedades como atributos. Las relaciones entre elementos se representan como asociaciones. Incluye la generalización permitiendo expresar que un elemento es una especialización de otro. Además, MOF usa paquetes si el metamodelo que se va a desarrollar es muy grande. (García, 2007)

En el 2006 Object Management Group publicó dos variantes de MOF:

- ✓ Essential MOF(EMOF)
- ✓ Complete MOF(CMOF)

### - Ecore

Ecore es un lenguaje común basado en EMOF que es parte de la especificación MOF. Ecore es usado por Eclipse Modeling Framework (EMF) para la definición de metamodelos. Los metamodelos y modelos usados por EMF se representan con documentos XML. (Lee, Leung y Son, 2007)

### - UML 2.0

UML es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. La expresividad de UML no es suficiente para cubrir todo tipo de situaciones. Nacen así extensiones a UML para modelar otros elementos, tales como persistencia, seguridad e interfaces gráficas. Ante la proliferación de tantas extensiones, y la carencia de un estándar en que dichas extensiones sean especificadas, surge la necesidad de definir claramente un Metamodelo para UML. Aunque desde un principio la especificación de UML contenía un metamodelo, éste era bastante ambiguo en cuanto a extender sus capacidades. Después de varios años de desarrollo apareció un nuevo estándar para UML, la versión 2.0, la cual incluye no sólo un nuevo metamodelo para el lenguaje de modelado en sí, sino que además incluye un meta-metamodelo. La idea detrás de esto es proveer un lenguaje que permita a cada grupo de desarrolladores definir sus propios lenguajes de modelado.

[3] OCL: es un lenguaje declarativo para la descripción de reglas de UML.

Con el uso de leguajes de metamodelado se pueden definir Lenguajes Específicos del Dominio (DSL, por sus siglas en inglés), los cuales al estar restringidos a un dominio particular, ofrecen técnicas más potentes de análisis y generación de código, aumentando la productividad y mejorando la calidad de los productos generados (De Lara y Pérez, 2006).

### 1.3.1 Lenguajes Específicos del Dominio

El estudio de los DSL resulta de gran importancia para profundizar los conocimientos sobre metamodelado pues un DSL cuenta con una sintaxis textual y una sintaxis abstracta, las cuales se definen a través de metamodelos construidos en algún lenguaje de metamodelado.

Los DSL son lenguajes de programación especialmente diseñados para desarrollar software restringido a un dominio determinado. A diferencia de los lenguajes llamados de propósito general que son orientados al desarrollo en ciertos ámbitos como Java, C++ o C#, los DSL cuentan con un universo limitado de aplicación. No obstante, gracias a esta especialización, presentan facilidades y ventajas a la hora de abordar los problemas de software para los que fueron diseñados y desarrollados, es decir, pueden ayudar porque al enfocarse en un área bien delimitada se vuelven especialistas del campo y ahorran tiempo de codificación de software. (Perdiguero, 2009)

A lo largo de esta década se ha revitalizado el interés por los DSL con el surgimiento del MDSD en especial con MDA. Se debe tener en cuenta que este concepto no es algo nuevo, ya en los años 50 se idearon DSL para programar aplicaciones en máquinas controladas numéricamente. Muchas de las notaciones y lenguajes ampliamente utilizados hoy día tal como se plantea en (Gómez y Sánchez, 2006) pueden considerarse lenguajes específicos del dominio. Algunos de ellos bastante conocidos son: EXCEL, HTML, LATEX, MATLAB, SQL, VHDL.

Un ejemplo de código de Nokia para crear software de teléfonos móviles, expresado con un DSL, se muestra en la Figura 1.

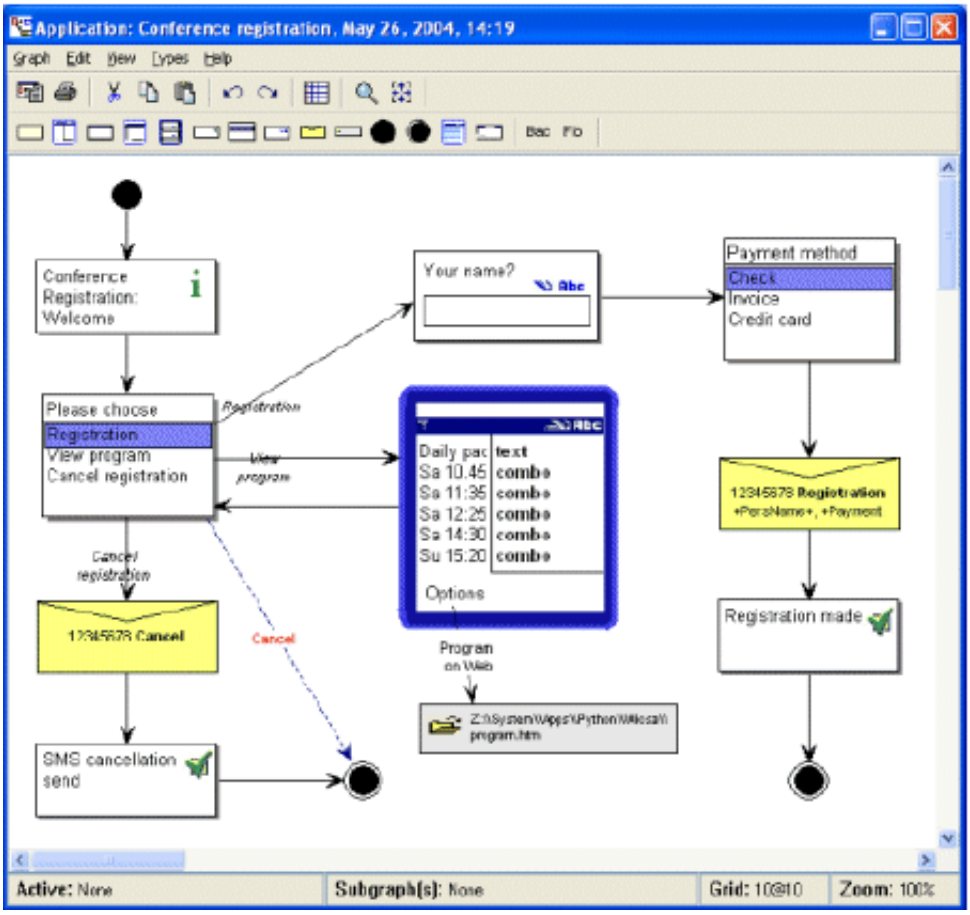



**Figura 1. Ejemplo de DSL**

## 1.4 Herramientas CASE, MetaCASE y MDA

Las herramientas AToM3, AndroMDA y ArgoUML se encuentran dentro de alguno de los tipos de herramientas que se describen a continuación.

### 1.4.1 CASE

Las herramientas CASE (*Computer Aided Software Engineering*) son herramientas diseñadas para dar soporte al programador a través de una interfaz intuitiva y gráfica que le permita, mediante el posicionamiento de objetos gráficos concretos, relacionados entre sí siguiendo una nomenclatura específica, desarrollar la base de un programa automáticamente. En (Bombadil, 2009) se plantea que la mayoría de herramientas CASE se basan en el uso de diagramas UML que permitan especificar una organización del código, un comportamiento, con definición de casos de uso y secuencias, para después autogenerar la estructura o esqueleto base y escribir el resto del código, dentro de los métodos que quedasen vacíos después de la autogeneración.

A partir de la consolidación de las metodologías de desarrollo, integrando diferentes técnicas, comenzaron a aparecer las primeras herramientas CASE.

Se puede definir a las herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de un software. En (Tuesta, 2005) CASE se define también como un concepto avanzado en la evolución de tecnología, con un potencial efectivo y profundo en las organizaciones. Se le puede ver como la unión de las herramientas automáticas de software y las metodologías de desarrollo de software formales.

Estas herramientas pueden proveer muchos beneficios en todas las etapas del proceso de desarrollo de software según (Nazareno, 1999), algunas de ellas son:

- ✓ Verificar el uso de todos los elementos en el sistema diseñado.
- ✓ Automatizar el dibujo de diagramas.
- ✓ Ayudar en la documentación del sistema.
- ✓ Ayudar en la creación de relaciones en la Base de Datos.
- ✓ Generar estructuras de código

La principal ventaja de la utilización de una herramienta CASE, es la mejora de la calidad de los desarrollos realizados y, en segundo término, el aumento de la productividad. Sin embargo tiene varias limitaciones como son:

- ✓ No generan código fuente en sí, sino plantillas de código fuente que el desarrollador debe completar para obtener algo que sea similar a un código compilable o ejecutable.
- ✓ Cualquier cambio en las especificaciones requiere que se actualice la herramienta CASE completa.
- ✓ Las herramientas CASE que permiten la generación de código a partir de los modelos construidos en ellas, poseen métodos de traducción muy rígidos, que impiden tomar decisiones de diseño en la traducción.

Las herramientas CASE más usadas en el entorno laboral (Bombadil, 2009), disponibles de forma libre (algunas no gratuitas) son las siguientes:

- ✓ Visual Paradigm
- ✓ ArgoUML

- ✓ Innovator

Las herramientas CASE han evolucionado en otras más flexibles denominadas herramientas MetaCASE.

### 1.4.2 MetaCASE

En (Anaya de Páez, Arango y Zapata, 2007) se plantea que las herramientas MetaCASE surgieron como una evolución de la tecnología CASE, para superar algunas de las limitaciones identificables en este tipo de tecnologías. Estas herramientas poseen mecanismos que permiten la creación y modificación de paradigmas de metamodelado, además de la adición de restricciones que se pueden emplear en chequeos de asistencia o transformaciones de refinamiento. Como ejemplo de herramienta MetaCASE se puede citar a AToM3.

### 1.4.3 MDA

MDA, propuesto por el OMG, está tomando cada vez más fuerza en el desarrollo de software. Se trata de un marco de trabajo para el desarrollo de software, cuya principal característica es la definición de modelos como elementos de primer orden en el diseño, desarrollo e implementación del software y las transformaciones entre los diferentes modelos. En los últimos años han aparecido numerosas herramientas MDA (como por ejemplo AndroMDA) que permiten en mayor o menor grado automatizar las transformaciones y que generan, en algunos casos, semi-automáticamente código para distintas plataformas.

En (Lopez, Gonzales y Moisés, 2007) se plantea que la idea clave de MDA es que si el desarrollo está guiado por modelos de software, se obtendrán importantes beneficios en aspectos fundamentales como:

- ✓ Productividad.
- ✓ Portabilidad.
- ✓ Interoperabilidad.
- ✓ Mantenimiento y documentación.

## 1.5 Herramientas de metamodelado

“Las herramientas de metamodelado son las herramientas básicas para diseñar el metamodelo de acuerdo al lenguaje de metamodelado”, tal como se plantea en (Gómez y

Sánchez, 2006). Según (Quasar Tecnología, 2007) la aplicación de herramientas de metamodelado en una entidad o proyecto puede traer consigo varias ventajas, entre las que se encuentran:

- ✓ Reducción en tiempo y recursos para el mantenimiento de las aplicaciones existentes.

  Todo desarrollador de software sabe que los productos que realizan, por representar modelos de lo que ocurre en el mundo real, son cambiantes y dinámicos, por lo que deben ser adaptables a las nuevas exigencias de los clientes o beneficiarios institucionales, esto conlleva a realizar un proceso que involucra grandes cambios. La aplicación del metamodelado, implica que los pasos a seguir para realizar la misma modificación sean menores. Este cambio en el paradigma del mantenimiento de las aplicaciones genera sustanciales beneficios a la organización que toma la decisión de adoptar esta tecnología para la construcción y mantenimiento de sus sistemas de información.

- ✓ Evita la introducción de errores en los programas.

  La capacidad de introducir una nueva funcionalidad en un sistema de información sin escribir líneas de código adicional, elimina la posibilidad de introducir errores de programación cuyo costo tanto para la información registrada (errores en la base de datos a causa de un programa erróneo) o el costo de ubicar, corregir y probar el código fuente causante del error, son eliminados.

- ✓ Reducción en el tiempo de entrenamiento a los usuarios.

  Luego de entrenar un usuario en la utilización de estas herramientas, el entrenamiento para que aprenda a utilizar la aplicación del metamodelo para otras aplicaciones, se reduce a trabajar con éste sobre los formatos, grupos y campos de información con los cuales se ha estructurado la información en el nuevo contexto. Esto gracias a la unicidad de las interfaces en el software de registro y consulta de información.

- ✓ Generación de consultas a la medida.

  El sistema generador de consultas para un metamodelo, se convierte en una herramienta muy poderosa en manos de las personas que dominan el contexto temático de la información registrada en este metamodelo, ya que puede

consultar, ordenar, agrupar, graficar o producir información alfanumérica para la información registrada.

### 1.5.1 Herramientas AToM3 y AndroMDA

Seguidamente se explicarán las herramientas en las cuales está centrado el objeto de la investigación.

#### 1.5.1.1 AToM3

A Tool for Multi-Formalism Modelling and Meta-Modelling (AToM3) es una herramienta MetaCASE escrita en el lenguaje de programación Python, y está enfocada a los flujos de trabajo de Análisis y Diseño. Posee un procesador que incluye un meta-metamodelo inicial basado en el modelo entidad-relación, que permite la definición de los diferentes metamodelos en un entorno gráfico con las mismas características que emplea el usuario en la construcción de los diferentes modelos. De esta manera, se puede definir cualquier tipo de metamodelo en términos de las entidades que forman parte del mismo y sus posibles interconexiones o relaciones. Una vez definido el metamodelo, se puede emplear su definición para construir los modelos pertinentes a un problema específico del mundo. (Zapata y Álvarez, 2005)

AToM3 tiene la posibilidad de expresar restricciones en términos de gramáticas de grafos incorporadas a su entorno. Las gramáticas de grafos tienen similitudes con las gramáticas basadas en texto en el sentido de que pueden ser usadas para describir las transformaciones a un grafo determinado; la diferencia radica en que las reglas de ese tipo de gramáticas se expresan de manera gráfica y no a modo de texto.

Las gramáticas de grafos se definen como un conjunto de reglas que poseen un lado izquierdo (LHS, por sus siglas en inglés, *left-hand side*) que contiene las precondiciones (expresadas de forma gráfica) que deben ser cumplidas para activar una determinada regla y un lado derecho (RHS, por sus siglas en inglés, *right-hand side*) que contiene el grafo que remplazará el que equivale al lado izquierdo de la regla. Para las reglas expresadas de esta manera se deben definir condiciones y acciones para ejecutar cuando la regla se active. La gramática de grafos de AToM3 posee también un mecanismo que va reescribiendo el modelo a medida que las diferentes reglas se van activando hasta que no haya reglas que se puedan ejecutar. (Zapata y Álvarez, 2005)

¿Cómo funciona AToM3?

AToM3 trae en su instalación algunos metamodelos que definen el comportamiento de algunos de los principales diagramas UML, como son, diagramas de clases, entidad-relación, entre otros. A partir de estos metamodelos la herramienta le brinda al usuario la posibilidad de modelar sus diagramas. En caso de que el usuario necesite modelar un diagrama del cual no se tenga el formalismo definido, entonces la aplicación permite la creación del metamodelo, mediante un diagrama entidad-relación y un conjunto de restricciones que definen su comportamiento.

Según (De Lara, 2005) la idea principal de la herramienta es: "*todo es un modelo*", en el sentido de que incluso la interfaz de usuario de AToM3 es un modelo, que se interpreta al cargarse, y que por tanto puede modificarse. (Ver Figura 2)

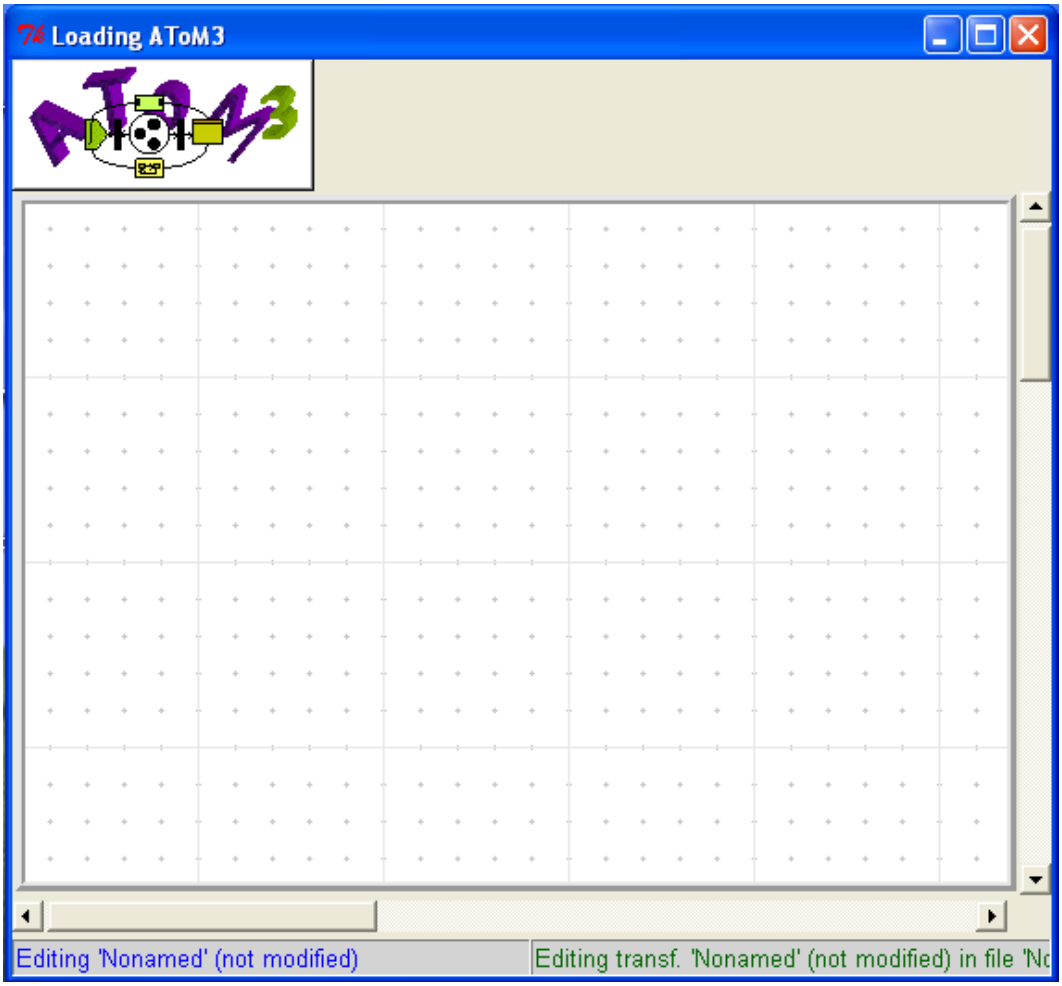



**Figura 2. Interfaz de usuario de AToM3.**

#### 1.5.1.2 AndroMDA

AndroMDA (pronunciado “Andrómeda”) es una herramienta de metamodelado escrita en el lenguaje Java bajo la licencia *Berkeley Software Distribution* (BSD), que está enfocada a todo el proceso de desarrollo de software. Es un programa informático de tipo framework. Se considera un motor de transformación de modelos en código fuente. AndroMDA convierte modelos de algunas herramientas de modelado UML en componentes listos para su despliegue en un gran número de lenguajes entre los que se encuentran Java, PHP, .NET, HTML, sólo con utilizar los cartuchos[4] adecuados. Estos cartuchos dirigen el desarrollo de aplicaciones basadas en componentes y son fundamentales para el proceso de generación de código. La herramienta permite la creación de mecanismos para la construcción de nuevos cartuchos. (AndroMDA, 2006)

¿Cómo funciona AndroMDA?

AndroMDA lee el modelo y coloca los objetos del modelo en memoria para que los mismos estén disponibles para sus plugins. Éstos definen exactamente lo que AndroMDA va o no a generar. Cada uno de estos es totalmente adaptable a las necesidades del proyecto.

Según (AndroMDA, 2006) esta herramienta ofrece ciertas garantías a los desarrolladores como pueden ser:

- ✓ Pertenecer a una comunidad: un desarrollador siempre tiene apoyo de la comunidad AndroMDA para resolver dudas, además de participar en el proceso de desarrollo de nuevas funcionalidades, a través del envío de parches o exigiendo nuevas funcionalidades.
- ✓ Visión: algunos desarrolladores propios de AndroMDA mantienen un control estricto de su desarrollo, pues para ellos es importante que el producto no se vea lleno de funcionalidades que nadie vaya a usar, además de encargarse que las nuevas estén completamente documentadas.
- ✓ Abierto: nunca se tendrá que pagar por usar AndroMDA, pues uno de los aspectos más importantes de la herramienta es que está hecha para ayudar a la comunidad de desarrolladores a realizar buenas implantaciones de Software.

[4] Cartuchos: Representan módulos de generación de código para la herramienta AndroMDA.

- ✓ Modular: AndroMDA es un micro-kernel con plugins para distintos tipos de componentes, AndroMDA mantiene la implementación por defecto de cada uno de los módulos, sin embargo estos pueden ser adaptados dependiendo de las necesidades del usuario.
- ✓ Documentación: la documentación de todas las versiones del producto siempre están actualizadas, pues refleja el estado del producto.

Para poder iniciar AndroMDA es necesaria una herramienta adicional que sea capaz de construir los proyectos de ésta. Existen varias herramientas que se adaptan al entorno de desarrollo de AndroMDA, entre las más utilizadas se encuentran Ant y Maven. A continuación se exponen las principales características de éstas:

**Ant**[5] o **Apache Ant,** como también se le conoce, es una herramienta usada en programación para la realización de tareas mecánicas y repetitivas, normalmente durante las etapas de compilación y construcción. Esta herramienta, hecha en Java, tiene la ventaja de no depender de las órdenes del intérprete de comandos de cada sistema operativo, sino que se basa en archivos de configuración XML y clases Java para la realización de las distintas tareas, siendo idónea como solución multi-plataforma, además es un proyecto de código abierto.

**Maven**[6] es una herramienta de software para la gestión y construcción de proyectos Java muy similar en funcionalidad a Apache Ant, pero tiene un modelo de configuración de construcción más simple, basado en un formato XML.

Es la herramienta recomendada para la construcción y el despliegue de las aplicaciones generadas por AndroMDA, usa un Project Object Model (POM), fichero XML para describir el proyecto de software a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos. Realiza tareas como la compilación del código y su empaquetado.

Una característica clave de Maven es que se usa en red. El motor incluido en su núcleo puede descargar dinámicamente plugins de un repositorio, el mismo repositorio que provee acceso a muchas versiones de diferentes proyectos de código abierto en Java. Maven provee soporte no sólo para obtener archivos de su repositorio, sino también para

5 http://ant.apache.org/

6 http://maven.apache.org/

subir artefactos al repositorio al final de la construcción de la aplicación, dejándola al acceso de todos los usuarios.

De estas dos herramientas, como se menciona anteriormente es Maven la que mejor construye los proyectos de AndroMDA, debido a que cada versión de AndroMDA trae consigo una serie de plugins para Maven que le sirven a éste para construir dichos proyectos, por lo que en la presente investigación se utiliza Maven para la construcción y gestión de los proyectos de AndroMDA.

## 1.6 Herramienta de modelado ArgoUML

AndroMDA utiliza los modelos realizados por alguna herramienta de modelado UML para generar código. ArgoUML es una de las herramientas que puede integrarse perfectamente con AndroMDA.

ArgoUML es una herramienta libre de modelado escrita en Java, publicada bajo la licencia BSD open source y que incluye soporte para diagramas bajo estándar 1.4 de UML. Incluye una interfaz muy intuitiva, estable y de sencillo manejo, tomado de (Ubuntu, 2007).

Esta herramienta está basada en estándares abiertos como son : XMI, SVG y PGML, está disponible en ocho idiomas, dirigida al apoyo de los flujos de trabajo de modelamiento del negocio, requerimientos, análisis, diseño e implementación, genera esqueletos de código java relativos a los diagramas representados, soporta XMI, OCL y bases de datos. Además exporta diagramas como gráficos y posibilita la extensión a otros lenguajes.

Ventajas:

- ✓ Código abierto y licencia libre.
- ✓ Facilidad de uso.
- ✓ Independencia de la plataforma.

La interfaz de usuario de ArgoUML se divide en 4 paneles (ver Figura 3), tomado de (ArgoUML, 2008)

- ✓ Parte superior izquierda: una vista jerárquica del archivo actual de proyecto.
- ✓ Parte superior derecha: editor de la parte seleccionada del proyecto, en este caso un diagrama de clases.
- ✓ Parte inferior izquierda: el diseñador de la lista de “cosas por hacer”.

- ✓ Parte inferior derecha: los detalles del objeto seleccionado en el diagrama o en la lista de “cosas por hacer”.

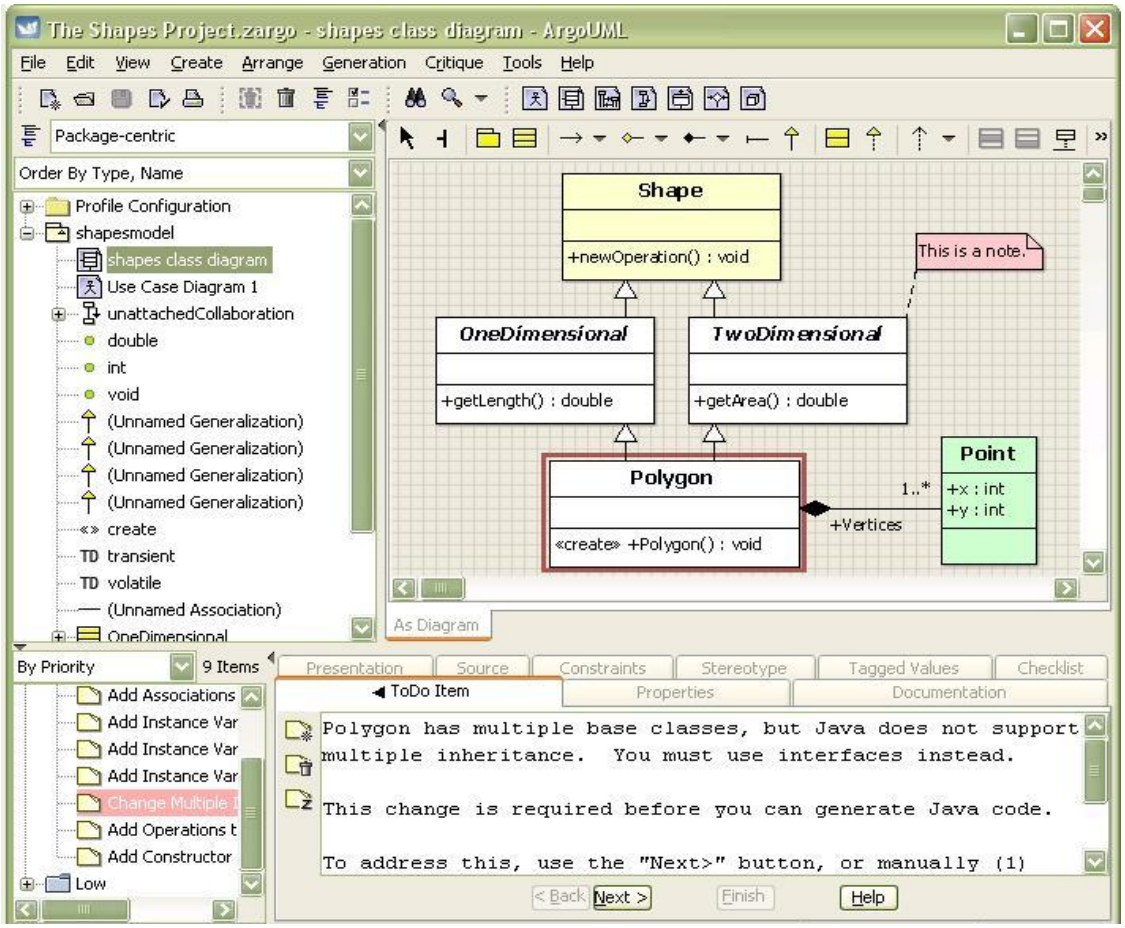



**Figura 3. Interfaz de usuario de ArgoUML.**

## 1.7 Resumen de las herramientas propuestas

En la Tabla 2 se muestra un resumen de las principales características de las herramientas propuestas según los siguientes criterios de comparación:

- ✓ Plataforma: se verificará en cuales plataformas puede ser instalada la herramienta.
- ✓ Licencia: aquí se puede contemplar bajo que licencia se distribuye la herramienta.
- ✓ Usabilidad: este criterio permitirá saber si la herramienta es fácil o difícil de usar.
- ✓ Fases en las que se enfoca: se apreciará hacia que parte del proceso de desarrollo de software está enfocada la herramienta.
- ✓ Grado de interacción con el usuario: aquí se determinará si la interacción de la herramienta con el usuario es baja, media o alta.

- ✓ Generación de código: en este criterio se plasmarán los lenguajes que en mayor medida son generados por la herramienta.
- ✓ Lenguajes de almacenamiento y gestión de modelos: se presentará que lenguaje es utilizado por la herramienta para el almacenamiento y la gestión de modelos.
- ✓ Ámbito de aplicación: en este criterio se puede contemplar en que ámbitos es aplicada la herramienta, es decir, a que se enfoca su desarrollo.
- ✓ Interoperabilidad: aquí se apreciará si se puede o no integrar con otras herramientas

| **Criterios de comparación** | **AToM3** | **AndroMDA** | **ArgoUML** |
|---|---|---|---|
| **Plataforma** | Multiplataforma | Multiplataforma | Multiplataforma |
| **Licencia** | GPL | BSD | BSD |
| **Usabilidad** | Difícil de usar | Difícil de usar | Fácil de usar |
| **Flujos en los que se enfoca** | Análisis y diseño | Negocio, requerimiento, análisis, diseño e implementación | Negocio, requerimiento, análisis, diseño e implementación |
| **Grado de interacción con el usuario** | Alto | Alto | Alto |
| **Generación de código** | Python | Cualquier lenguaje (Según los cartuchos) | Java, C++, C#, PHP4, PHP5 |
| **Lenguajes de** | | | |

| | | | |
|---|---|---|---|
| **almacenamiento y gestión de modelos** | GML | XMI | XMI |
| **Ámbito de aplicación** | Desarrollo orientado a servicios | Desarrollo orientado a servicios | Desarrollo orientado a servicios |
| **Interoperabilidad** | Sí | Sí | Sí |

**Tabla 2: Tabla resumen de las herramientas propuestas.**

## 1.8 Situación actual del metamodelado

En la actualidad es abundante la información que se puede obtener respecto a las herramientas de metamodelado. Con el paso del tiempo y el auge que ha adquirido el desarrollo de sistemas de software se hace necesario el uso de herramientas que ayuden al mejoramiento y la comprensión de sistemas extremadamente complejos.

El metamodelado es un tema aún joven pero por el avance que significa es bastante utilizado en el mundo por varios países como Uruguay, Colombia y Estados Unidos, se tiene conocimiento que uno de los países que más ha investigado y trabajado con el tema es España, donde se emplea para muchas aplicaciones complejas que tienen un estrecho vínculo con la Inteligencia Artificial (IA), como por ejemplo, en la creación de redes sensoriales inalámbricas, en el desarrollo de Sistemas Multi-Agente, en la evaluación de Calidad en Procesos de Desarrollo Web, en la realización de simulaciones discretas con un enfoque educativo (De Lara, 2005), entre otras.

En nuestro país, el uso de herramientas de metamodelado es prácticamente nulo, no se tiene conocimiento de que algún proyecto haya usado o esté usando metamodelado, solo se ha llevado este tema a la fase de investigación en trabajos de diplomas, de pos-grado y de doctorado. Un ejemplo de esto es el trabajo que realizó el cubano Emilio Soler en trabajo conjunto con tres españoles Juan Trujillo, Eduardo Fernández y Mario Piattini, “Una extensión del metamodelo relacional de CWM para representar Almacenes de Datos Seguros a nivel lógico”. (Soler, Trujillo y Fernández, 2007)

Con el surgimiento de la UCI, Universidad donde se forma al profesional desde la producción, se produce un cambio en la Industria Cubana de Software (INCUSOFT) mediante proyectos para informatizar la propia Universidad, el país y la exportación de los

mismos. En la UCI, hasta ahora el metamodelado no forma parte de los planes de estudio de pregrado o postgrado. Sin embargo, en el ya mencionado Trabajo de Diploma *"Metamodelado para la construcción de software en entornos libres",* que sirvió de base a la presente investigación, se realizó un amplio estudio del tema, quedando formalizado el conocimiento relativo a esta área, lo cual debe marcar el inicio y servir de base para el estudio de este tema tan importante.

Por otro lado, y por los mismos motivos expuestos anteriormente, las herramientas para metamodelado no se han utilizado en la UCI. En dicho trabajo, ver (Isla y Llanes, 2008) se analiza un grupo de herramientas, de las cuales se propuso utilizar, por ser las más aceptadas y utilizadas, las herramientas que son validadas en el presente trabajo, en el cual además se propone agregar ArgoUML a la propuesta inicial.

### 1.8.1 Usos de las herramientas

**- AToM3**

Existen numerosos trabajos realizados con esta herramienta, un ejemplo es la "Conversión de diagramas de procesos en diagramas de casos de uso" (Zapata y Álvarez, 2005) en el cual se emplea AToM3 para la definición de los metamodelos del diagrama de procesos y el diagrama de casos de uso, con el fin de reexpresar el primero para obtener algunos elementos básicos del segundo.

En el artículo "Simulación Educativa Mediante Meta-Modelado y Gramáticas de Grafos" (De Lara, 2005) se propone por medio del metamodelado, usando AToM3 describir de forma gráfica formalismos de simulación para usarlos con fines educativos.

Otro proyecto desarrollado utilizando esta herramienta es la "Conversión de esquemas preconceptuales a diagramas de casos de uso" (Zapata, Tamayo y Arango, 2007).

**- AndroMDA**

Existen muchos desarrolladores de software y equipos de trabajo que utilizan AndroMDA para desarrollar sus proyectos entre los que se encuentra SAPO-XP, el cual emplea dicha herramienta por su característica de generar código dada la entrada del modelo de diseño, diagramas de actividades, entre otros modelos. En un estudio realizado, (SAPO-XP, 2008) se evidencia claramente como la herramienta AndroMDA ayudó a la disminución del esfuerzo de escribir código, ya que lo generado por el equipo de trabajo representa el 11% del total de líneas de código implementadas hasta el momento.

Otro de los productos desarrollados utilizando esta herramienta es un Modelo de Costos Basado en Actividades (Navia, 2008).

Entre las compañías que utilizan esta herramienta para desarrollar sus productos, se pueden citar AutoMagic, Kombi Verkehr y Lufthansa Systems. (Ver **¡Error! No se encuentra el origen de la referencia.**5)


**Figura 4. Compañías que usan AndroMDA.**

**- ArgoUML**

Existen empresas que están comercializando productos o servicios basados en o relacionados con esta herramienta, entre ellas se encuentra Gentleware que desarrolló Poseidon para UML y Genuitec que hizo MyEclipse UML, productos que fueron originalmente basados en ArgoUML. Ver **¡Error! No se encuentra el origen de la referencia.**6.


**Figura 5. Compañías que usan ArgoUML.**

En este capítulo se han expuesto conceptos fundamentales para la investigación. Se han caracterizado las herramientas AToM3, AndroMDA y ArgoUML. A partir del estudio y análisis de los temas asociados al metamodelado se puede llegar a la conclusión de que esta es una actividad que ha tomado auge en los últimos años en el ámbito del desarrollo de software. El uso de las herramientas de metamodelado puede traer consigo grandes beneficios en cuanto al ahorro de tiempo por el código generado, por lo que sería de gran importancia la validación de las mismas para su futura utilización.

# CAPÍTULO 2: PROPUESTA DE VALIDACIÓN

En el presente capítulo se realiza una propuesta de validación para herramientas de metamodelado partiendo del análisis de las versiones de las herramientas propuestas en la investigación que sirvió de base al presente trabajo, se muestran los principales elementos a tener en cuenta para la instalación de las herramientas.

## 2.1 Análisis de las versiones propuestas

En la investigación referida anteriormente, *"Metamodelado para la construcción de software en entornos libres"* (Isla y Llanes, 2008), se propuso la herramienta AndroMDA versión 3.2, en ese momento era la última actualización del software. Para el objeto del presente trabajo se utilizará la versión 3.3 por ser más actual y presentar mejoras como:

- ✓ La lectura de modelos desde un archivo. Permite leer archivos con extensiones EMX generados por la herramienta Rational Software Architect (RSA) sin necesidad de convertirlos a la extensión UML 2.0.
- ✓ El soporte para las últimas versiones de Maven.

Respecto a AToM3, la versión propuesta fue la 0.2.2, y en la presente investigación se trabajará con la versión 0.3, puesto que esta presenta mejoras con respecto la versión anterior como son:

- ✓ Incluye una mejora en el manejo de excepciones.
- ✓ Adiciona menús contextuales.
- ✓ Permite el manejo de múltiples nodos y relaciones al mismo tiempo.
- ✓ Los nodos y el texto pueden ser escalados.
- ✓ Se pueden cortar, copiar y pegar los nodos, sus relaciones y atributos.
- ✓ Adiciona las opciones deshacer y rehacer.

En la propuesta inicial de validación sólo se tenían en cuenta las dos herramientas mencionadas anteriormente. Sin embargo, en el transcurso de la presente investigación, luego de descubrir su perfecta integración con AndroMDA y su facilidad de uso, se decidió incluir ArgoUML en la propuesta. ArgoUML (ver sección 1.6) posee abundante documentación y es muy fácil de instalar. Es una herramienta de modelado UML que

exporta modelos, los cuales pueden ser interpretados por otras herramientas entre las que se encuentra AndroMDA. Se utilizará la versión 0.26.2 de ArgoUML.

## 2.2 Elementos necesarios para la instalación de las herramientas

A continuación se describen brevemente las herramientas y librerías necesarias para la correcta instalación de las herramientas a validar, así como la localización de estos componentes.

### 2.2.1 AToM3 0.3

Para AToM3 se requiere la instalación de Python 2.3 y Tk/tcl 8.3 o superiores, puede ser instalada sobre Windows, Macintosh o Unix.

| Instalador | Ubicación |
|---|---|
| **Python 2.3 o superior** | http://www.python.org/download/ |
| **Tk/tcl 8.3 o superior** | http://www.tcl.tk/software/tcltk/downloadnow83.tml |
| **AToM3** | http://AToM3.cs.mcgill.ca/people/denis/files/AToM3.zip |

**Tabla 3. Sitios de descarga de las herramientas necesarias para instalar AToM3.**

### 2.2.2 AndroMDA 3.3

Esta herramienta necesita la plataforma Java (Java 2 SDK), versión 1.4 o superior. Para iniciar AndroMDA se recomienda el uso de Maven porque la mayoría de las herramientas AndroMDA vienen con un plugin de Maven.

| Instalador | Ubicación |
|---|---|
| **AndroMDA 3.3** | http://sourceforge.net/project/showfiles.php?group_id=73047&package_id=117392&release_id=593519 |
| **Java 2 SDK** | http://java.sun.com<br>\\10.0.0.22\Software\Development\Java\JVM |
| **Maven** | http://maven.apache.org/ |

**Tabla 4. Sitios de descarga de las herramientas necesarias para instalar AndroMDA.**

### 2.2.3 ArgoUML 0.26.2

Para la instalación de esta herramienta es necesario tener instalada la Máquina Virtual de Java, es decir, el JDK[7] de la plataforma correspondiente, ya sea Windows o Linux.

| Instalador | Ubicación |
|---|---|
| **JDK para Linux** | http://javabasico.osmosislatina.com/java_linux.htm |
| **JDK para Windows** | http://javabasico.osmosislatina.com/java_windows.htm<br>\\10.0.0.22\Software\Development\Java\JVM |
| **ArgoUML para Windows** | http://argouml.tigris.org |

**Tabla 5. Sitios de descarga de las herramientas necesarias para instalar ArgoUML.**

## 2.3 Propuesta de validación

La validación que se efectuará en el presente trabajo se centra en proyectos que actualmente se desarrollan en la Facultad 10. El propósito no es documentar completamente estos proyectos, ni demostrar cuán correcta o no esté elaborada dicha documentación, sino a través de una muestra, en este caso de Teleformación, conformada por ROA, ROXS, Repositorio Semántico y C2Site & C2SCORM, demostrar la utilidad de las herramientas propuestas en proyectos de un dominio común.

A continuación una breve explicación del estado en que se encuentra cada uno de los proyectos y en lo que consisten cada uno de ellos.

**- ROA:** Repositorio de Objetos de Aprendizaje, su función fundamental es almacenar objetos de aprendizaje en forma de paquetes SCORM. Incluye funcionalidades de autoría, que brindan la posibilidad de crear objetos de aprendizaje a partir de recursos ya existentes. Este proyecto se encuentra en la fase de transición.

**- ROXS:** es una herramienta de autor, se centra en diseñar y crear objetos de aprendizaje en forma de paquetes SCORM que posteriormente serán almacenados en el repositorio ROA. Actualmente este software está en proceso de elaboración.

[7] Java Development Kit software que provee herramientas de desarrollo para la creación de programas en Java.

**- Repositorio Semántico:** constituye una extensión de ROA, utilizando la tecnología de la Web Semántica. La Web Semántica pretende resolver limitaciones de la Web actual, mejorando el sistema de búsqueda y recuperación de información en ROA al crecer el número de Objetos de Aprendizaje, este proyecto está en fase de elaboración.

**- C2Site & C2SCORM:** son módulos que se agregan a Moodle 1.8 en forma de bloques. C2Site (Course to Site), tiene la función de tomar un curso del Moodle y exportarlo a un sitio estático. Por su parte C2SCORM (Course to SCORM) toma el curso de la plataforma de aprendizaje y lo exporta en forma de un SCORM para poder ser reutilizado en otros sistemas compatibles con el estándar. Este proyecto se encuentra en la fase de transición.

Teniendo en cuenta que cada una de las herramientas propuestas está dirigida a determinados flujos de trabajo en el proceso de desarrollo de software, son asignadas de manera que les permita a los equipos de proyecto desarrollar artefactos correspondientes a los flujos establecidos.

En la siguiente tabla se muestran cuáles son las herramientas que se utilizarán por cada uno de los proyectos en los distintos flujos de trabajo.

| **Proyectos**<br>**Flujos** | **ROA** | **ROXS** | **Repositorio Semántico** | **C2Site & C2SCORM** |
|---|---|---|---|---|
| **Modelado del negocio** | ArgoUML | | | |
| **Requerimientos** | | | | ArgoUML |
| **Análisis y Diseño** | AndroMDA | ArgoUML | AToM3 | |

**Tabla 6. Asignación de las herramientas por proyectos y flujos de trabajo.**

Para el proceso de validación se deben tener en cuenta los siguientes aspectos:

- ✓ La velocidad y facilidad del proceso de instalación para el usuario.
- ✓ Comportamiento de la herramienta en cuanto a la generación de los modelos deseados.
- ✓ El funcionamiento de las herramientas en la generación de código.

- ✓ La interoperabilidad de las herramientas.
- ✓ Otros aspectos como la Interfaz de usuario y la usabilidad.

En este capítulo se han expuesto los elementos principales para realizar la instalación de las herramientas de metamodelado presentadas en la investigación. También se analizaron las versiones a validar de cada una de ellas presentando sus mejoras y se realizó la propuesta para la validación de las herramientas en diversos tipos de proyectos de la Facultad.

# CAPÍTULO 3: ANÁLISIS DE LOS RESULTADOS

En el presente capítulo se exponen los resultados obtenidos en cada uno de los proyectos en los cuales se llevó a cabo la propuesta de validación presentada en el capítulo anterior.

## 3.1 Resultados obtenidos en los proyectos seleccionados

A continuación se exponen (en letra *cursiva*) las consideraciones emitidas por los respectivos equipos de desarrollo de cada uno de los proyectos designados en los cuales fueron utilizadas las herramientas objeto de validación.

Luego del periodo de instalación y uso de las herramientas, a los equipos de desarrollo se les entregó un cuestionario *(*Anexo 3) a partir del cual se recogerían sus criterios acerca de la(s) herramientas utilizadas. Los aspectos a considerar son los definidos en la sección 2.4 del capítulo anterior.

Los desarrolladores designados aparecen referenciados en el (Anexo 4).

### 3.1.1 ROA

A este proyecto se le asignaron las herramientas ArgoUML y AndroMDA para los flujos de trabajo modelado del negocio, análisis y diseño respectivamente. A continuación se resumen los criterios del equipo de desarrollo acerca de estas herramientas:

- ✓ **ArgoUML**

*El proceso de instalación de la herramienta es bastante simple, y cómodo, con la peculiaridad de que debe tenerse instalado en su computador alguna versión de Java Runtime Enviroment, que permite hacer una selección de algunas herramientas que usa para la generación de código en distintos lenguajes como PHP 4, PHP5, C++, CSharp y Java.*

***La herramienta resulta bastante rápida al importar modelos y diagramas de casos de uso como los que se muestran en (***

Anexo 5 *y* Anexo 6*). En ocasiones, al cargar algunos modelos se tiende a perder la organización de los mismos. Por lo demás, la herramienta trabaja bien, aunque no posee la organización en cuanto a la estructura de un expediente de proyecto, ésta no sugiere organización como lo hace la herramienta de modelado Rational Rose Enterprise Edition.*

*La aplicación en la parte referida al modelado del negocio no prevé que haya generación del código aunque cuando se selecciona algún estereotipo y se busca la pestaña código*

*fuente, este muestra el estilo de una definición en código C++ como se muestra en el* Anexo 7*.*

*En el modelado del negocio, no se considera muy útil, incluyendo además que no se usan los estereotipos estándares como otras herramientas. ArgoUML sería más útil en el final de la fase de inicio y comienzo de la fase de elaboración, en los flujos de trabajo de análisis y diseño e implementación.*

*La herramienta es interesante, puede ser de mucha ayuda sobre todo en los flujos de trabajos mencionados anteriormente, puede ser igual de útil cuando el software se desarrolla usando metodologías ligeras como la XP, pero presenta algunos detalles como son: los estereotipos, hay que tener un buen dominio de la herramienta, pues no sugiere las actividades a realizar dentro de ella, ni contribuye a un trabajo organizado. La herramienta posee una interfaz gráfica que no es la mejor pero se puede trabajar con ella.*

- ✓ **AndroMDA**

*La herramienta AndroMDA necesita para su uso la instalación de la herramienta Maven, la misma se encarga de construir y gestionar los proyectos de AndroMDA. La instalación de Maven es sencilla, es solamente bajar la aplicación compactada y descomprimirla en algún lugar de la computadora. Maven también necesita que se tenga instalado con anterioridad el JDK de Java. Para su correcto funcionamiento, Maven necesita de un repositorio de plugins, este repositorio se encuentra disponible en Internet, en repositorios en línea Cuando la aplicación es iniciada construye un repositorio local, bajando de los repositorios en línea todas las dependencias que necesite. Este proceso es un poco engorroso, debido a que se pueden tener problemas de conectividad y puede que las dependencias no se bajen correctamente, hasta que todas ellas no sean descargadas Maven no funcionará de manera correcta. Una vez que Maven construya su repositorio local, entonces será capaz de construir los proyectos para AndroMDA (Ver* Anexo 8 *y ).*

*Es bueno destacar que esta herramienta no genera modelos. Ella se encarga de interpretar modelos construidos con alguna herramienta de modelado UML y transformarlos a código.*

*En el proceso de generación de código de AndroMDA, Maven sigue siendo protagonista, ya que todo este proceso es guiado por esta herramienta. Una vez que sea generado un proyecto AndroMDA con Maven, entonces se puede comenzar con la generación de código. Una de las herramientas recomendadas para modelar los diagramas UML es*

*ArgoUML. La misma genera los diagramas en formato XMI, siendo este formato el que es interpretado por AndroMDA.*

*Para traducción de estos modelos a código, AndroMDA utiliza unos plugins llamados cartridges o "cartuchos" que son los encargados de generar la estructura y el código de la aplicación mediante reglas definidas en los mismos. Esta forma de crear el código es ventajosa debido a que los desarrolladores pueden crear sus propios cartuchos, o pueden personalizar los ya existentes.*

*La herramienta es capaz de interpretar modelos en formato XMI de otras aplicaciones. Además, los proyectos de AndroMDA pueden ser abiertos en el Eclipse (Ver ).*

### 3.1.2 ROXS

Para la validación en este proyecto se asignó la herramienta ArgoUML para análisis y diseño. A continuación se resumen los criterios del equipo de desarrollo acerca de la herramienta:

- ✓ **ArgoUML**

*ArgoUML puede ser instalada tanto el Linux como en Windows, solo que es necesario tener instalada la maquina virtual de Java, por lo que se hace un poco más compleja la instalación, pero no obstante a eso es un proceso bastante rápido, sin mucha complejidad. Decir además que ya instalada la aplicación se torna un poco complicado el trabajo con la misma puesto que no brinda mucha información gráfica por lo que se debe tener una noción de la misma para comenzar a desarrollar cualquier trabajo.*

*La herramienta genera los diagramas correctamente, aunque puede llegar a ser un trabajo un poco complicado puesto que se deben conocer los estereotipos de las clases del análisis y diseño, ya que estos no aparecen identificados claramente en la herramienta, lo que obliga al usuario a tener un completo dominio de dichos diagramas para crear todos los estereotipos deseados. El trabajo con atributos y operaciones resulta fácil, debido a que hay varias maneras de definir dichos elementos y todas sencillas, ver* Anexo 11, Anexo 12 *y* **Anexo 13**.

*La aplicación genera código en varios lenguajes (Java, C++, PHP, entre otros), este aspecto es muy importante ya que sólo con solicitar que genere las clases en el lenguaje deseado, la aplicación genera el código referente a toda la declaración de los atributos, las operaciones y las relaciones entre clases. Lo negativo de este proceso es que crea*

*muchos comentarios que pueden llegar a confundir al desarrollador, ya que hay mucha información que no es importante pero sí extensa. El código útil no es mucho pero sirve como base al desarrollador para realizar su labor.*

*Un aspecto positivo es que en la herramienta se realizan diagramas poco complejos de manera sencilla, resulta fácil para generar código, es una herramienta que todavía está en desarrollo porque existen varios proyectos para mejorarla en cuanto a comodidad y otros aspectos importantes. Resulta un poco rústica en cuanto a la interfaz gráfica, tiene muchas opciones que resultan un poco complejas, por lo que al realizar proyectos de gran magnitud los usuarios deben estar bien capacitados y documentados en cuanto a la herramienta.*

### 3.1.3…Repositorio Semántico

A este proyecto se le asignó la herramienta AToM3 la cual se centra en el flujo de trabajo análisis y diseño. En el Anexo 14: Diagrama entidad relación realizado con AToM3 por el proyecto Repositorio Semánt, Anexo 16 y Anexo 18 se muestran los diagramas de entidad relación, clases, y secuencia respectivamente, todos ellos realizados con esta herramienta, además de ejemplos de código generado a partir de estos diagramas (Anexo 15 y **Anexo 17**). A continuación se resumen los criterios del equipo de desarrollo acerca de sus experiencias con esta herramienta:

- ✓ **AToM3**

*Para instalar la herramienta AToM3 es necesario instalar previamente Python 2.3.x o una versión superior así como el módulo Tcl/Tk en su versión 8.1 o superior. Una vez que se tengan estas herramientas instaladas el proceso de instalación será sencillo y rápido. Se ejecuta el fichero AToM3.py[8], este pedirá información acerca del directorio donde se guardarán los ficheros creados por el usuario y creará una serie de directorios propios de la instalación.*

*Con la herramienta AToM3 se pueden definir los metamodelos de los modelos que se desean representar. Por defecto esta trae con su instalación la definición de varios modelos como son: Diagramas de Clases, Diagramas Entidad Relación, Diagramas de Secuencia y Diagramas de Colaboración que a modo general cubren el flujo de Análisis y Diseño. Para poder generar algún otro tipo de diagrama, es necesario definir su*

[8] Fichero en código Python, el cual se utiliza para la instalación de la herramienta AToM3.

*metamodelo, los cuales se crean a partir de un diagrama entidad-relación y sobre este se definen un conjunto de reglas que son las que determinan las relaciones en el diagrama. Para un correcto uso de esta herramienta se recomienda el conocimiento sobre el lenguaje de programación Python, ya que resulta importante para la definición de las reglas de transformación.*

*La aplicación genera el código correspondiente a cada modelo creado en el lenguaje Python. Cada vez que el desarrollador genera un nuevo modelo o un metamodelo, AToM3 genera, para cada uno de estos, una serie de ficheros propios del lenguaje Python donde se guarda toda la información de estos.*

*La herramienta es capaz de intercambiar modelos que se encuentren en el formato GML (Geography MarckUp Language ó Lenguaje de Marcado Geográfico) e integrarse con otras que soporten este formato.*

*La herramienta es compleja un cuanto a su utilización, su interfaz es poco amigable, debido a que para crear un modelo es necesario que el metamodelo haya sido construido con anterioridad y no hay un vínculo directo donde se pueda crear directamente el diagrama o el modelo. Tiene como un requerimiento casi indispensable el conocimiento del lenguaje Python para su utilización. Además la aplicación está actualmente en fase de desarrollo. No obstante, esta herramienta es potente en lo que respecta a la creación de modelos, por las características que presenta.*

### 3.1.4 C2Site & C2SCORM

A este proyecto se le asignó la herramienta ArgoUML en el flujo de trabajo de requerimiento. En el Anexo 19, Anexo 20 y Anexo 21 se muestran los diagramas de caso de uso no.1 y 2, y un diagrama de clases respectivamente, todos ellos realizados con esta herramienta, además de ejemplos de código que se genera a partir del diagrama de clases, el cual se muestra en el Anexo 22. A continuación se resumen los criterios del equipo de desarrollo acerca de la herramienta utilizada:

- ✓ **ArgoUML**

*El proceso de instalación de la herramienta es rápido y no consume muchos recursos del sistema, la misma es fácil de instalar tanto en Linux como en Windows y no requiere ningún conocimiento adicional.*

*La herramienta realiza correctamente los diagramas y pueden ser generados en distintas extensiones y con buena calidad visual.*

*ArgoUML es bastante completa en el sentido de que permite la generación de código en diferentes lenguajes partiendo solamente de las clases del proyecto, ahorra gran cantidad de tiempo a los programadores y organiza mejor el código identado correctamente para una mejor visualización y entendimiento del mismo.*

*La aplicación puede importar diagramas en XMI y XML, es capaz tanto de generar como de importar estos diagramas y también exporta a UML.*

*La herramienta es de gran ayuda para el desarrollo de cualquier proyecto, es fácil de usar y todas las opciones tienen una breve explicación de su función que da una mejor idea al usuario de cuando utilizar cada una. Puede exportar los diagramas en diferentes extensiones. Este software es libre y se considera que la Universidad debería analizar la posibilidad de sustituir las herramientas de modelado que hoy se utilizan como Visual Paradigm y Rational Rose Enterprise Edition por esta que se adapta mejor a las necesidades de la Universidad permitiendo además la generación de código en diferentes lenguajes a partir del diseño de clases del proyecto ganando en tiempo y organización a la hora de desarrollar.*

## 3.2 Análisis valorativo de los resultados obtenidos

En las secciones que siguen se analizan los criterios emitidos por cada uno de los equipos de proyecto a partir de los resultados de el cuestionario (ver Anexo 3) aplicado a los desarrolladores y las valoraciones de las autoras de la presente investigación. Se tienen en cuenta los aspectos definidos en la sección 2.4. Se muestran las principales valoraciones para cada herramienta por separado. En cada caso, se incluye una figura donde se muestra la evaluación integrada de cada aspecto dada por los desarrolladores y las autoras:

- ✓ **AToM3**

Esta herramienta en general presenta una interfaz de usuario poco amigable. Para la realización de un modelo debe estar definido su metamodelo con anterioridad, de lo contrario se debe crear uno nuevo a partir del modelo entidad relación. Además se debe tener un conocimiento mínimo de Python lo que la hace compleja para un usuario común. Al realizar un modelo o metamodelo se crean ficheros de código Python con la información de cada uno de ellos, lo que permite definir nuevos metamodelos y ahorrar

tiempo a los desarrolladores. En cuanto a la generación de modelos, esta herramienta trae algunos metamodelos definidos que se ajustan al flujo de trabajo análisis y diseño. AToM3 se puede integrar a otras herramientas que soporten el formato GML, en el cual ella exporta modelos. Aún está en fase de desarrollo por lo que futuras versiones podrían incluir más funcionalidades.

AToM3, pudiera ser de gran utilidad para proyectos que trabajen en Python puesto que facilitaría la generación de código en ese lenguaje. Resalta como aspecto negativo que sólo se integra al análisis y diseño. También necesita de usuarios capacitados en el área del metamodelado, pues no existe mucha experiencia en este campo y, por otro lado, debe dominarse el formalismo utilizado.

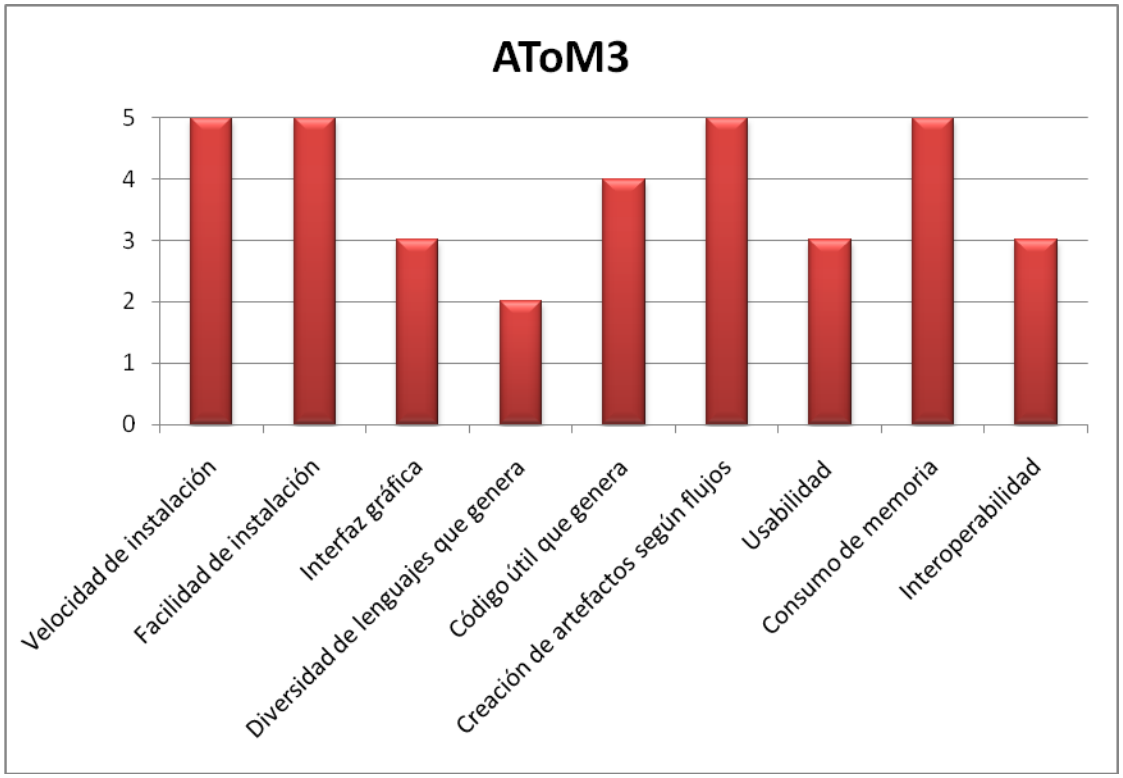



**Figura 6: Evaluación de AToM3.**

- ✓ **AndroMDA**

Teniendo en cuenta los aspectos analizados para la validación de la herramienta se concluye que el proceso de instalación resulta complicado, ya que es necesario tener conocimientos del trabajo con Maven y del lenguaje de programación Java. La herramienta no permite la creación de modelos, necesita obligatoriamente de otra herramienta que le proporcione los mismos, lo que la hace dependiente. No presenta una

interfaz gráfica, se accede a ella a través del Maven lo que la hace menos amigable al usuario. Se considera que la herramienta es bastante potente en el proceso de generación de código, creando para ello una estructura de proyecto proporcionando una mayor organización. Posee una alta interoperabilidad ya que puede importar proyectos de varias herramientas de modelado tales como: ArgoUML, MagicDraw 9.x, MagicDraw 15.5 y RSM 6. El proyecto generado con AndroMDA puede ser abierto con herramientas como el Eclipse. AndroMDA se enfoca al apoyo de todo el proceso de desarrollo de software.

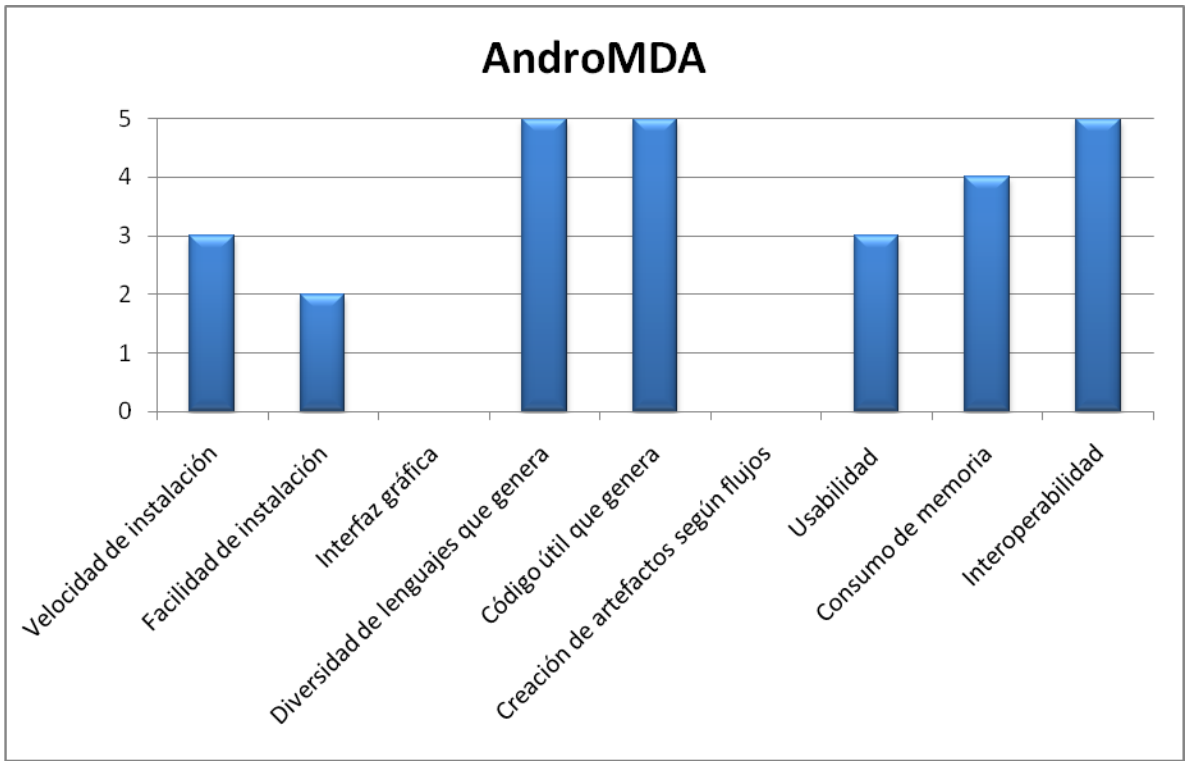



**Figura 7: Evaluación de AndroMDA.**

- ✓ **ArgoUML**

Se puede destacar positivamente su instalación ya que es rápida y sencilla, pudiéndose realizar en Windows y Linux. Uno de los aspectos negativos de la herramienta, señalado según el proyecto ROA, es que en el modelado del negocio, la herramienta no genera código y no usa los estereotipos estándares. En cuanto al primer señalamiento, se debe aclarar que no es necesaria la generación de código en este flujo, ya que el código que se pueda generar a partir de los diagramas de casos de uso no es útil. ArgoUML genera código a partir de diagramas de clases en varios lenguajes de programación pudiendo importar y exportar diagramas en distintas extensiones, mostrando alto grado de interoperabilidad. Es muy útil para el análisis, diseño e implementación de un proyecto

debido a su capacidad de generación de código a partir de los diagramas. Según un señalamiento del proyecto ROXS, la herramienta necesita completo dominio de los diagramas para la generación de modelos. Sin embargo, es fácil de aprender a usar si se tiene experiencia con cualquier otra herramienta de modelado. Está bien documentado. Cuando se desea realizar un diagrama se selecciona el correspondiente y se muestran los artefactos necesarios para su realización. El proyecto ROA señala como aspecto negativo que ArgoUML no sugiere una organización del trabajo, lo cual fue señalado por la costumbre en el uso de la metodología RUP, la cual posee una estructura de organización detallada. ArgoUML, sin embargo, se basa en metodologías agiles, y por ello posee una estructura organizativa más sencilla.

El uso de esta herramienta podría ser de gran utilidad para el desarrollo de proyectos informáticos, aunque se sugiere que dichos proyectos sean pequeños y que empleen metodologías agiles. Ahorra tiempo a los desarrolladores con el código que genera a partir de los diagramas definidos que aunque no es abundante representa el código base para que un programador se centre en las partes complejas que requiera desarrollar del software. Debido a que es software libre, se adapta mejor a las necesidades del país basándose en los principios de independencia tecnológica.

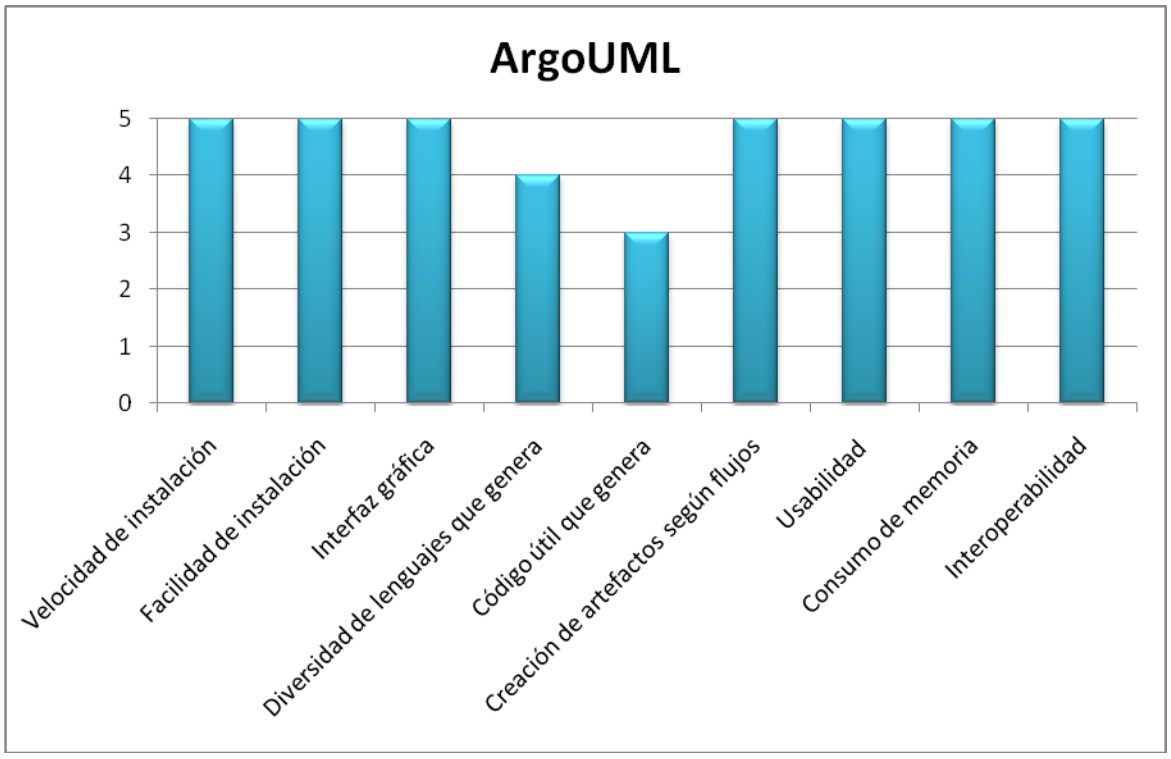



**Figura 8: Evaluación de ArgoUML.**

En la Figura 9 se muestra el comportamiento de AndroMDA, AToM3 y ArgoUML en cuanto a los aspectos mostrados en la propuesta de validación.

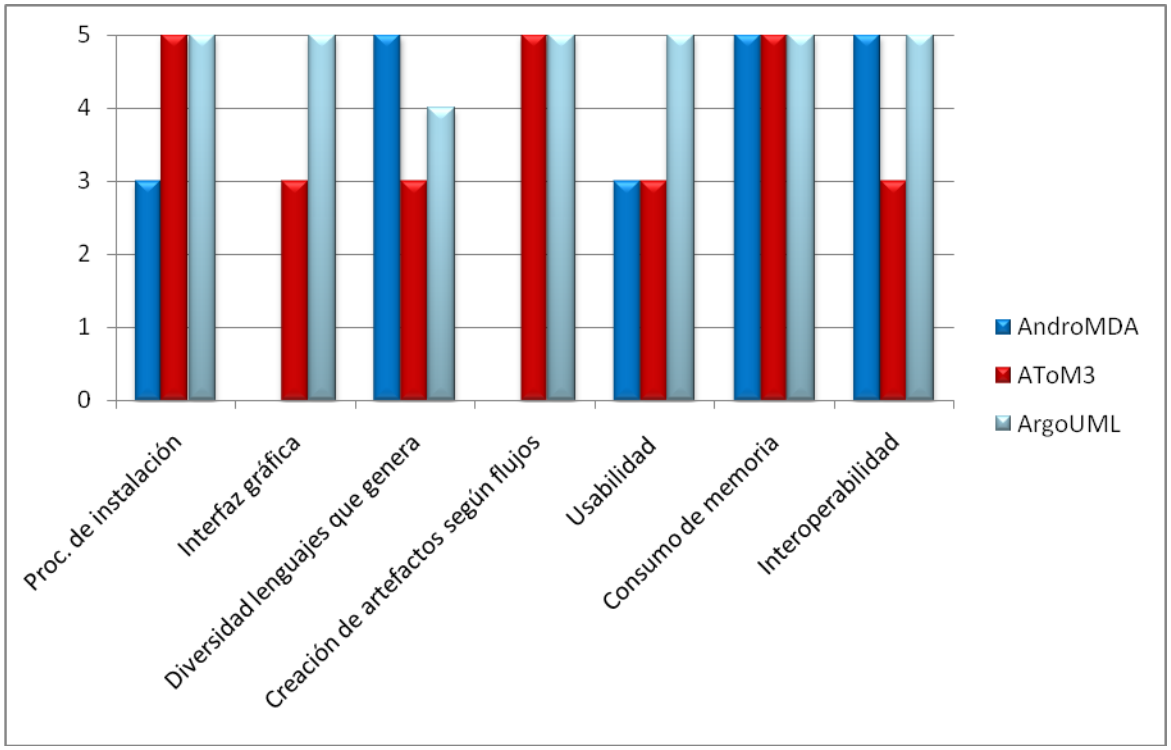



**Figura 9: Resultados de la validación.**

En este capítulo se analizaron los resultados obtenidos a partir de la puesta en práctica de la propuesta de validación; resaltando como aspectos más significativos que AToM3 permite la creación de cualquier modelo a partir de la definición de su metamodelo correspondiente, AndroMDA resulta de gran potencia en cuanto a la generación de código en varios lenguajes de programación y ArgoUML posibilita de manera sencilla la creación de modelos que luego serán empleados por AndroMDA. A partir de ello se puede arribar a la conclusión de que el uso de estas herramientas posibilitaría a los desarrolladores enfocarse más hacia el “qué” y menos en el “cómo” se desarrollan las aplicaciones.

# CONCLUSIONES

Teniendo en cuenta el estudio y las actividades llevadas a cabo para realizar el proceso de validación de herramientas de metamodelado se puede llegar a las siguientes conclusiones:

- Las herramientas de metamodelado pueden ser utilizadas para la creación de metamodelos para un dominio común y por tanto, también lenguajes y artefactos específicos para ese dominio.
- AToM3 trae consigo un conjunto de formalismos definidos que se ajustan al flujo de trabajo de análisis y diseño, éstos permiten la creación de modelos de los cuales se obtiene código Python, en caso de que se desee construir otro tipo de diagrama se puede definir un nuevo metamodelo.
- AndroMDA proporciona una estructura de proyecto y brinda facilidades para la generación de código para una amplia gama de lenguajes a partir de modelos (por ejemplo ArgoUML). Es dependiente de una herramienta de modelado que le proporciona los modelos, y de una aplicación que la soporte ya que no cuenta con una interfaz de usuario. Puede ser extendida con cartuchos específicos para otros lenguajes.
- ArgoUML es una herramienta de modelado fácil de usar. Puede ser utilizada de modo independiente o vinculada a AndroMDA, a la cual le proporciona los modelos para el proceso de generación de código. Resulta de gran compatibilidad con esta herramienta por los estereotipos con los que cuenta para desarrollar sus proyectos y las extensiones en las cuales son exportados.

# RECOMENDACIONES

Para dar continuidad al presente trabajo y teniendo en cuenta las experiencias adquiridas durante el desarrollo del mismo, se recomienda:

- Profundizar en el estudio del metamodelado y de las herramientas propuestas, así como, brindar capacitación a especialistas e ingenieros de software.
- Aplicar las herramientas de metamodelado AToM3 y AndroMDA a proyectos teniendo en cuenta que pueden ser usadas por un grupo de proyectos de un mismo dominio.
- Utilizar los conceptos asociados al metamodelado para el estudio de Dominios Específicos, para los cuales se pudieran crear metamodelos, y herramientas basadas en éstos, que faciliten el desarrollo de software específico para dichos dominios. Esto se pudiera hacer a nivel de Líneas de Investigación, Polos Productivos o Líneas Temáticas.

# REFERENCIAS BIBLIOGRÁFICAS

**Anaya de Páez, Raquel. Arango, Fernando. Zapata, Carlos M. 2007.** *Estudio comparativo de las herramientas MetaCASE bajo consistencia y refinamiento.* [pdf] 2007.

**AndroMDA. 2006.** AndroMDA_ES. [En línea] 2006. [Citado el: 12 de febrero de 2009.] http://svn.assembla.com/svn/bJ15ViGayr3yf1ab7jnrAJ/OpenMdDX%20vs%20AndroMDA/AndroMDA_ES.doc.

**ArgoUML. 2008.** Open Source Software Engineering Tools. [En línea] 2008. [Citado el: 16 de marzo de 2009.] http://argouml.tigris.org/features.html.

**Bombadil. 2009.** BosqueViejo. [En línea] 27 de marzo de 2009. [Citado el: 3 de marzo de 2009.] http://bosqueviejo.net/wordpress/category/noticias/.

**De Lara Jaramillo, Juan. 2005.** *Simulación Educativa Mediante Meta-Modelado y Gramáticas de Grafos.* [pdf] Madrid, España : s.n., 2005.

**De Lara Jaramillo, Juan y Perez, Francisco. 2006.** *Hacia la definición de lenguajes específicos de dominio con sintaxis gráfica y textual.* s.l., España : Departamento de Ingeniería Informática. Universidad Autónoma de Madrid, 2006.

**De Lara Jaramillo, Juan. Vangheluwe, H. Alfonseca, M. 2003.** *Using Meta-Modelling and Graph- Grammars to Create Modelling Environments. Electronic Notes in Theoretical Computer.* 2003. Vol. 72. 3.

**DSDM. 2008.** DSDM: Desarrollo de Software Dirigido por Modelos. MDA y Aplicaciones. [En línea] 18 de abril de 2008. http://www.lcc.uma.es/~av/MDD-MDA/.

**Estévez, Antonio, Pelechano, Vicente y Vallecillo, Antonio. 2007.** IV Taller sobre Desarrollo de Software Dirigido por Modelos, MDA y Aplicaciones (DSDM'07). [En línea] 11 de septiembre de 2007. http://www.taro.ull.es/dsdm07/.

**García, Jesús. 2007.** *Tema 2. Fundamentos del DSDM. Metamodelado.* s.l., España : Universidad de Murcia. Departamento de Informática y Sistemas, 13 de junio de 2007.

**Gómez Palarea, Pablo. Sánchez Ramón, Óscar. 2006.** *Herramientas de Metamodelado Microsoft DSL Tools vs MetaEdit+.* Murcia, España : s.n., 2006.

**González, Jaime. 1998.** El discreto encanto del metamodelo de UML. *zeusconsult.com.mx.* [En línea] agosto de 1998. [Citado el: 26 de febrero de 2009.] http://www.zeusconsult.com.mx/DISCRET.pdf.

**Lee, Insup, Leung, Joseph y Son, Sang. 2007.** *Handbook of Real-Time and Embedded Systems.* [ed.] Chapman y Hall. s.l. : CRC Press, 2007.

**Isla Rodríguez, Yaimet y Llanes Martell, Lázara. 2008.** *Metamodelado para la construcción de software en entornos libres.* Universidad de las Ciencias Informaticas. Ciudad de la Habana : s.n., 2008.

**Lopez L, Edna. Gonzales G, Moisés. Lopez S, Máximo. Iduñate R, Erick. 2007.** *Proceso de Desarrollo de Software Mediante Herramientas MDA.* [pdf] 2007.

**Navia Cárdenas, Andrés. 2008.** *Sistema de Costos Basado en Actividades.* [pdf] agosto de 2008.

**Nazareno, Gonzalo. 1999.** *Herramientas Case.* [pdf] Domicilio, Redacción y Talleres, Jesus Maria : s.n., noviembre de 1999.

**Perdiguero Castellanos, Gema. 2009.** Tutorial: Domain-Specific Languages (DSLs) en Apache Camel. [En línea] 6 de marzo de 2009. http://www.tecsisa.com/index.igw?item=1628.

**Quasar Tecnología. 2007.** Metamodelado. El futuro en el desarrollo de aplicaciones. [En línea] 2007. [Citado el: 15 de enero de 2009.] http://www.quasartecnologia.com/servicios/link/attachments/METAMODELADO64727421.pdf.

**Reina, A. Torres, J. Toro, M. 2006.** *Hacia lenguajes de metamodelado orientados a aspectos.* [pdf] Barcelona, España : s.n., Octubre de 2006.

**Sánchez, Omar. 2008.** Modelos, Control y Sistema de Visión. [En línea] 2008. [Citado el: 20 de febrero de 2009.] http://omarsanchez.net/conceptomod.aspx.

**SAPO-XP. 2008.** *Estimaciones y Mediciones.* [doc] 2008.

**Soler, Emilio. Trujillo, Juan. Fernández-Medina, Eduardo. Piattini, Mario. 2007.** *Una extensión del metamodelo relacional de CWM para representar Almacenes de Datos Seguros a nivel lógico.* [pdf] España-Cuba : s.n., 2007.

**Tasof. 2005.** De Modelos, Metamodelos y Metametamodelos. [En línea] 13 de octubre de 2005. http://tasof-ucn.blogspot.com/2005/10/de-modelos-metamodelos-y.html.

**Tuesta Pereyra, Martin. 2005.** El Prisma.Ingenieria de Sistemas e Informatica. [En línea] 2005. [Citado el: 2 de abril de 2009.] http://www.elprisma.com/apuntes/curso.asp?id=13324.

**Ubuntu. 2007.** Guia-ubuntu. [En línea] 3 de octubre de 2007. http://www.guia-ubuntu.org/index.php?title=ArgoUML.

**Zapata J., Carlos M. Tamayo O, Paula. Arango I, Fernando. 2007.** Scielo. [En línea] 8 de agosto de 2007. [Citado el: 2 de abril de 2009.] http://www.scielo.org.co/scielo.php?pid=S0012-73532007000300026&script=sci_arttext.

**Zapata, Carlos . Álvarez, Carlos Alberto. 2005.** *CONVERSION DE DIAGRAMAS DE PROCESOS EN DIAGRAMAS DE CASOS DE USOS USANDO ATOM3.* [pdf] Medellin, Colombia : Universidad NAcional de Colombia, 2005. Vol. 72.

# BIBLIOGRAFÍA

**Anaya de Páez, Raquel. Arango, Fernando. Zapata, Carlos M. 2007.** *Estudio comparativo de las herramientas MetaCASE bajo consistencia y refinamiento.* [pdf] 2007.

**AndroMDA, 2006.** AndroMDA_ES. [En línea] 2006. [Citado el: 12 de febrero de 2009.] http://svn.assembla.com/svn/bJ15ViGayr3yf1ab7jnrAJ/OpenMdDX%20vs%20AndroMDA/AndroMDA_ES.doc.

**ArgoUML. 2008.** Open Source Software Engineering Tools. [En línea] 2008. [Citado el: 16 de marzo de 2009.] http://argouml.tigris.org/features.html.

**Bellinger, Gene. 1997.** *Modeling & Simulation.* s.l. : Outsights Corp, 1997.

**Bollati, Verónica. Vara, Juan. Vela, Belén. Marcos, Esperanza. 2008.** Una revisión de herramientas MDA. [En línea] 2008. [Citado el: 18 de febrero de 2009.] http://www.sistedes.es/sistedes/pdf/2007/dsdm-07-bollati-revisionHerramientas.pdf.

**Bombadil. 2009.** BosqueViejo. [En línea] 27 de marzo de 2009. [Citado el: 3 de marzo de 2009.] http://bosqueviejo.net/wordpress/category/noticias/.

**De Lara Jaramillo, Juan. 2005.** *Simulación Educativa Mediante Meta-Modelado y Gramáticas de Grafos.* [pdf] Madrid, España : s.n., 2005.

**De Lara Jaramillo, Juan y Perez, Francisco. 2006.** *Hacia la definición de lenguajes específicos de dominio con sintaxis gráfica y textual.* s.l., España : Departamento de Ingeniería Informática. Universidad Autónoma de Madrid, 2006.

**De Lara Jaramillo, Juan. Vangheluwe, H. Alfonseca, M. 2003.** *Using Meta-Modelling and Graph- Grammars to Create Modelling Environments. Electronic Notes in Theoretical Computer.* 2003. Vol. 72. 3.

**DSDM. 2008.** DSDM: Desarrollo de Software Dirigido por Modelos. MDA y Aplicaciones. [En línea] 18 de abril de 2008. http://www.lcc.uma.es/~av/MDD-MDA/.

**Estévez, Antonio, Pelechano, Vicente y Vallecillo, Antonio. 2007.** IV Taller sobre Desarrollo de Software Dirigido por Modelos, MDA y Aplicaciones (DSDM'07). [En línea] 11 de septiembre de 2007. http://www.taro.ull.es/dsdm07/.

**García, Jesús. 2007.** *Tema 2. Fundamentos del DSDM. Metamodelado.* s.l., España : Universidad de Murcia. Departamento de Informática y Sistemas, 13 de junio de 2007.

**Gómez Palarea, Pablo. Sánchez Ramón, Óscar. 2006.** *Herramientas de Metamodelado Microsoft DSL Tools vs MetaEdit+.* Murcia, España : s.n., 2006.

**González, Jaime. 1998.** El discreto encanto del metamodelo de UML. *zeusconsult.com.mx.* [En línea] agosto de 1998. [Citado el: 26 de febrero de 2009.] http://www.zeusconsult.com.mx/DISCRET.pdf.

**Lee, Insup, Leung, Joseph y Son, Sang. 2007.** *Handbook of Real-Time and Embedded Systems.* [ed.] Chapman y Hall. s.l. : CRC Press, 2007.

**Llanes Martell, Lázara y Isla Rodríguez, Yaimet. 2008.** *Metamodelado para la construcción de software en entornos libres.* Universidad de las Ciencias Informaticas. Ciudad de la Habana : s.n., 2008.

**Lopez L, Edna. Gonzales G, Moisés. Lopez S, Máximo. Iduñate R, Erick. 2007.** *Proceso de Desarrollo de Software Mediante Herramientas MDA.* [pdf] 2007.

**Martín Martínez, Gema. Pareja León, Ángel. 2006.** *HERRAMIENTAS CASE: ARGO UML.* 2006. Vol. pdf.

**Navia Cárdenas, Andrés. 2008.** *Sistema de Costos Basado en Actividades.* [pdf] agosto de 2008.

**Nazareno, Gonzalo. 1999.** *Herramientas Case.* [pdf] Domicilio, Redacción y Talleres, Jesus Maria : s.n., noviembre de 1999.

**Perdiguero Castellanos, Gema. 2009.** Tutorial: Domain-Specific Languages (DSLs) en Apache Camel. [En línea] 6 de marzo de 2009. http://www.tecsisa.com/index.igw?item=1628.

**Quasar Tecnología. 2007.** Metamodelado. El futuro en el desarrollo de aplicaciones. [En línea] 2007. [Citado el: 15 de enero de 2009.] http://www.quasartecnologia.com/servicios/link/attachments/METAMODELADO64727421.pdf.

**Reina, A. Torres, J. Toro, M. 2006.** *Hacia lenguajes de metamodelado orientados a aspectos.* [pdf] Barcelona, España : s.n., Octubre de 2006.

**Sánchez, Omar. 2008.** Modelos, Control y Sistema de Visión. [En línea] 2008. [Citado el: 20 de febrero de 2009.] http://omarsanchez.net/conceptomod.aspx.

**SAPO-XP. 2008.** *Estimaciones y Mediciones.* [doc] 2008.

**Soler, Emilio. Trujillo, Juan. Fernández-Medina, Eduardo. Piattini, Mario. 2007.** *Una extensión del metamodelo relacional de CWM para representar Almacenes de Datos Seguros a nivel lógico.* [pdf] España-Cuba : s.n., 2007.

**Tasof. 2005.** De Modelos, Metamodelos y Metametamodelos. [En línea] 13 de octubre de 2005. http://tasof-ucn.blogspot.com/2005/10/de-modelos-metamodelos-y.html.

**Tuesta Pereyra, Martin. 2005.** El Prisma.Ingenieria de Sistemas e Informatica. [En línea] 2005. [Citado el: 2 de abril de 2009.] http://www.elprisma.com/apuntes/curso.asp?id=13324.

# Bibliografía

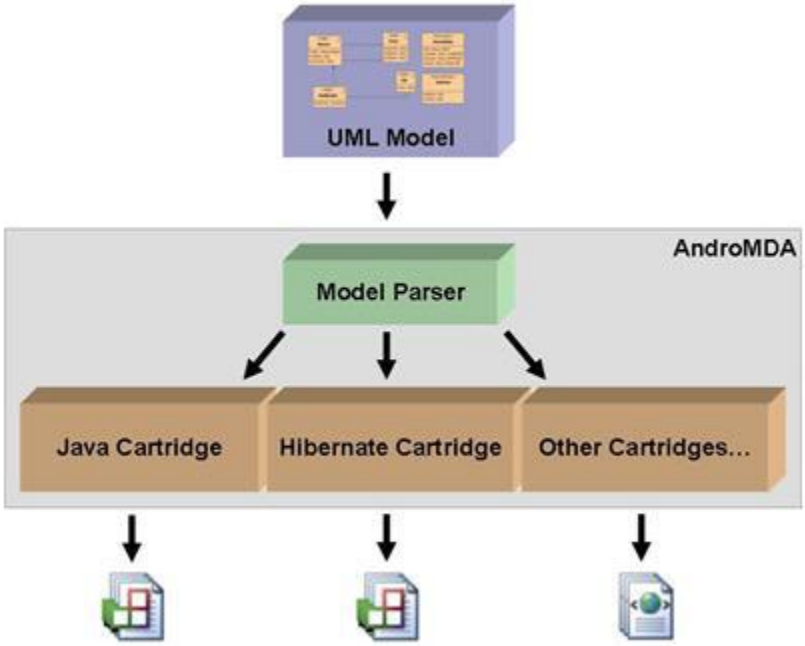
**Ubuntu. 2007.** Guia-ubuntu. [En línea] 3 de octubre de 2007. http://www.guia-ubuntu.org/index.php?title=ArgoUML.

**Zapata J., Carlos M. Tamayo O, Paula. Arango I, Fernando. 2007.** Scielo. [En línea] 8 de agosto de 2007. [Citado el: 2 de abril de 2009.] http://www.scielo.org.co/scielo.php?pid=S0012-73532007000300026&script=sci_arttext.

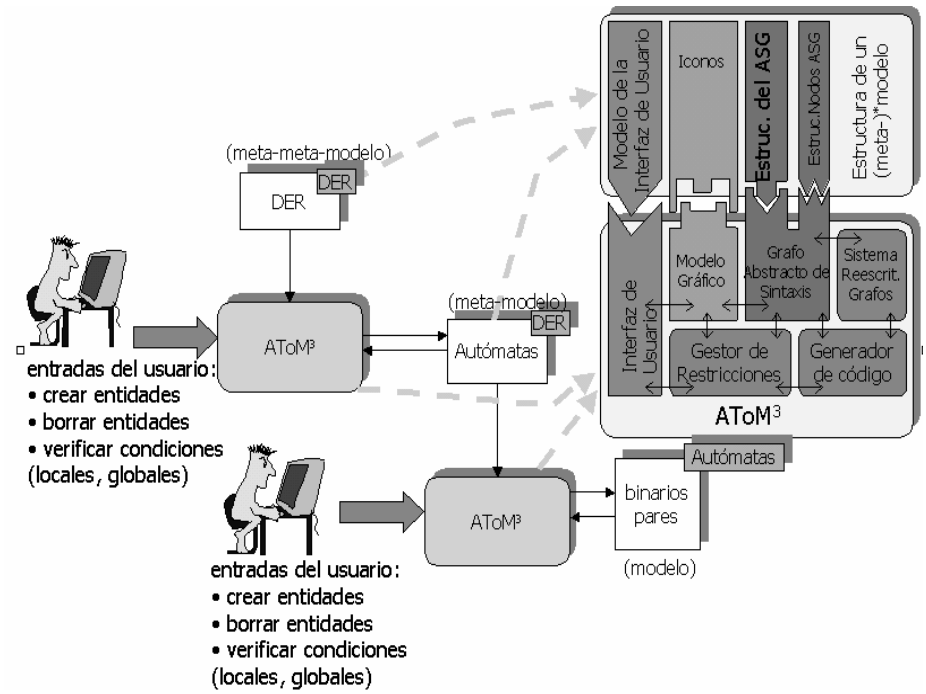
**Zapata, Carlos . Álvarez, Carlos Alberto. 2005.** *CONVERSION DE DIAGRAMAS DE PROCESOS EN DIAGRAMAS DE CASOS DE USOS USANDO ATOM3.* [pdf] Medellin, Colombia : Universidad NAcional de Colombia, 2005. Vol. 72.

# ANEXOS

**Anexo 1: Arquitectura de AndroMDA.**




**Anexo 2: Estructura de AToM3.**



**Anexo 3: Encuesta realizada a los equipos de desarrollo.**

**CUESTIONARIO PARA EL TRABAJO DE DIPLOMA "VALIDACION DE HERRAMIENTAS DE METAMODELADO"**

El presente cuestionario tiene como objetivo capturar los resultados de la utilización de las herramientas AToM3, AndroMDA y ArgoUML por parte de los equipos de desarrollo de cada proyecto. Las dos primeras son herramientas de metamodelado[9], mientras que la última es de modelado.

**Importante:** Responda con la mayor sinceridad todos los aspectos del cuestionario.

**Seleccione según corresponda:**

**Herramientas a validar:**

AToM3____ AndroMDA____ ArgoUML____

**Flujo de trabajo:**

Modelado del negocio___ Requerimientos___ Análisis y Diseño___

**Proyecto:** _____

**Nombre y apellidos:** _____

**Rol que desempeña:** _____

**1. Según su experiencia con la herramienta, evalúe los criterios presentados en la tabla.**

Los criterios estarán comprendidos entre los valores del 1al 5, teniendo en cuenta que 1 representa el nivel bajo, y 5 el nivel más alto.

| No. | Criterios | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 1 | Velocidad de instalación | | | | | |
| 2 | Facilidad de instalación | | | | | |
| 3 | Interfaz gráfica | | | | | |
| 4 | Diversidad de lenguajes que genera | | | | | |
| 5 | Código útil que genera | | | | | |

[9] Metamodelado: es una actividad que produce metamodelos.

| | | | | | | |
|---|---|---|---|---|---|---|
| **6** | Creación artefactos según flujos | | | | | |
| **7** | Usabilidad | | | | | |
| **8** | Consumo de memoria | | | | | |
| **9** | Interoperabilidad | | | | | |

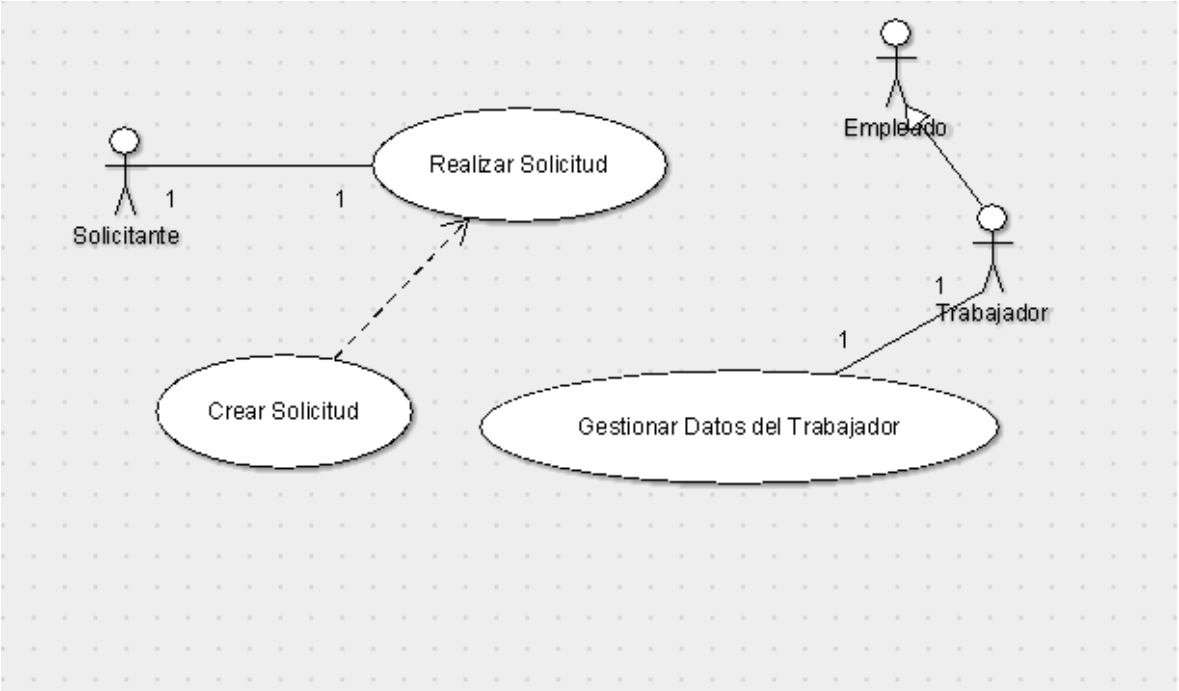
**2. Justifique el valor otorgado a cada criterio mencionado en la tabla, según el trabajo con la herramienta.**

_____
_____
_____
_____
_____

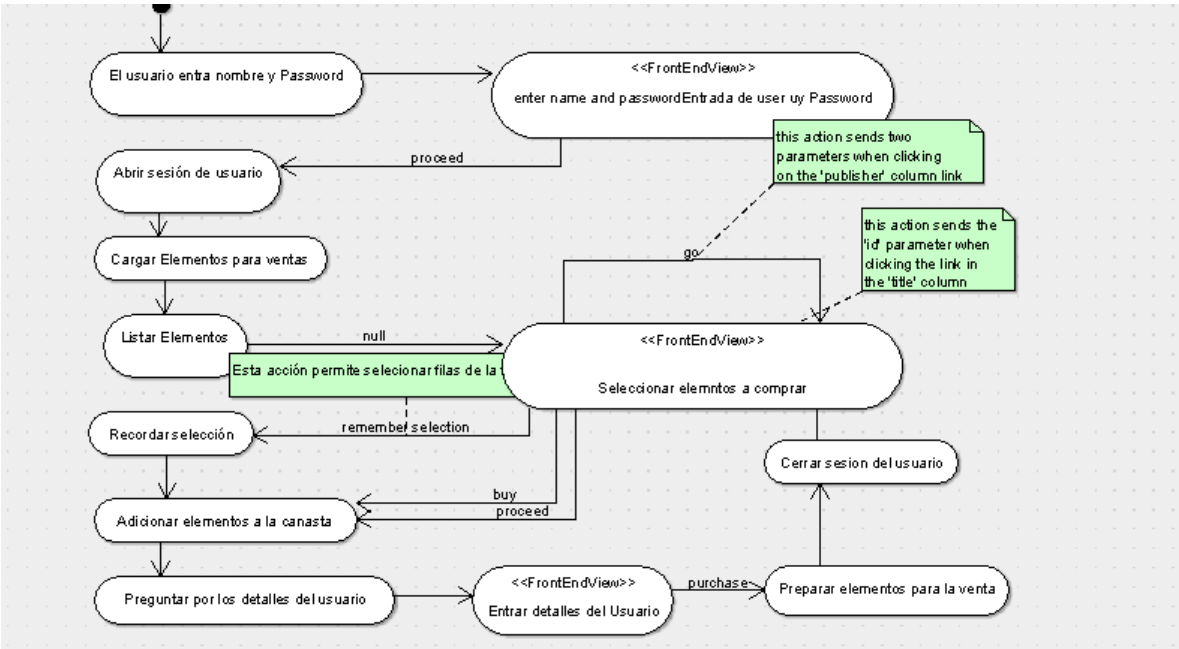
**Anexo 4: Responsables de realizar el proceso de validación en cada uno de los proyectos seleccionados.**

| Proyecto | Nombre y apellidos | Rol |
|---|---|---|
| **ROA** | Leonardo Rodríguez González | Desarrollador |
| **ROXS** | Pedro Gustavo Sáez Vallejo | Desarrollador |
| **Repositorio Semántico** | Maikel Aparicio Reytor | Desarrollador |
| **C2Site & C2Scorm** | Annier Pérez Ricardo | Desarrollador |

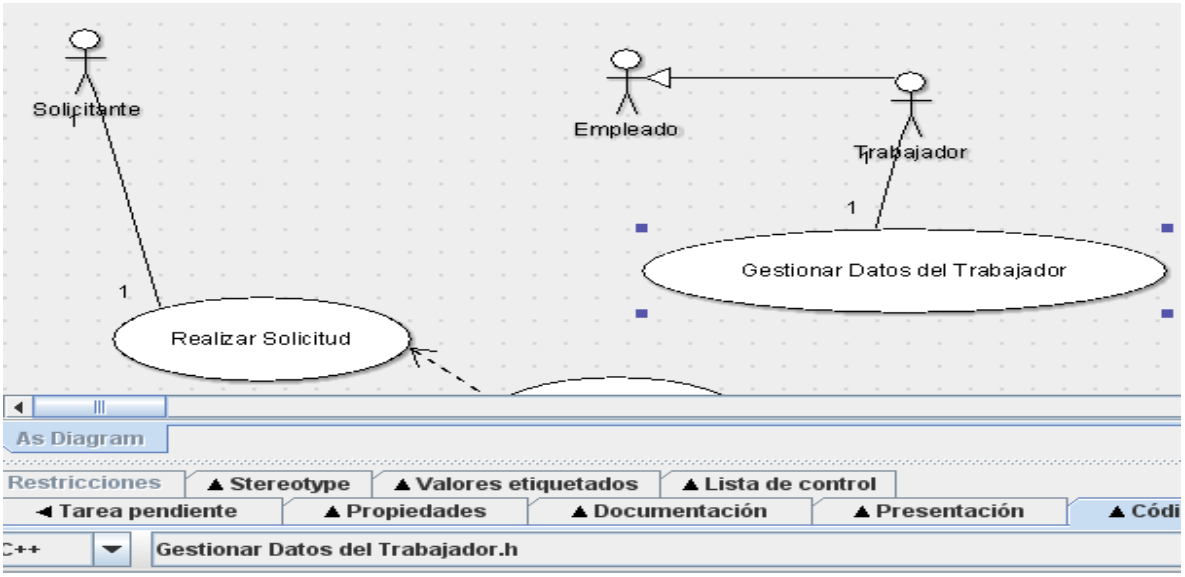
**Anexo 5: Diagrama de casos de uso realizado con la herramienta ArgoUML por el proyecto ROA.**




**Anexo 6: Diagrama de estados realizado con ArgoUML por el proyecto ROA.**

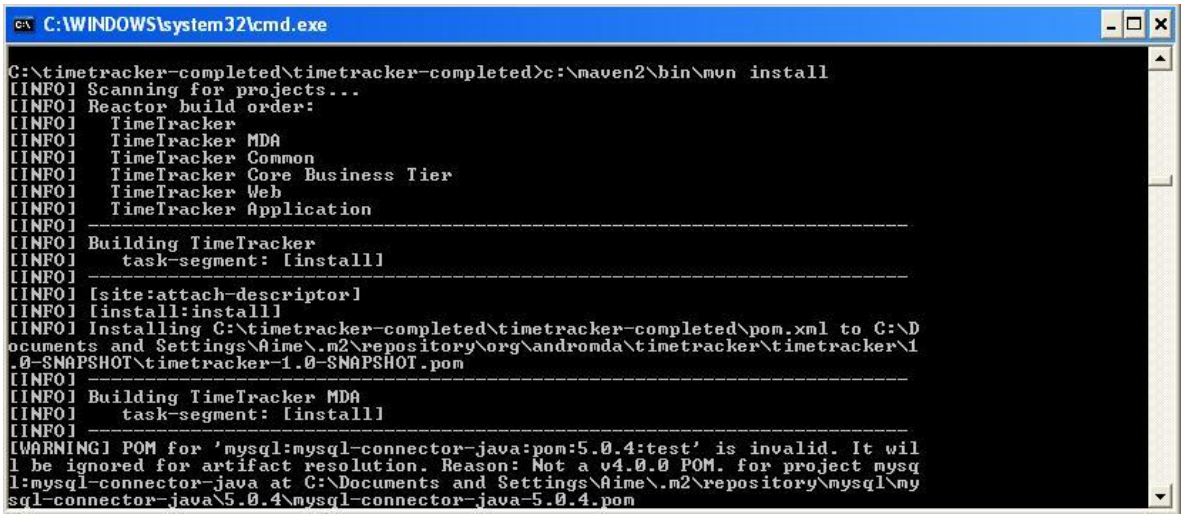


**Anexo 7: Muestra de código al seleccionar un estereotipo con ArgoUML por el proyecto ROA.**




```
ifndef Gestionar Datos del Trabajador_h
define Gestionar Datos del Trabajador_h


endif // Gestionar Datos del Trabajador_h
```

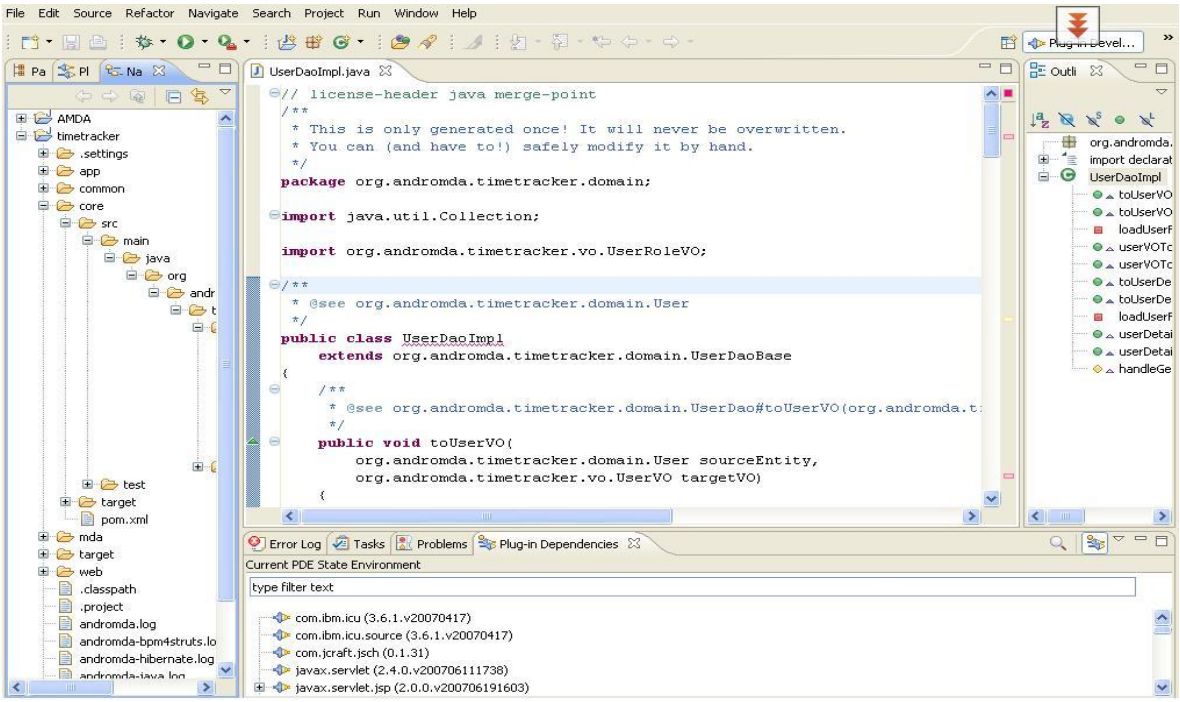
**Anexo 8: Generación del proyecto AndroMDA con Maven 2, vista de consola.**



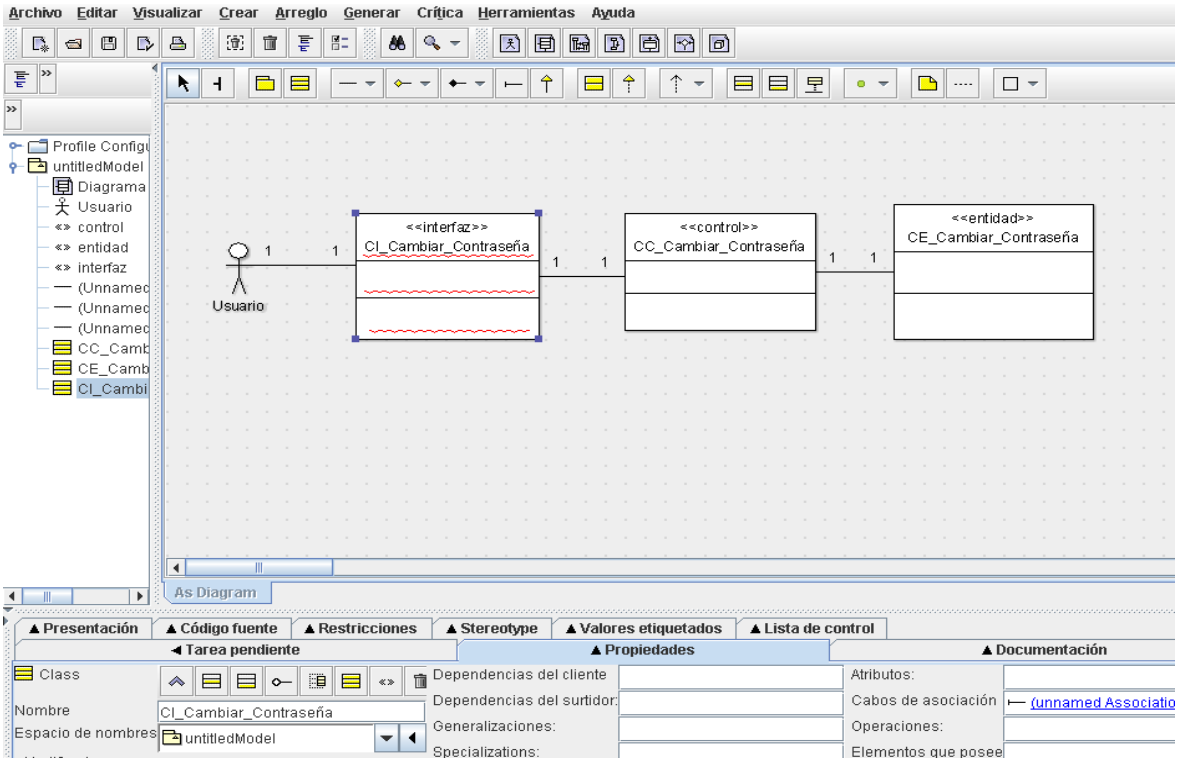
**Anexo 9: Esqueleto de proyecto AndroMDA, visto desde explorador de archivos.**

| | | | |
|---|---|---|---|
| .settings | | Carpeta de archivos | 29/05/2009 04:09 p... |
| app | | Carpeta de archivos | 29/05/2009 04:09 p... |
| common | | Carpeta de archivos | 29/05/2009 04:09 p... |
| core | | Carpeta de archivos | 29/05/2009 04:09 p... |
| mda | | Carpeta de archivos | 29/05/2009 04:09 p... |
| web | | Carpeta de archivos | 29/05/2009 04:09 p... |
| .classpath | 4 KB | Archivo CLASSPATH | 10/02/2007 01:01 p... |
| .project | 1 KB | Archivo PROJECT | 16/08/2006 04:31 p... |
| pom.xml | 29 KB | Documento XML | 22/10/2008 03:45 p... |
| readme.txt | 11 KB | Documento de texto | 10/02/2007 01:01 p... |

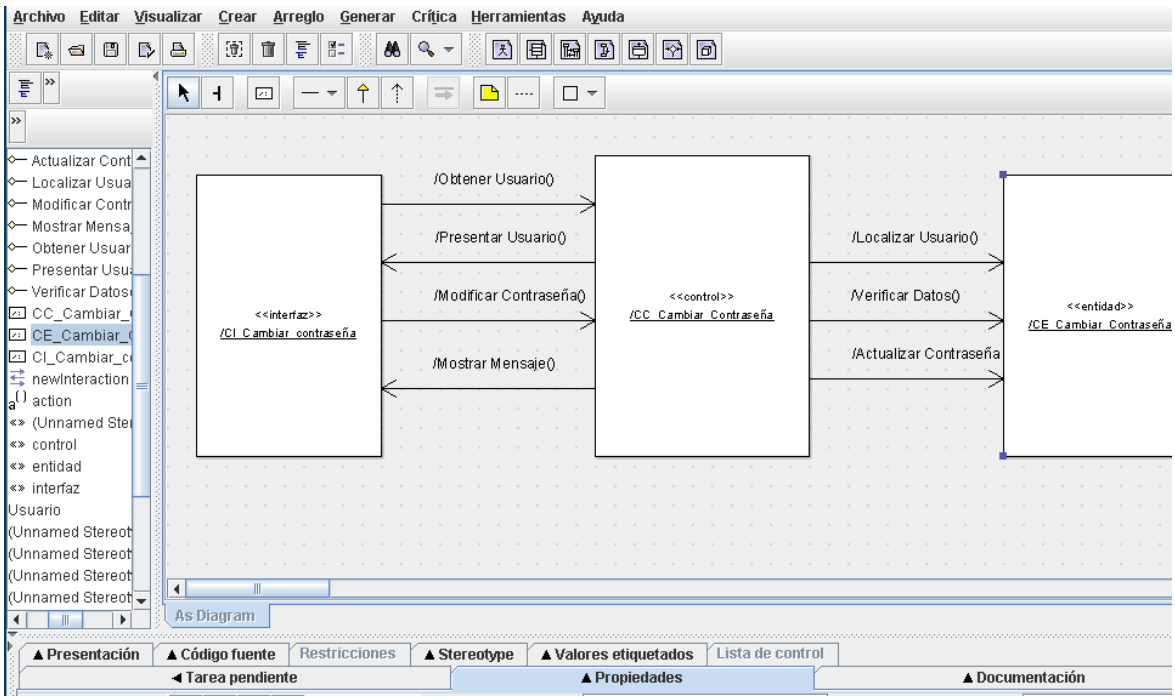
**Anexo 10: Código generado por AndroMDA, visto desde el Eclipse.**



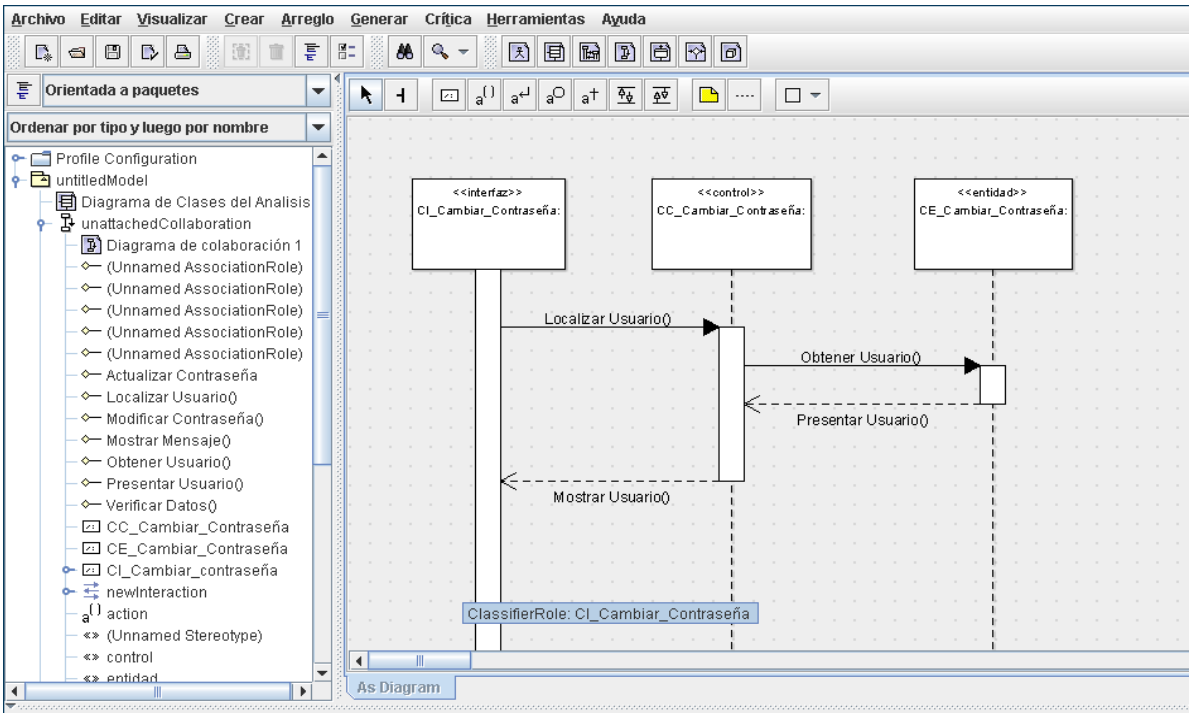
**Anexo 11: Diagrama de clases del análisis realizado con ArgoUML por el proyecto ROXS.**



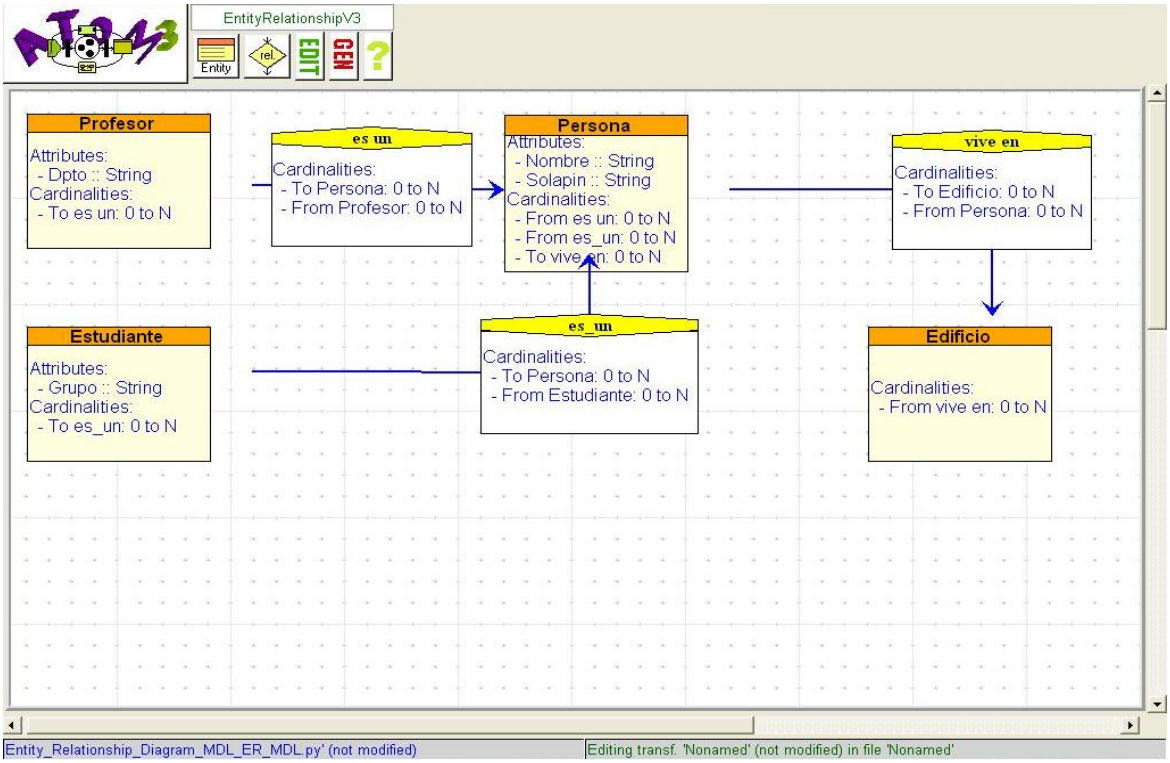

**Anexo 12: Diagrama de colaboración realizado con ArgoUML por el proyecto ROXS.**



**Anexo 13: Diagrama de secuencia realizado con ArgoUML por el proyecto ROXS.**



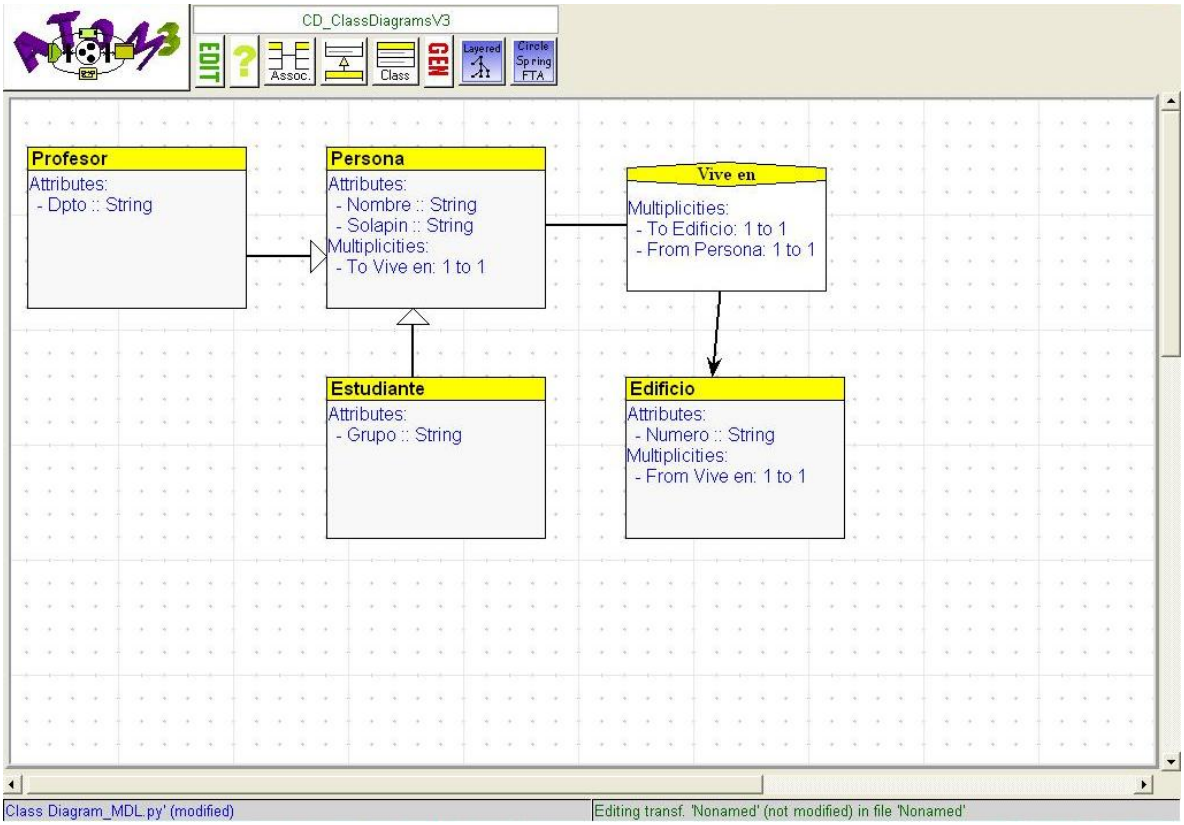

**Anexo 14: Diagrama entidad relación realizado con AToM3 por el proyecto Repositorio Semántico.**



**Anexo 15: Fragmento de código Python generado por AToM3 en la realización del diagrama entidad relación por el proyecto Repositorio Semántico.**

```
from ATOM3MSEnum import *

def Entity_Relationship_Diagram_MDL_ER_MDL(self, rootNode, EntityRelationshipV3RootNode=None):

    # --- Generating attributes code for ASG EntityRelationshipV3 ---
    if( EntityRelationshipV3RootNode ):
        # attributes
        EntityRelationshipV3RootNode.attributes.setActionFlags([ 1, 1, 1, 0])
        lcobj1 =[]
        cobj1=ATOM3Attribute(self.types)
        cobj1.setValue(('name', 'String', None, ('Key', 1), ('Direct Editing', 1)))
        cobj1.initialValue=ATOM3String('', 20)
        cobj1.isDerivedAttribute = False
        lcobj1.append(cobj1)
        cobj1=ATOM3Attribute(self.types)
        cobj1.setValue(('author', 'String', None, ('Key', 0), ('Direct Editing', 1)))
        cobj1.initialValue=ATOM3String('Annonymous', 20)
        cobj1.isDerivedAttribute = False
        lcobj1.append(cobj1)
        cobj1=ATOM3Attribute(self.types)
        cobj1.setValue(('description', 'Text', None, ('Key', 0), ('Direct Editing', 1)))
        cobj1.initialValue=ATOM3Text('\n', 60,15 )
```

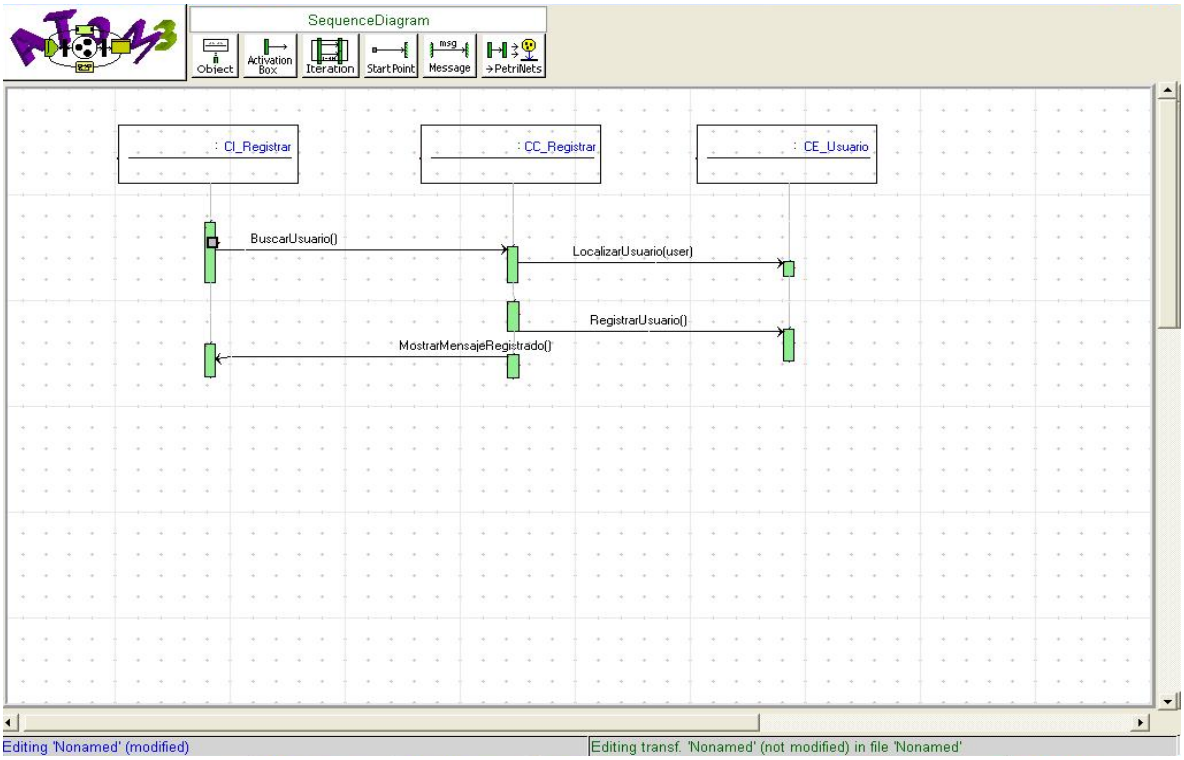
**Anexo 16: Diagrama de clases realizado con AToM3 por el proyecto Repositorio Semántico.**



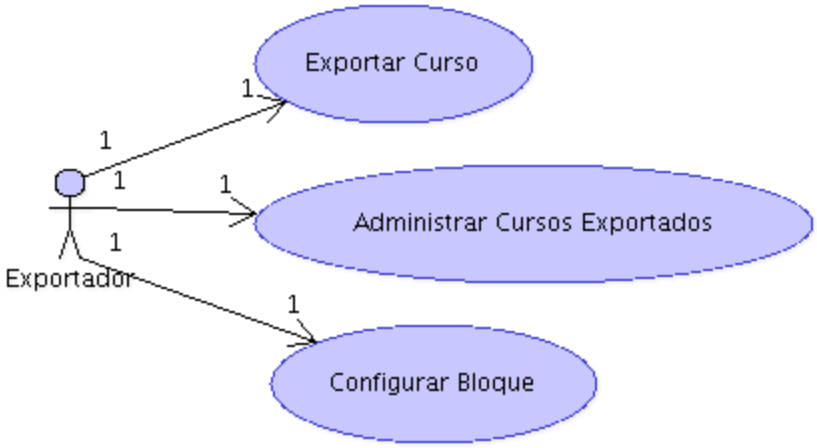
**Anexo 17: Fragmento de código Python generado por AToM3 en la realización del diagrama de clases por el proyecto Repositorio Semántico.**

```
from ATOM3Port import *
from ATOM3MSEnum import *

def Class Diagram_MDL(self, rootNode, CD_ClassDiagramsV3RootNode=None):

    # --- Generating attributes code for ASG CD_ClassDiagramsV3 ---
    if( CD_ClassDiagramsV3RootNode ):
        # name
        CD_ClassDiagramsV3RootNode.name.setValue('')
        CD_ClassDiagramsV3RootNode.name.setNone()

        # author
        CD_ClassDiagramsV3RootNode.author.setValue('Annonymous')

        # showCardinalities
        CD_ClassDiagramsV3RootNode.showCardinalities.setValue((None, 1))
        CD_ClassDiagramsV3RootNode.showCardinalities.config = 0

        # showAssociationBox
        CD_ClassDiagramsV3RootNode.showAssociationBox.setValue((None, 1))
        CD_ClassDiagramsV3RootNode.showAssociationBox.config = 0
```

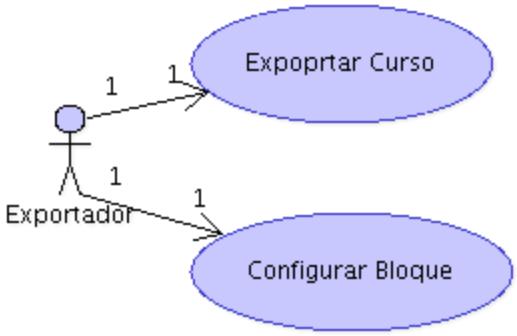
**Anexo 18: Diagrama de secuencia realizado con AToM3 por el proyecto Repositorio Semántico.**



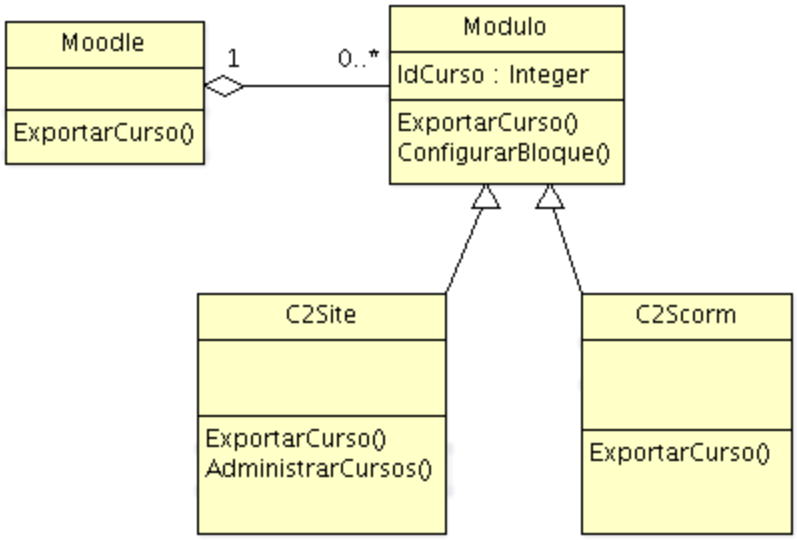
**Anexo 19: Diagrama de caso de uso no.1 realizado con ArgoUML por el proyecto C2Site & C2SCORM.**




**Anexo 20: Diagrama de caso de uso no.2 realizado con ArgoUML por el proyecto C2Site & C2SCORM.**



**Anexo 21: Diagrama de clases realizado con ArgoUML por el proyecto C2Site & C2SCORM.**




**Anexo 22: Fragmento de código en C++ generado con ArgoUML por el proyecto C2Site & C2SCORM.**

```
#ifndef Modulo_h
#define Modulo_h

class Moodle;

class Modulo {

 public:

    virtual void ExportarCurso();

    virtual void ConfigurarBloque();

 public:
    Integer IdCurso;

 public:

    Moodle *myMoodle;

    Moodle *myMoodle;

    Moodle *myMoodle;

};

#endif // Modulo_h
```

# GLOSARIO DE TÉRMINOS

**API:** una API (del inglés Application Programming Interface - Interfaz de Programación de Aplicaciones) es el conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

**Cartucho:** representan un módulo de generación de código para AndroMDA; son el principal complemento de esta herramienta, además le proporcionan la capacidad de procesar los elementos del modelo que se desea interpretar.

**EMF:** Eclipse Modeling Framework es un framework de modelado y facilidad de generación de código para construir herramientas y otras aplicaciones basadas en un modelo de datos estructurado.

**EXCEL:** mejor conocido sólo como Microsoft Excel, es una aplicación para manejar hojas de cálculos. Este programa fue y sigue siendo desarrollado y distribuido por Microsoft, y es utilizado normalmente en tareas financieras y contables.

**Framework:** conjunto de APIs y herramientas destinadas a la construcción de un determinado tipo de aplicaciones de manera generalista.

**Hibernate:** Herramienta de Mapeo objeto-relacional para la plataforma Java (y disponible también para .Net con el nombre de NHibernate) que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) que permiten establecer estas relaciones.

**Java Development Kit o (JDK):** es un software que provee herramientas de desarrollo para la creación de programas en java.

**Java EE:** Java Platform, Enterprise Edition (anteriormente conocido como Java 2 Platform, Enterprise Edition o J2EE hasta la versión 1.4), es una plataforma de programación—parte de la Plataforma Java—para desarrollar y ejecutar software de aplicaciones en Lenguaje de programación Java con arquitectura de N niveles distribuida, basándose ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones.

**JSF:** JavaServer Faces es un framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE. **JSF** usa

JavaServer Pages (JSP) como la tecnología que permite hacer el despliegue de las páginas, pero también se puede acomodar a otras tecnologías como XUL.

**LaTeX:** es un lenguaje de marcado para documentos, y un sistema de preparación de documentos, formado por un gran conjunto de macros. Es muy utilizado para la composición de artículos académicos, tesis y libros técnicos, dado que la calidad tipográfica de los documentos realizados con LaTeX es comparable a la de una editorial científica de primera línea. LaTeX es software libre bajo licencia LPPL.

**MATLAB:** (abreviatura de *MATrix LABoratory*, "laboratorio de matrices") es un software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Está disponible para las plataformas Unix, Windows y Apple Mac OS X. Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware.

**Micro-kernel:** Micronúcleo (en inglés: microkernel) es un tipo de núcleo de un sistema operativo que provee un conjunto de primitivas o llamadas al sistema mínimas, para implementar servicios básicos como espacios de direcciones, comunicación entre procesos y planificación básica.

**Moodle:** *Modular Object-Oriented Dynamic Learning Environment* (Entorno de Aprendizaje Dinámico Modular Orientado a Objetos) herramienta para producir cursos basados en internet y páginas web.

**PGML:** (Precision Graphics Markup Language) es un lenguaje basado en XML para la representación de gráficos vectoriales **.**

**PHP:** lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en interpretación del lado del servidor.

**Java:** es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems para aplicaciones software independiente de la plataforma.

**HTML:** HTML, siglas de HyperText Markup Language (Lenguaje de Marcas de Hipertexto), es el lenguaje de marcado predominante para la construcción de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de

"etiquetas", rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir un script (por ejemplo Javascript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML.

.**NET:** es un proyecto de Microsoft para crear una nueva plataforma de desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones.

**Plug-in:** es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica.

**Python:** lenguaje de programación de propósito general, orientado a objetos.

**SAPO-XP:** software desarrollado generalmente con herramientas libres que están aún en la fase de construcción.

**SCORM:** estándar para crear/ empaquetar objetos.

**SGL:** el Lenguaje de consulta estructurado (en inglés **S**tructured **Q**uery **L**anguage) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en éstas. Una de sus características es el manejo del álgebra y el cálculo relacional permitiendo efectuar consultas con el fin de recuperar -de una forma sencilla- información de interés de una base de datos, así como también hacer cambios sobre ella.

**Spring:** framework de código abierto de desarrollo de aplicaciones para la plataforma Java.

**Struts:** es una herramienta de soporte para el desarrollo de aplicaciones Web bajo el patrón MVC bajo la plataforma J2EE. Su carácter de "*software libre*" y su compatibilidad con todas las plataformas en que Java Entreprise esté disponible, lo convierte en una herramienta altamente disponible. Con la versión 2 del framework se introdujeron algunas mejoras sobre la primera versión, de cara a simplificar las tareas más comunes en el desarrollo de aplicaciones web, así como mejorar su integración con AJAX, etc.

**SVG:** (Scalable Vector Graphics) es un lenguaje para describir gráficos vectoriales bidimensionales, tanto estáticos como animados.

**Tk/tcl:** Tk, del inglés *Tool Kit*, es una extensión para el lenguaje script Tcl. Tcl, originado del acrónimo en inglés "*Tool Command Language*" o lenguaje de herramientas de comando. La combinación de Tk con Tcl se utiliza para la creación de interfaces gráficas.

**VHDL:** Es un lenguaje usado por ingenieros definido por el IEEE (*Institute of Electrical and Electronics Engineers*) que se usa para diseñar circuitos digitales.