Una Propuesta de Proceso Explícito de V&V en el Marco de MDA *

Francisco Javier Lucas Martínez, Fernando Molina Molina, Ambrosio Toval Álvarez
Grupo de Ingeniería del Software
Dept. de Informática y Sistemas
Universidad de Murcia
{fjlucas,fmolina,atoval}@um.es

Resumen

La propuesta MDA del OMG está teniendo un gran auge en los últimos años. A pesar de la numerosa documentación existente sobre ella, todavía existen aspectos no definidos, definidos de forma imprecisa o a los que todavía no se les ha prestado la suficiente atención. Entre estos aspectos, se ha detectado la ausencia de metodologías para el desarrollo software basadas en MDA, la cual ha sido suplida con la definición de algunas de ellas. A pesar de ello, en todas estas metodologías se ha detectado la ausencia de una definición clara y explícita de una de las actividades más cruciales: la validación y verificación de modelos, que mejore la calidad de los modelos construidos. En este trabajo se propone cubrir esta carencia con la definición de un proceso genérico de V&V de modelos enmarcado en el ciclo de desarrollo de software bajo la propuesta de MDA.

1. Introducción y Motivación

En los últimos años, la propuesta *Model Driven Architecture* (MDA) [1] promovida por el OMG está teniendo un gran auge. Solamente tenemos que prestar atención a las numerosas publicaciones y congresos relacionados con MDA para darnos cuenta del fuerte impacto que ha provocado la propuesta en la comunidad del Desarrollo de Software. Son muchos los grupos de investigación que están dirigiendo su trabajo en este sentido y muchas las empresas que han visto en MDA un marco adecuado para la gestión de sus procesos de desarrollo software o, simplemente, han visto el fuerte auge de MDA como una buena oportunidad de negocio, lanzando al mercado productos bajo la leyenda *MDA Compliance* [2].

Sin embargo, si analizamos detenidamente la numerosa documentación existente relacionada con MDA, observamos que muchos de los conceptos y actividades relacionados con MDA no están claramente definidos, se definen de forma vaga e imprecisa o son interpretados de diferentes formas, según el autor al que acudamos. En principio, podría suponerse que acudiendo a las fuentes oficiales de MDA, como la *MDA Guide* [1], se podrían obtener soluciones a estos problemas. Sin embargo, en su estado actual, como se comenta en [2, 3], la *MDA Guide* es insuficiente para construir sistemas completos, define la mayoría de conceptos importantes de la propuesta MDA de forma imprecisa, dando pie a la posibilidad de múltiples interpretaciones de los mismos, algo que resulta paradójico en el marco de MDA, que precisamente hace hincapié en el uso de estándares (como UML, XMI, o QVT) como mecanismo para asegurar la coherencia de la propuesta [3].

De la misma forma que se han detectado las

*Financiado por el Ministerio Español de Ciencia y Tecnología, proyecto DYNAMICA/PRESSURE TIC 2003-07804-C05-05

imprecisiones comentadas, analizando la documentación también se detecta la ausencia de tratamiento o la escasa importancia que se presta a determinados aspectos que consideramos importantes. En particular, pensamos que no se da la importancia que merecen a los aspectos relacionados con las importantes actividades de validación y verificación de modelos. Por todos es aceptado que MDA propone que los modelos, en sus diferentes niveles de abstracción, sean la piedra angular que de soporte a toda la propuesta. Se propone utilizar los modelos como mecanismo que conducirá todo el proceso de desarrollo software, desde la construcción de los modelos independientes de la plataforma (CIM y PIM) hasta los modelos dependientes de la plataforma (PSM) y la generación automática de código a partir de los mismos.

Hasta ahora, la necesidad de disponer de modelos completos, consistentes y precisos era muy importante cuando se construía software. Sin embargo, pensamos que esta necesidad cobra especial relevancia en el ámbito de MDA, en el que todas las actividades están dirigidas por los modelos. La V&V de modelos puede ser usada para detectar errores e inconsistencias en las etapas tempranas del desarrollo, evitando así la propagación de esos errores a las etapas posteriores. Si en la etapa de modelado independiente de la plataforma se construyeran modelos incompletos, imprecisos o inconsistentes se produciría una propagación de estos errores al resto de etapas, especialmente en MDA, ya que se propone conseguir una transformación automática entre modelos y la generación automática de código a partir de los modelos obtenidos. Para evitar confusión en lo que se entiende por validación y verificación en este trabajo, el significado de ambos conceptos será explicado en la sección 3.

Por otro lado, el estándar MDA tampoco describe una metodología específica que guíe el proceso de desarrollo de proyectos software bajo el marco MDA. Para intentar suplir esta carencia, se han publicado unas cuantas metodologías para el desarrollo de software basadas en MDA o en otros enfoques MDD (*Model Driven Developement*) o MDE (*Model Driven Engineering*). Algunas de estas metodologías serán analizadas en la sección 2. Como se verá en esta sección, dichas metodologías no suelen incluir una fase de verificación y/o validación de los modelos implicados en el desarrollo o, en caso de incluirla, se reduce a una descripción muy breve en la que no se especifican claramente las actividades que habría que realizar para llevar a cabo dicha fase. Este trabajo intenta cubrir esta carencia detectada en las distintas metodologías que se están publicando, proponiéndose un proceso de V&V genérico y que pueda aplicarse en el marco de metodologías basadas en MDA o MDD, siendo ésta la principal contribución del mismo.

La estructura del resto del trabajo es la siguiente: en la sección 2 se analizarán algunas propuestas de metodologías de desarrollo basadas en MDA y las carencias que muestran en cuanto a las fases de V&V. En la sección 3, se explicará en qué consisten los conceptos y el proceso de V&V que se propone en este trabajo. Por último, se mostraran las conclusiones y los trabajos futuros que podrían realizarse relacionados con éste.

2. Metodologías y Procesos Existentes para MDA

Existen distintas metodologías y procesos publicados para el desarrollo de proyectos basadas en MDA o en MDD, pero en la mayoría de ellas no se contempla expresamente como parte del proceso de desarrollo la validación y verificación de los modelos construidos, o bien si se contempla, a esta fase no se le da la importancia que tiene tratándola solo parcialmente.

Algunas de estas metodologías están orientadas a dominios concretos, como la presentada en [4]. En este trabajo se presenta una metodología basada en RM-ODP (*Reference Model of Open Distributed Processing*), el modelo de referencia de ISO para el desarrollo de aplicaciones abiertas y distribuidas. La metodología presentada se centra en aplicaciones de este ámbito y, en ella, no se muestran actividades dedicadas a la V&V de los modelos que guían el proceso de desarrollo.

En [5] es presentada otra metodología. En ella se describe un flujo de trabajo en el que se integran todas las actividades implicadas en el desarrollo de software, desde la captura de requisitos hasta el despliegue de la aplicación. En este trabajo se incluye una fase de *testing*, aunque ésta se realiza una vez que el programa está codificado, por lo que la corrección de los fallos detectados en esta fase será mucho más costosa que si estos fallos se hubieran detectado antes.

Otra de las investigaciones que se están llevando a cabo y que parece tener más respaldo es la presentada en [6]. En este trabajo se identifican distintas categorías de usuarios involucrados en el desarrollo, así como una descripción de las fases que forman parte del mismo. En esta metodología sí se contempla, en la fase de ejecución, una actividad explícita de validación y verificación para garantizar que los productos de la actividad de modelado cumplen los requisitos especificados y están libres de fallos. Aunque la metodología sí incluye una actividad de V&V, ésta sólo se describe superficialmente y no se dan criterios o guías concretas para llevarla a cabo.

En [7], se esboza de forma muy simplificada como podrían integrarse algunas actividades de verificación en el proceso de desarrollo software. Nuestro trabajo analiza con más detalle las actividades que incluirían la fase de verificación y, además, se incluyen actividades de simulación y validación a nivel de modelos, las cuales no son consideradas en [7].

Como se ha mostrado, las metodologías que están surgiendo no incluyen las actividades de V&V para los modelos construidos o éstas no son tratadas con la importancia que merecen. Este trabajo intenta completar esta carencia, desarrollando un proceso de V&V genérico, que puede integrarse dentro de cualquier metodología para el desarrollo basado en MDA o MDD.

3. Proceso de Validación y Verificación

Antes de comenzar con la descripción del proceso, por razones de claridad y dado que no existe un acuerdo claro entre autores con respecto a la definición y significado de lo que se entiende por validación y verificación de modelos, vamos a comenzar definiendo qué entendemos por cada una de estas actividades. A lo largo de este trabajo seguiremos las siguientes definiciones:

- La verificación intenta mostrar que un modelo satisface una propiedad específica [8]. Generalmente, esto puede realizarse sin ejecutar el modelo. Esta verificación puede llevarse a cabo formal o informalmente. *Checklist* y algoritmos que analizan un modelo en busca de situaciones peligrosas o con errores son técnicas de verificación informal. La demostración de teoremas y el *model checking* son técnicas de verificación formal.

- La validación evalúa si el comportamiento observable del modelo se ajusta a los requisitos (así se define la validación de un producto en [8]). Respecto a modelos UML, esta validación suele realizarse a través de la simulación de escenarios y casos de uso.

La verificación puede llevarse a cabo de distintas formas:

- Inspección realizada por humanos leyendo el modelo. Es una revisión pobre, aunque es la que se usa normalmente.

- Verificación estática. Para ello, generalmente, se usan algoritmos específicos que buscan errores concretos que suelen cometerse con frecuencia o inconsistencias sintácticas o semánticas en el modelo.

- Demostraciones formales basadas en el uso de lenguajes formales, los cuales usan resultados que provienen de teorías matemáticas y lógicas.

Una vez que se ha establecido el significado de estos conceptos, vamos a comenzar con la descripción del proceso que se propone.

3.1. Ciclo de Vida General de MDA

Con el fin de ubicar la fase de V&V propuesta, en la Figura 1 mostramos, de forma general, las principales etapas que componen el ciclo de vida para el desarrollo basado en MDA, que ha sido adaptado del ciclo que aparece en [6]. Debemos aclarar que, aunque la actividad de validación y verificación debe llevarse a cabo a lo largo de todo el ciclo de vida, para centrar claramente esta propuesta, en la Figura 1 no se muestran otras fases de V&V aparte de las que se realizan directamente sobre los modelos. Para completar esta figura sería necesario añadir otras actividades de V&V al ciclo de desarrollo.

La primera etapa corresponde a la fase de *análisis de requisitos*. Su objetivo es obtener una especificación adecuada de cuáles son los requisitos del sistema que se desea desarrollar. Consideramos que ésta es una de las fases más importantes, ya que disponer de un documento de requisitos completo y preciso evitará propagar errores a las siguientes fases del ciclo de vida. Hay multitud de referencias en la literatura justificando y proporcionando guías y métodos para abordar esta tarea. Una vez obtenidos los requisitos del sistema, se pasará a la fase de *modelado independiente de la plataforma*, en la que se construirán los modelos que describen la funcionalidad del sistema de forma independiente a la plataforma elegida para la implementación final del mismo.

Una vez realizadas las dos etapas anteriores, creemos que es el punto más adecuado en el que incluir la fase de validación y verificación de los modelos construidos. La situación de esta fase viene motivada porque, según la propuesta MDA, la transformación de modelos PIM a PSM e incluso la generación de código debe realizarse de forma (semi-)automática. Si estas transformaciones y la generación de código son definidas correctamente, no deberían añadir errores a los modelos del sistema. Esto motiva que las actividades de V&V se centren especialmente en los modelos PIM.

Esta fase recibiría como entrada los modelos PIM construidos en la etapa anterior y se encargaría de verificar y validar su corrección, precisión, completitud, etc. Toda la informa-

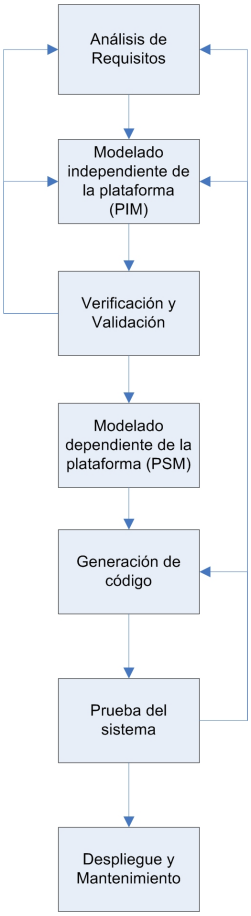

Figura 1: Ciclo de Vida Genérico para MDA

ción detectada en esta fase puede servir como realimentación a la fase de *modelado independiente de la plataforma*, y también puede ayudar a detectar errores en los requisitos, con el objetivo de no propagar errores en los modelos a las siguientes etapas del ciclo de vida.

Finalizada la fase de validación y verificación, que comentaremos detalladamente más adelante, dispondremos de modelos PIM correctos que se transformarán, si es posible de forma automática, en *modelos dependientes de la plataforma*. Estos modelos serán usados como entrada para la fase de *generación de código*.

Por último, quedaría realizar la fase de *pruebas del sistema* (si los resultados obtenidos son erróneos, pueden implicar el regreso a la fase anterior de generación de código o a la de modelado de los PIM o incluso la modificación de determinados requisitos) y, por último, la implantación y mantenimiento del sistema.

Finalmente, debemos comentar que, aunque en la Figura 1 la fase de V&V aparece después de la construcción de los modelos PIM, algunas de sus actividades podrían comenzar durante la construcción de dichos modelos. Las propias herramientas de modelado podrían ayudar a la construcción de modelos correctos realizando sugerencias al modelador o recomendándole buenas prácticas aceptadas por la comunidad de desarrollo de software. Herramientas como *ArgoUML* [9] realizan lo comentado durante la construcción de modelos UML.

Una vez ubicada la fase de V&V que proponemos para el marco de MDA, en la siguiente sección se explicarán de forma más detallada las distintas actividades necesarias para llevarla a cabo.

3.2. Fase de Validación y Verificación

En esta sección nos centraremos en la *fase de verficación y validación* de los modelos que se han construido durante la fase anterior. En la Figura 2, en la que se ha utilizado la notación SPEM (Software Process Engineering Metamodel) [10], se muestran las actividades que formarían esta fase.

La primera etapa a realizar en la fase de V&V consistiría en identificar propiedades que deberían cumplir los modelos construidos (1 en la Figura 2). Las propiedades a identificar pueden estar relacionadas con diferentes ámbitos. Por ejemplo, pueden ser propiedades relacionadas con la semántica estática de los modelos de acuerdo con las reglas de su metamodelo (es decir, con cómo una instancia de un constructor se conecta con otras instancias), propiedades relacionadas con la semántica dinámica de los modelos (comprobar que los modelos cumplen con las *well-formedness rules* definidas por su metamodelo), propiedades relacionadas con la consistencia dentro de un modelo o entre modelos y, también, propiedades relacionadas con la semántica concreta del dominio al que pertenecen los modelos.

A continuación, o bien en paralelo con la actividad anterior, se realizará una actividad de especificación/implementación del metamodelo (3 en la Figura 2), que nos dará como resultado una representación del mismo en un determinado lenguaje. Esta implementación está motivada por la necesidad de poder usar los modelos concretos que definen el sistema en las actividades de validación y verificación, las cuales pueden requerir para su realización una implementación para el metamodelo distinta a la usada en el PIM origen. Para realizar esta implementación se utilizará como entrada el metamodelo, sobre el que se definen modelos, y podrá llevarse a cabo tanto usando las técnicas/lenguajes habituales de programación (Java, ...), como técnicas formales, basados en formalismos matemáticos [11, 12, 13, 14]. Cualquiera que sea la técnica elegida para realizar la verificación, debería tener soporte automático. Esta actividad produciría como salida una representación del metamodelo utilizable en el resto de actividades (PIM_R).

Una vez identificadas las propiedades, la siguiente actividad se encargará de su implementación (2 en la Figura 2) pudiéndose utilizar, como en la etapa anterior, técnicas formales o no formales. Esta actividad, además de las propiedades identificadas, hace uso de la representación del metamodelo creado en la

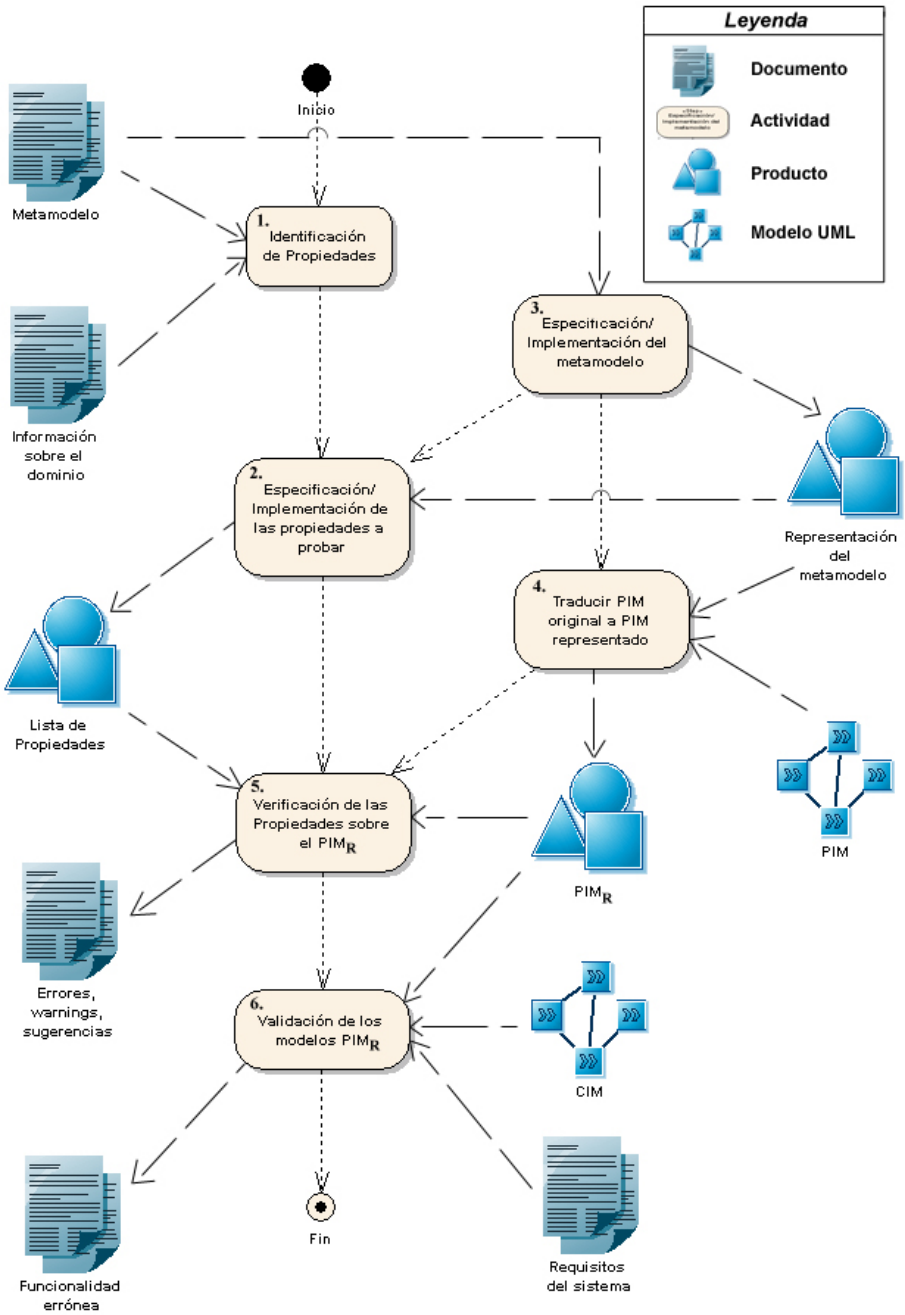

Figura 2: Fase de Validación y Verificación

actividad 3.

La siguiente actividad sería traducir el PIM original, obtenido en la fase anterior, a la representación que acabamos de crear (4 en la Figura 2). De esta forma, todos los elementos del PIM original podrán ser accesibles en las actividades de V&V. La salida de esta actividad es un nuevo PIM, al que denominamos PIM_R, representado usando el metamodelo implementado en la actividad anterior. En este punto conviene decir, que las actividades 3 y 4 pueden no ser necesarias si la técnica usada para la verificación usa la misma representación que el PIM original.

A continuación, pasaríamos a la etapa de verificación de las propiedades sobre el PIM (5 en Figura 2). Esta etapa recibe como entrada la lista de propiedades implementadas en las actividades anteriores y los modelos PIM_R sobre los que se quiere realizar la verificación de las mismas. Como salida de esta etapa obtendremos información sobre errores detectados en los modelos o sugerencias que nos permitan mejorarlos en algún sentido (métricas, ...). Toda esta información puede ser utilizada para mejorar los modelos PIM que teníamos inicialmente.

Un modelo que supere las comprobaciones comentadas pasaría a la siguiente actividad: la *validación de modelos* (6 en la Figura 2). Los modelos se validan después de ser verificados. Así se evitan posibles errores en la validación del sistema que vengan dados como consecuencia de errores de modelado que podrían eliminarse en la fase de verificación. En esta última actividad se realizaría una simulación de los modelos creados. Esta simulación suele realizarse mediante animación de diagramas de estados o secuencia o usando lenguajes de simulación (como ASL: *Action Semantics Language*). Esta actividad tiene como entrada, además de los modelos PIM_R a simular, los modelos CIM e información sobre los requisitos del sistema. Esto nos permitirá, por un lado, comprobar que el funcionamiento de los modelos es coherente respecto de lo que indican los requisitos o los modelos CIM y, por otro lado, comprobar si la ejecución de un determinado escenario deja el sistema en un estado inconsistente. Para la correcta realización de esta actividad será necesario realizar un estudio previo de las casos de prueba que se deben simular con el propósito de cubrir el mayor número de escenarios posibles.

Existen algunas herramientas en el mercado que permiten la simulación de algunos de los diagramas UML que se usan normalmente en el desarrollo de sistemas (STATEMATE [15],...). La salida de esta actividad también debe ser usada para mejorar los modelos del sistema. Para integrar todas estas actividades en la herramienta de modelado es necesario que ésta ofrezca soporte para el acceso a los elementos del metamodelo a través de algún mecanismo de extensión de la herramienta (plugins,...). Una vez que los modelos hayan superado correctamente esta fase de V&V, ya estarían preparados para pasar a la siguiente fase del ciclo de vida. Si obligamos a los modelos a pasar con éxito esta etapa de V&V reduciremos notablemente el número de errores e inconsistencias que se propagan a las siguientes fases.

4. Conclusiones y Trabajos Futuros

En este trabajo se presenta un proceso de V&V genérico, que puede ser utilizado en el ámbito de una metodología de desarrollo basada en MDA o MDD. Para ello, se ha establecido el lugar más adecuado para situar esta fase de V&V dentro del ciclo de vida de desarrollo, y se han definido las actividades que habría que realizar dentro de esta fase para garantizar que los modelos construidos tienen menos errores y son más correctos.

Este proceso de V&V es especialmente importante en MDA o MDD. Si hasta ahora la corrección y precisión de los modelos construidos era importante, con la aparición del desarrollo dirigido por modelos ha ganado todavía más importancia, ya que los modelos guían todo el proceso de desarrollo y un fallo en un modelo PIM será arrastrado al PSM y al código generado a partir de este último. Además, el proceso definido sirve para identificar funciones importantes que deberían cumplir las herramientas de soporte a MDA.

Como continuación a este trabajo se está validando el proceso propuesto con respecto a distintas metodologías publicadas, así como con algunas herramientas de modelado basadas en MDA, para lo cuál se está siguiendo la filosofía del método investigación en acción. De esta forma se pretende, por un lado, demostrar la genericidad de este proceso y su posible aplicación en muchos ámbitos y, por otro lado, mejorar esta fase de V&V con la información que se obtenga durante este proceso.

Agradecimientos

Agradecemos a los revisores anónimos la lectura crítica de este trabajo y los valiosos comentarios que nos brindaron.

Referencias

[1] OMG: MDA Guide Version 1.0.1, http://www.omg.org/mda. (2001)

[2] Brown, A.W.: Model Driven Architecture: Principles and Practice. Software and Systems Modeling. Volume 3, Number 4 (2004)

[3] Muñoz, J., Pelechano, J.: MDA a Debate. Actas del Primer Taller sobre Desarrollo Dirigido por Modelos, MDA y Aplicaciones (DSDM'04) (2004)

[4] Gervais, M.P.: Towards an MDA-Oriented Methodology. Proceedings of the 26th International Computer Software and Applications Conference on Prolonging Software Life: Development and Redevelopment table of contents. ISBN:0-7695-1727-7 (2002)

[5] Larrucea, X., García Díez, A.B., Mansell, J.X.: Practical Model Driven Development Process. Second European Workshop on Model Driven Architecture (MDA) with an emphasis on Methodologies and Transformations (2004)

[6] Gavras, A., Belaunde, M., Ferreira Pires, L., Almeida, J.P.A.: Towards an MDA-Based Development Methodology. Software Architecture: First European Workshop, EWSA 2004, St Andrews, UK, May 21-22, 2004. Proceedings (2004)

[7] de Lara, J., Guerra, E., Vangheluwe, H.: Model-Based Development: Meta-Modelling, Transformation and Verification. Chapter of the book 'Management of the Object-Oriented Development Process', Idea Group Publishers, edited by Liping Liu y Boris Roussev (2004)

[8] Binder, R.: Testing object-oriented systems: models, patterns and tool Massachusetts. The Addison-Wesley object technology series. A. Wesley. (1999)

[9] Tigris.org: ArgoUML Modelling tool. http://argouml.tigris.org (2004)

[10] OMG: Object Management Group. Software Process Engineering Meta-model V1.1, Retrieved from: http://www.omg.org. (2005)

[11] Fernández Alemán, J.L., Toval Álvarez, A.: Improving System Reliability via Rigorous Software Modeling: The UML Case. Proceedings of the 2001 IEEE Aerospace Conference (track 10: Software and Computing), Montana, USA IEEE Computer Society (2001)

[12] Lucas Martínez, F.J., Toval Álvarez, A.: Formal Verification of Properties in the UML Collaboration Diagram. ICSSEA 2004: 3rd Workshop on SYSTEM TESTING AND VALIDATION. Paris (2004)

[13] Fernández Alemán, J.L., Toval Álvarez, A.: Can Intuition Become Rigorous? Foundations for UML Model Verification Tools. International Symposium on Software Reliability Engineering, Published by IEEE Press (2000)

[14] Whittle, J., Araújo, J., Toval Álvarez, A., Fernández Alemán, J.L.: Rigorously Automating Transformations of UML Behavior Models. Dynamic Behaviour in UML Models: Semantic Questions in conjunction with UML 2000 York, UK ,

ACM SIGSOFT, IEEE Computer Society (2000)

[15] Harel, D., Naamad, A.: The STATEMATE semantics of statecharts. ACM Transactions on Software Engineering and Methodology (TOSEM), Volume 5 Issue 4 (1996)