Integración de técnicas orientadas al cliente y técnicas formales en el desarrollo de software con UML y RUP

Favre, Liliana; Leonardi, Carmen; Mauco, Virginia; Felice, Laura; Pereira, Claudia; Martínez, Liliana
INTIA
Facultad de Cs. Exactas
UNCPBA

1. Introducción

En los últimos años UML ("*Unified Modeling Language*") se ha impuesto como un estándar de–facto para expresar modelos orientados a objetos. Es un lenguaje diseñado para especificar, visualizar, construir y documentar "artefactos" de sistemas de software [Booch'99].

A diferencia de los lenguajes visuales que lo precedieron, UML posee una definición semántica más precisa que combina notación gráfica, reglas bien formadas expresadas en OCL [OMG'01] y lenguaje natural. Esta definición semántica da una estructura rigurosa al lenguaje aunque aún varias de sus construcciones están definidas débilmente.

Si bien UML estandariza un lenguaje de modelamiento y no impone ningún proceso de desarrollo fue concebido pensando en procesos dirigidos por casos de uso, centrados en arquitecturas, iterativos e incrementales. En noviembre de 2001 OMG presenta "*Software Process Engineering Metamodel*" (SPEM) que es usado para describir un proceso de desarrollo de software concreto o una familia de procesos que usan UML. El proceso más popular que se ajusta a SPEM es RUP ("*Rational Unified Process*") [Krutchen'00].

La existencia de un lenguaje de modelamiento estándar como UML brinda la posibilidad de concentrar esfuerzos en la definición de potentes herramientas CASE UML. Pueden mencionarse entre las numerosas existentes en el mercado a Argo/UML, Together, GDPro, Stp/UML, Rational Rose, MagicDraw/UML, Rhapsody y Objecteering. Las mismas asisten en el análisis, diseño e implementación de sistemas orientados a objetos. Proveen facilidades, si bien limitadas, para homogeneizar diagramas y realizar comprobaciones que detecten inconsistencias y errores como asimismo para procesos de ingeniería directa (*forward engineering*) e inversa (*reverse engineering*).

Estas herramientas evolucionan constantemente, con el objeto de cubrir aspectos no considerados y así brindar asistencia en las distintas etapas del proceso de desarrollo de software. Por ejemplo, las mismas aún no brindan una completa asistencia para la construcción de modelos del negocio durante las primeras etapas de desarrollo en donde la interacción con los *stakeholders* es crucial. Por otro lado, la falta de una semántica formal para UML dificulta la validación de modelos UML y el análisis de consistencia entre las diferentes vistas del sistema. Ésto crea dificultades en el desarrollo de sistemas complejos y críticos donde es crucial realizar un análisis riguroso de las propiedades expresadas por los modelos y detectar ambigüedades e inconsistencias desde los primeros niveles de un diseño. Asimismo, los procesos de generación de código son bastante limitados y no permiten transformar toda la información registrada en los modelos UML. La ingeniería inversa soporta un subconjunto de la notación UML y requiere de la habilidad del programador para mantener la integridad entre modelos UML y el código. Tampoco brindan asistencia para la optimización de modelos (*refactoring*).

Teniendo en cuenta los aspectos mencionados anteriormente, se propone en esta investigación definir las bases para extender la funcionalidad de las CASE UML existentes. Este proyecto pretende definir modelos y estrategias que partiendo desde el modelo de negocios permitan obtener código orientado a objetos. Para esto se presenta la incorporación de modelos de requisitos orientados al cliente y un conjunto de heurísticas para poder guiar la construcción del modelo de negocios propuesto por RUP, más concretamente los *Business Use Cases y* el Modelo de Objetos

del Negocios. Asimismo, se presenta la integración de UML con técnicas formales que permitiría el uso de verificación formal y la definición de procesos rigurosos de ingeniería directa e inversa que tiendan a la automatización facilitando el reuso y la evolución del software.

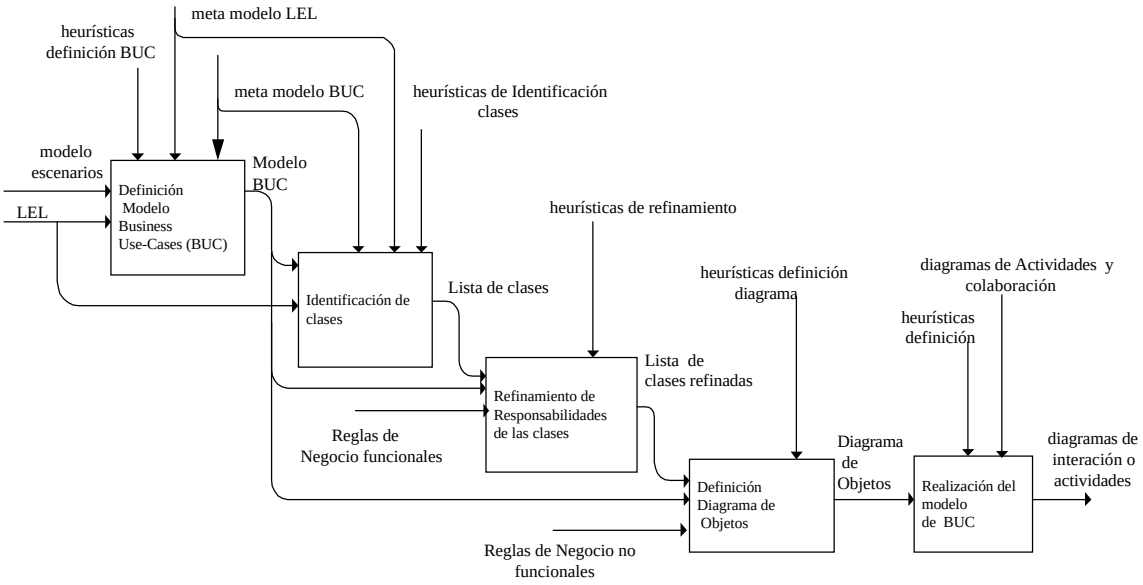
Se describen en la sección 2 la propuesta para la definición de modelos del negocio a partir de modelos de requisitos orientados al cliente. La sección 3 describe las bases de la integración de UML con técnicas formales y los resultados logrados en cuanto a la definición de procesos rigurosos de generación de código. La sección 4 presenta conclusiones y extensiones de esta investigación.

2. Definición del Modelo de Negocios de RUP a partir de modelos orientados al cliente

"*Se puede considerar al problema del desarrollo de software como un problema de construir un artefacto. Este artefacto será instalado en el mundo con el cual va a interactuar* [Jackson'95]". Esa parte del mundo en la cual los efectos del artefacto serán sentidos, evaluados y aprobados en caso de éxito, se *denomina Dominio de Aplicación* o *Universo de Discurso, UofD* [Leite'95]. El *UofD* es el contexto general en el cual el software será desarrollado, operado y mantenido. Incluye todas las fuentes de información y personas o sectores relacionados con la aplicación. El *UofD* es donde se originan los requisitos [Jackson'95; Jacobson'99], por lo que, sino se define apropiadamente no será posible enfocarse en los mismos. Con este objetivo, se propone para esta etapa modelar el *UofD* a partir de modelos en lenguaje natural y orientados al cliente, favoreciendo la interacción con el cliente, muy importante en esta etapa. Estos son: el Modelo Escenarios para definir el comportamiento del *UofD* [Leite´97], el modelo del Léxico Extendido del Lenguaje (LEL) que describe el vocabulario del mismo [Leite´95], y finalmente el Modelo de Reglas de Negocio que describe las políticas de la organización [Leonardi´98, Leite´98].

A partir de estos modelos se pueden derivar un modelo conceptual de objetos que permite a los ingenieros de software visualizar el *UofD* en términos de los conceptos que se manejarán a lo largo de todo el proceso de desarrollo, haciendo que la transición desde el modelado conceptual al diseño e implementación sea un proceso más natural. En el contexto de RUP y UML este modelo es el modelo de Negocios.

Manipular esta información para obtener un modelo conceptual de objetos no es una tarea trivial, ya que debe filtrarse y modelarse en términos de los conceptos que se manejarán en la solución, en nuestro caso, objetos y relaciones. La definición de clases es un proceso dual, así como se definen clases determinando abstracciones relevantes para el sistema, es igualmente importante poder descartar clases. Por esta razón, es crucial poder manipular la información de los modelos basados en lenguaje natural para poder definir y descartar las abstracciones. Por esta razón, en esta etapa se propone una estrategia basada en heurísticas que asista al ingeniero en la construcción del modelo de negocios a partir de los modelos de requisitos. El uso de las heurísticas mejora los aspectos de *traceability* entre los modelos generadores y los generados. El siguiente gráfico muestra las actividades de esta etapa, en donde en cada una de ellas se define un conjunto de heurísticas que guían la construcción del *Modelo del Negocio*. La estrategia completa de esta etapa se describe en [Leonardi'03]. Actualmente se han desarrollado prototipos que permiten la semi-automatización de ciertos aspectos de esta etapa [Petersen'01; Antonelli'99], pero no en el contexto de RUP/UML.



3. Ingeniería directa e inversa de modelos estáticos UML

La falta de una semántica formal para UML pone límites a la potencia de las herramientas CASE UML. Éstas proveen poco soporte para la validación de modelos UML especificados en OCL y además no permiten transformar toda la información registrada en los modelos UML. Por ejemplo, una clase especificada en OCL brinda información importante para la identificación de clases y "frameworks" que puedan ser adaptados y reutilizados. A partir de diagramas de clase con especificaciones en OCL, también podrían generarse precondiciones y postcondiciones de operaciones e invariantes de una clase en lenguajes orientados a objetos que las soporten, como por ejemplo EIFFEL.

Una alternativa para evitar estas limitaciones es integrar a UML con técnicas formales. Se propone una integración con especificaciones algebraicas. Las bases de este enfoque son la definición de un lenguaje de modelamiento algebraico, la definición de un modelo de componentes reusables y de procesos transformacionales que permiten transformar modelos UML/OCL a especificaciones algebraicas y a código orientado a objetos.

Con respecto a la formalización algebraica se optó por diseñar un lenguaje de modelamiento, denominado GSBL^{oo} [Favre'01] cuyas primitivas reflejaran los conceptos de UML en forma directa. La característica esencial y original del mismo es que su diseño está centrado en abstracciones de relaciones (*relation-centric*). El lenguaje provee una jerarquía de tipos constructores para relaciones de dependencia, agregación, composición, y asociaciones. El lenguaje puede verse como un "intermediario" entre UML y otros lenguajes algebraicos, en particular se trabaja con los lenguajes CASL [COFI'00] y RSL [George'95].

Con respecto a la especificación formal de componentes reusables se definió un modelo que integra, en diferentes niveles de abstracción, especificaciones formales y código orientado a objetos. Un componente reusable consiste de tres diferentes niveles de abstracción (especialización, realización e implementación) que conectan especificaciones algebraicas incompletas, especificaciones algebraicas completas y código orientado a objetos respectivamente. El primer nivel consta de dos vistas, una algebraica y otra en OCL. Se definieron formalmente operadores de reuso (*Rename, Combine, Extend* y *Hide).* A partir del modelo se definió un método para el reuso de diseños y código basado en la adaptación e integración de componentes en los tres niveles. En particular para el método RAISE [George'95] para el cual no se hace referencia explícita a la reusabilidad de especificaciones, se ha definido un modelo [Felice'03] para describir la estructura de una componente reusable en los distintos niveles de abstracción. Este modelo permitirá la

integración de especificaciones de componentes en el lenguaje RSL y clases concretas en un lenguaje orientado a objetos.

El proceso de generación de código se basa en establecer correspondencias formales entre diferentes lenguajes. Las transiciones entre los diagramas UML y todas las especificaciones intermedias se realizan aplicando operadores de transformación que preservan la integridad entre especificaciones y código. Toda la información contenida en los modelos (por ejemplo asociaciones, multiplicidades y "constraints" OCL) tiene su contrapartida en las especificaciones algebraicas y tendrá implicancias en el código generado.

El proceso traduce en una primera etapa, modelos estáticos *UML* a una especificación algebraica. Un aspecto importante a destacar en esta etapa es la transformación de precondiciones, postcondiciones e invariantes de clases OCL a axiomas en la especificación algebraica. Se obtiene así una especificación a partir de la cual es posible realizar un análisis riguroso del comportamiento modelado con técnicas de razonamiento basadas en especificaciones algebraicas. La especificación obtenida será la base para la generación de implementaciones que se construyen a partir de esquemas de código reusables y transformaciones de especificaciones a código orientado a objetos.

Se prevé abordar *refactoring* de modelos UML y la ingeniería inversa de sistemas *legacy* a partir de las bases propuestas.

4. Conclusiones

Este proyecto pretende integrar las experiencias obtenidas por dos líneas de investigación que ya han tenido resultados en forma independiente: por un lado trabajos desde el área de ingeniería de requisitos [Leonardi´03, Leonardi´01, Leonardi´98, Leite´98] y por el otro de especificaciones formales [Favre´02, Favre´01, Mauco´02]. De esta forma, la estrategia integraría modelos orientados al cliente que ya han sido probados en el área de ingeniería de requisitos en el contexto de un desarrollo riguroso de software orientado a objetos. Resultados parciales de esta integración se encuentran en [Felice´02]. Para esto es necesario no sólo lograr un mayor grado de formalización en las heurísticas sino también refinar las heurísticas para obtener especificaciones más precisas de los modelos (por ejemplo, profundizar las heurísticas de tratamiento de las reglas del negocio para generar *constraints* en OCL). Estos modelos son la entrada para el proceso de ingeniería *forward*.

Asimismo, también se trabaja en la prototipación de etapas claves vinculadas a las extensiones propuestas. Los resultados logrados muestran que sería factible extender la funcionalidad de las herramientas CASE existentes y aumentar el grado de automatización de los procesos que soportan. Las actividades posteriores incluirán la integración de estos resultados con CASE UML.

Referencias

[Antonelli'99] Antonelli L., Rossi G., Oliveros A. Baseline Mentor, An Application that Derives CRC Cards from Lexicon and Scenarios. Anais de WER'99: Workshop en Requerimientos. 28 JAIOO, SADIO, pp.5-16.

[Booch'99] Booch G., Rumbaugh J., Jacobson, I. (1999). *The Unified Modeling Language User Guide*. Addison-Wesley.

[COFI'99] COFI Task Group on Language Design (1999). CASL: The Common Algebraic Specification Language. Available in: http://www.bricks.dk/Projects/CoFi/Documents/CASL.

[Favre'02] Favre, L., Martínez, L., Pereira, C. (2002). Forward Engineering and UML: From UML Static Models to Eiffel Code, Proceedings of IRMA 2002, 2002 Information Resources Management Association, Seattle, USA.

[Favre'01] Favre, L. (2001). A Formal Mapping between UML Static Models and Algebraic Specifications. Practical UML-Based Rigorous Development Methods-Countering or Integrating the eXtremist (eds. A. Evans, R. France, A. Moreira, B. Rumpe), Lecture Notes in Informatics (P 7) SEW, pp. 113-127, GI Edition Konner Kollen-Verlag, Alemania

[Favre'01-a] Favre, L. Clérici, S. (2001). A Systematic Approach to Transform UML Static Models to Object-Oriented Code. *Unified Modeling Language: System Analysis, Design and Development Issues* (eds. K. Siau, T. Halpin), Chapter 2, Idea-Group Publishing, USA.

[Felice´02] Felice L., Leonardi C., Favre L., Mauco V. Enhancing a rigorous reuse process with natural language requirement specifications. Successful Software Reengineering (ed. Sal Valenti). IRM Press. pp 129-142, 2002.

[Felice'03] Felice, L., Riesco, D. A rigorous model for raise specifications reusability. Capítulo 6 Practicing Software Engineering (eds. Joan Peckham, Scott J. Lloyd). pp 63-81. Idea Group Publishing, 2003.

[George'92] George C.,Haff, P.,Havelund, K. Haxthaunsen, A., Milne, R., Nielsen, C., Prehn, K. Ritter, K. *The RAISE Specification Language.* Prentice Hall. 1992.

[George'95] George, C., Haxthausen, A.,Hughes, S. Milne, R., Prehn, S., Pedersen, J. *The RAISE Development Method* Prentice Hall. 1995.

[Jackson´95] Jackson, M. "Software Requirements & Specifications" Addison-Wesley-1995.

[Jacobson´99] Jacobson I Booch G, Rumbaugh J "The Unified Software Development Process" Addison Wesley 1999.

[Kruchten'00] Kruchten, P. (2000). *The Rational Unified Process: An Introduction*. Addison-Wesley.

[Leonardi´03] Leonardi C. Enhancing RUP Bussiness Model with Client-Oriented Requeriments Models. UML and the Unified Process (ed. Liliana Favre). Idea Group Publishing, 2003.

[Leonardi´01] Leonardi C.. Una Estrategia de Modelado Conceptual de Objetos basada en Modelos de Requisitos en Lenguaje Natural. Tesis de Magíster en Ingeniería de Software. Facultad de Informática- Universidad Nacional de La Plata. Noviembre de 2001. Director Prof. Julio Cesar Leite, PhD. Pontificia Universidad Católica Río de Janeiro. Co-Director Dr. Gustavo Rossi, Fac. Informática, UNLP.

[Leonardi´98] Leonardi, C. Leite, J., Rossi, G. Estrategias para la identificación de Reglas de Negocio. Proceeding de Sbes98 "Simposio Brasilero de Engenharia de Software" Sociedad Brasilera de Computacao, Maringa, Brasil, 14-16 de Octubre de 1998. pág. 53-67.

[Leite´98] Leite, J., Leonardi, C. Business Rules as Organizational Policies IEEE IWSSD9: Ninth International Workshop on Software Specification and Design, IEEE Computer Society Press,1998,pp.68-76. ISBN:0-8186-8439-9.

[Leite´97] Leite, J., Rossi G., Maiorana V., Balaguer F., Kaplan G., Hadad G., Oliveros A. Enhancing a Requirements Baseline with Scenarios. Requirements Engineering Journal. Vol. 2 Nº 4. Springer-Verlag 1997. pp 184-198.

[OMG'01] Unified Modeling Language Specification, v1.4, Object Management Group, 2001 http://cgi.omg.org /cgi-bin/doc?ad/01-02-14.

[Mauco´02] Mauco V., Riesco D., George C. A Layered Architecture for a Formal Specification in RSL. Proceedings of the ACIS International Conference on Computer Science, Software Engineering, Information Technology, e-Business, and Applications (CSITeA-02). June 2002.

[Petersen'00] Petersen L., Tornabene S., Leonardi C, Doorn J. HeaR: Una Herramienta de Adquisición de Requisitos. Anais III Workshop en Engenharia de Requisitos, Rio de Janeiro, Julio del 2000, pp. 38-53.