Proceso Ágil para Desarrollo de Software

Autores

Arturo Carlos SERVETTO[(1)] *(aserve@mara.fi.uba.ar)*
Ramón GARCÍA MARTÍNEZ[(1 y 2)] *(rgm@mara.fi.uba.ar)*
Gregorio PERICHINSKY[(1)] *(gperi@mara.fi.uba.ar)*

[(1)] *Tel. (011) 4342-9184 / 4343-0891, Int. 142*
Laboratorio de Bases de Datos y Sistemas Operativos
Departamento de Computación
Facultad de Ingeniería de la Universidad de Buenos Aires

[(2)] *Instituto Tecnológico de Buenos Aires*

Introducción

La realidad de la industria del software de gestión, particularmente en nuestro país, impone la adopción de procesos ágiles de desarrollo para lograr competitividad. Reflejo de ello, a nivel internacional, es la creciente consolidación de la filosofía *Agile*, representada por los principios del *Agile Manifiesto[8]*. El objetivo principal de un proceso ágil es minimizar la documentación de desarrollo, incluso al extremo de considerarla descartable, y solamente como vehículo de comprensión de problemas dentro del grupo de trabajo y de comunicación con los usuarios.

Otra característica de los procesos ágiles es que, atendiendo la importancia de la participación de los usuarios finales, suelen ser iterativos e incrementales. Un ejemplo de proceso ágil pero que confiere cierta importancia a la documentación de desarrollo es el denominado Iconix[14], formalizado por Rosemberg y Scott, que es iterativo e incremental y se basa en el UML (Unified Modeling Language)[5].

Los cultores del desarrollo ágil (**http://www.agilealliance.org/**) suelen ser reacios a utilizar herramientas CASE, dada la rigidez, complejidad o parcialidad que suelen tener, cuando no por lo oneroso de adoptarlas[1]. En general son pocas las herramientas capaces de cubrir o soportar ciclos completos de desarrollo, y su utilización requiere tiempo y recursos que los desarrolladores ágiles no están dispuestos a resignar. Pero si se concibiese una herramienta que proponga un proceso ágil de desarrollo bien definido y que sea fácil de usar, sería de gran ayuda tanto para los adherentes a *Agile Alliance*, como para los desarrolladores en general.

Proyecto

El proyecto que motiva esta presentación tiene como objetivo el desarrollo de una herramienta ICASE (Integrate Computer Aided Software Engineering) que produzca sistemas, esto es, el código y la documentación completos, a partir de especificaciones gráficas de alto nivel de abstracción. Para ello se definió un proceso de desarrollo ágil basado en la *prototipación evolutiva*, con ciclos conformados por la especificación o evolución de requerimientos, la especificación o evolución de diseño y la generación o regeneración del sistema[11].

Para la *especificación de requerimientos* se emplea un modelo gráfico que sintetiza funcionalidad y comportamiento en base a la teoría de autómatas finitos[10]. Se concibe a todo sistema como a un autómata cuyos estados se asimilan a interfaces, y sus transiciones a funciones. Cada interfaz es entonces un escenario de interacción con el sistema, y los ciclos de transición entre ellas casos de uso[5].

La *especificación de diseño* se efectúa a partir del diagrama de requerimientos, asociando:

- A cada escenario, el diseño detallado de una ventana como especificación de formulario o reporte
- A cada caso de uso, el diseño detallado de un diagrama conceptual de entidades y relaciones comprometidas y un guión en álgebra relacional de entidades conceptuales

La *generación o regeneración del sistema* es automática, y consiste en crear o evolucionar los esquemas de datos a partir de diagramas nuevos o cambios en diagramas preexistentes, y en recrear el código a partir de los guiones.

Lo original del proyecto es la definición del proceso ágil o de ingeniería liviana, la aplicación de la teoría de autómatas finitos para sintetizar especificación de requerimientos y modelar comportamiento de sistemas, y la definición del álgebra relacional de entidades conceptuales y su extensión para la especificación de guiones funcionales.

Se completó el diseño de la herramienta y ya se ha comenzado su desarrollo. Los metadatos asociados a los diagramas de especificación se modelaron separando la información de definición de la información de representación, y se prevé como futuro proyecto agregar a la herramienta el manejo de repositorios con control de versiones.

Para la modelación conceptual de entidades y relaciones se adapta una extensión del modelo propuesta por los autores Batini, Ceri y Navathe [3][15] de gran riqueza expresiva, y que admite la definición de atributos compuestos, atributos polivalentes, cardinalidades mínima y máxima de entidades en relaciones y jerarquías de generalización con cobertura. En esta extensión se impone la definición de al menos un identificador con sentido semántico para cada entidad, y se admiten identificadores externos y mixtos, y la herencia múltiple de abstracciones en jerarquías de generalización/especialización.

Si bien en la herramienta se induce a la definición de un diagrama de entidades y relaciones por caso de uso, se prevé el reuso, coordinación e integración panóptica de diagramas, así como la denotación contextual de entidades, relaciones y atributos.

Para la definición de esquemas de datos a partir de diagramas conceptuales se define una heurística de transformación de modelos conceptuales a lógicos (relacionales) basada en reglas sobre el diagrama conceptual.

También se definen métricas de calidad, tanto para las estructuras de datos como para las funciones, para validar y orientar el diseño, y métricas de complejidad para la valorización de los sistemas que se produzcan.

Especificación de Requerimientos

Para esta fase del desarrollo de sistemas se propone modelar los escenarios operativos o de interacción y los casos de uso de un sistema en dos niveles, con un método basado en la teoría de autómatas finitos.

El modelo de primer nivel, más abstracto, representaría las unidades funcionales de la empresa cuyo sistema de información ha de automatizarse. A cada escenario, representado por un círculo o burbuja, se le asigna el nombre de la unidad funcional correspondiente y, en forma implícita (no visible en el diagrama sino por solicitud), el usuario o actor que tiene permiso para acceder. Como este nivel modela la estructura funcional de una empresa o institución, los diagramas tienen forma jerárquica.

En el Proceso Unificado de Desarrollo con UML[5][12], el primer nivel podría asociarse a paquetes de agrupamiento de casos de uso. Los paquetes de niveles terminales de la jerarquía corresponderán a las funciones operativas de la empresa o institución, y los que estén más hacia la raíz de la jerarquía corresponderán a consultas u operaciones para la toma de decisiones.

Los modelos del segundo nivel, accesibles a partir de cada escenario de primer nivel, representan las funciones automáticas de la unidad respectiva. Otra vez, cada burbuja representa una interfaz, pero ya no exclusivamente de tipo menú como las previas sino que las puede haber de tipo formulario o reporte, no de diálogo, aunque seguramente incluyan opciones funcionales. Aquí los enlaces o transiciones entre interfaz e interfaz, que en la terminología del UML serían *boundary classes*, representan operaciones o métodos que generalmente dan contenido a la interfaz destino.

La interfaz inicial de cada modelo de segundo nivel es estándar y se designa como el escenario de primer nivel o unidad funcional de la empresa; las funciones que se originan en ella determinan el inicio de transacciones o casos de uso con el sistema, y los ciclos que determinen en el diagrama los distintos escenarios (en el sentido que se da a este término en el modelo de casos de uso) o derivaciones alternativas.

Los diagramas de segundo nivel sintetizan funcionalidad y comportamiento y, conceptualmente, equivalen a la síntesis de modelos de casos de uso, de interacción y de actividad del UML.

En la interfaz inicial de los modelos de segundo nivel se puede asociar un texto de documentación con la descripción de los escenarios o guiones de las operaciones que se pueden activar desde ella. El texto sería visible al seleccionar la burbuja correspondiente a dicha interfaz. Cada ciclo se implementa como una transacción en la base de datos del sistema.

Los metadatos para la especificación de cada diagrama de requerimientos se dividen en dos archivos XML: uno para la definición de interfaces y transiciones, y otro para registrar sus posiciones relativas en el diagrama correspondiente.

Especificación de Diseño

La especificación de diseño consiste en tres actividades que a efectos prácticos pueden distribuirse en dos etapas o fases bien delimitadas. En la primera etapa se modelan los datos y las interfaces del sistema correspondientes a la parte que se desee desarrollar, a partir de lo cual la herramienta estará en condiciones de generar el primer prototipo, no funcional. En la segunda etapa se especifican las transiciones entre interfaz e interfaz asociando definiciones automáticas o guiones en álgebra relacional de entidades a los arcos de los modelos de segundo nivel que se quieran implementar.

Para poder diseñar interfaces con atributos de entidades del sistema, en general en modelos de requerimientos de segundo nivel, es necesario primero definir las entidades o escogerlas entre las ya definidas y asociarlas al arco de transición que incida en la interfaz. Así, desde cada arco se podrá visualizar un subdiagrama o subesquema de entidades y relaciones del sistema. Desde las interfaces del modelo de requerimientos de primer nivel se puede acceder a los subesquemas correspondientes a las unidades funcionales que representan, o a todo el sistema, si fuera la interfaz raíz del primer nivel o menú principal del sistema.

Se adapta una versión del modelo de entidades y relaciones que permite especificar cardinalidades mínimas y máximas no sólo a las entidades en una relación sino también a los atributos. Además, se admiten atributos compuestos (tanto optativos como obligatorios, y mono como polivalentes) y dominios definidos por extensión. Para no sobrecargar los diagramas, tanto los atributos compuestos como los dominios definidos por extensión (lista de valores) se pueden definir independientemente como tipos de atributos nuevos, y a la hora de definir un atributo en una entidad, se le puede poner como dominio el nombre de un tipo definido.

Los metadatos correspondientes al modelo de entidades y relaciones del sistema en la herramienta ICASE se dividen en tres archivos XML: uno para la definición de entidades y relaciones, otro para las ubicaciones en la representación general, y otro para las ubicaciones y participaciones en subdiagramas.

Conclusiones

Se cree que esta herramienta importa una contribución para la comunidad informática dedicada al desarrollo de sistemas de pequeña y mediana complejidad, dado que implica la adopción de una metodología simple y precisa que favorece la participación de los usuarios finales y mantiene a todo desarrollo permanentemente documentado.

La participación y el compromiso de los usuarios finales en desarrollos basados en esta herramienta se presumen garantizados debido a que los modelos empleados para las especificaciones son de un alto nivel de abstracción y comprensibles para personas no especializadas; además, la combinación del modelo de casos de uso, que permite verificar la completitud y rastrear el cumplimiento de requerimientos, con la posibilidad de la prototipación temprana, asequible con la generación de sistemas a partir de la especificación del diseño de interfaces, optimiza las relaciones contractuales facilitando la aprobación de fases.

Referencias

[1] Anfossi, D. y Servetto, A. “Relevamiento de Herramientas CASE Orientadas a Objetos: Comparación, Evaluación y Conclusiones”. Anales del III ICIE (Congreso Internacional de Ingeniería en Informática) de la Facultad de Ingeniería de la Universidad de Buenos Aires; pág. 449-463.

[2] Bass, L. , Clements, P. and Kazman, R. "Software Architecture in Practice". Addison - Wesley. 1998.

[3] Batini, C., Ceri, S., Navathe, S. “Diseño Conceptual de Bases de Datos: Un enfoque de Entidades-Interrelaciones”. Addison-Wesley Iberoamericana, 1991; ISBN 0-201-60120-6.

[4] Bell, D. and Morrey, I. "Software Engineering, Second Edition".Prentice Hall.1992.

[5] Booch, Grady; Rumbaugh, James; Jacobson, Ivar. “The Unified Modeling Language - User Guide”. Addison-Wesley, 1999.

[6] Brown, A. W. & McDermid, J. A., 1991, On Integration and Reuse in a Software Development Environment, Software Engineering Environments ‘91, F. Long & M. Tedd (Editors), Ellis Horwood, Mar 7.

[7] Constantine, L. and Lockwood, L. "Software for use". A Practical Guide to the Models and Methods of Usage - Centred Design. Addison - Wesley. 1999.

[8] Fowler, M. And Highsmith, J. “The Agile Manifiesto”. Software Development Magazine. Agosto 2001.

[9] Ghezzi, C. , Jazayeri, M. , Mandrioli, D. "Fundamentals of Software Engineering". Prentice Hall. 1991.

[10] Grimaldi, Ralph P. “Matemáticas discreta y combinatoria. Introducción y aplicaciones”. Addison-Wesley Iberoamericana, 1989.

[11] IEEE STD 1074-1991 Customer Request IEEE Standard fo developing Software Life Cycles Process, Identify Ideas or Needs, Preliminary Statement of Need, Feasibility Studies, Statement of need.

[12] Jacobson, Ivar; Booch, Grady; Rumbaugh, James. “The Unified Software Development Process”. Addison-Wesley ISE. 1999.

[13] Pfleeger, S. "Software Engineering: Theory and Practice". Prentice Hall. 1998.

[14] Rosenberg, D. And Scott, K. “Use Case Driven Object Modeling with UML”. Addison-Wesley, 1999.

[15] Servetto, A. “Mecanismos de Abstracción en la Modelación Conceptual de Datos y su Aplicación en una Ampliación del Modelo de Entidades y Relaciones” 3ras Jornadas de Informática e Investigación Operativa de la Universidad de la República de Uruguay (12 al 14 de diciembre de 1996).

[16] Sommerville, I. "Software Engineering". Addison - Wesley. 1996.

[17] Thomas, I. & Nejmeh, B. A., 1992, Definitions of Tool Integration for Environments, IEEE Software, 9, 2 (Mar), 29-35.

[18] Wasserman, A., 1990, Tool Integration in Software Engineering Environments, Software Engineering Environments: Proceedings of the international Workshop on Environments, F. Long (Editor), Springer-Verlag , 137-149.

[19] Wasserman. A. I., 1988, “Integration and Standarization Drive CASE Advancements, Computer Design”, 27, 22 (Dec), 86.