Proceso y Métodos de Ingeniería Liviana de Sistemas

Autores

Arturo Carlos SERVETTO, Ramón GARCÍA MARTÍNEZ, Gregorio PERICHINSKY

aserve/rgm/gperi@mara.fi.uba.ar

Tel. (011) 4342-9184 / 4343-0891, Int. 142

Laboratorio de Bases de Datos y Sistemas Operativos

Facultad de Ingeniería

Universidad de Buenos Aires

Introducción

Las metodologías de desarrollo de software surgieron a raíz de la necesidad de controlar y documentar proyectos cada vez más complejos, impulsadas principalmente por instituciones económicamente importantes y con requisitos de seguridad y fiabilidad en sus sistemas sumamente estrictos, como el Departamento de Defensa de los Estados Unidos.

Pero la evolución de estas metodologías, que sugieren o imponen a menudo varias actividades paralelas para cada fase del ciclo de vida de los sistemas y que asimismo pueden requerir más de un modelo para cada actividad, determinó que el costo y el esfuerzo requeridos en producir y mantener los documentos relativos al proceso de desarrollo crecieran considerablemente, a tal punto que hoy en día sólo instituciones muy grandes o que desarrollan sistemas muy complejos las adoptan y cumplen formalmente [1].

Las metodologías de desarrollo tradicionales modelan independientemente tres aspectos o dimensiones de los sistemas informáticos: estructura de la información, funciones y comportamiento. Cada aspecto requiere distintos modelos y los documentos resultantes pueden ser múltiples en los últimos dos aspectos, y en general son bastante costosos de mantener durante la evolución de los sistemas [2, 4, 7, 8, 11, 13].

En el desarrollo de sistemas pequeños o de mediana complejidad, la adopción de una metodología formal se percibe como un obstáculo más que como una ventaja, puesto que las ventajas sólo se hacen notorias a largo plazo, y los objetivos primarios de cualquier institución son producir software en el menor tiempo posible y a bajo costo para resultar competitiva.

Paralelamente, el empleo de herramientas de automatización del desarrollo de sistemas aún resulta bastante limitado debido a la complejidad, diversificación y falta de estandarización de las metodologías, que impide que una única herramienta pueda cubrir procesos completos de desarrollo o soportar al mismo tiempo metodologías distintas; así es que, para un desarrollo asistido en toda su extensión se debe contar con varias herramientas que a menudo no son compatibles entre sí. Además, sólo para empresas importantes o de desarrollo de sistemas de alta complejidad resulta económicamente viable emplear este tipo de herramientas para cubrir íntegramente el proceso de desarrollo [1].

Actualmente el mayor esfuerzo de la industria de sistemas informáticos está dirigido a la búsqueda de una metodología estándar así como a la estandarización de notaciones y representaciones de la información a los efectos de lograr la integración de herramientas de desarrollo [14, 15, 16].

La consolidación de nuevas tecnologías como las basadas en objetos, con su rico bagaje de modelos de interfaces y sus facilidades para la reusabilidad, hace propicio el desarrollo de herramientas de automatización integral del desarrollo de sistemas mediante un proceso de ingeniería liviana; es

decir, minimizando la cantidad de modelos e integrándolos en una única herramienta en la que se puedan mantener sin mayores costos [6].

Proyecto

El proyecto que motiva esta presentación tiene como objetivo el desarrollo de una herramienta CASE de automatización integral 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 ingeniería liviana 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 [10].

Para la *especificación de requerimientos* se emplea un modelo gráfico que sintetiza funcionalidad y comportamiento en base a la teoría de autómatas finitos [9]. 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 operativo o de interacción con el sistema, y las opciones a partir de ella casos de uso del escenario [5]. Se puede asociar uno o más grupos de usuarios como actores tanto a escenarios como a casos de uso, de manera que si un grupo se asocia a un escenario será actor en todos los casos de uso, y si se asocia a algún caso de uso en particular quedará excluido como actor en los restantes casos de uso; esto permite definir grupos de usuario jerárquicos, con acceso restringido a funciones específicas.

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 de ingeniería liviana, la aplicación de la teoría de autómatas finitos para sintetizar especificación de requerimientos y 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, 12] 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.

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 que pueden cambiar según el estado de la base de datos.

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.

Se cree que esta herramienta importa una contribución significativa 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] Ghezzi, C. , Jazayeri, M. , Mandrioli, D. "Fundamentals of Software Engineering". Prentice Hall. 1991.

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

[10] 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.

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

[12] 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).

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

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

[15] 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.

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