Diagramas de Estructuras como Mapa del Sistema

Walter F. Panessi **Mario G. Oloriz** **Claudia S. Ortiz**
wpanessi@unlu.edu.ar moloriz@unlu.edu.ar cortiz@unlu.edu.ar
Universidad Nacional de Luján
Departamento de Ciencias Básicas
Área Sistemas de Información

La línea de trabajo que venimos desarrollando, tiene como objetivo la generación de métodos y técnicas que puedan ser aplicadas por desarrolladores independientes, en proyectos de pequeña y mediana escala. Esta preocupación surge de nuestra experiencia en el desarrollo de software, de esta magnitud y a nivel nacional, habiendo verificado que la práctica de no contar con un proceso de desarrollo preestablecido y la falta casi total de documentación, es seguida por la mayoría de los desarrolladores de este tipo de proyectos.
Desde esta perspectiva nos encontramos desarrollando un modelo denominado Mapa del Sistema, el cual permitirá, con bajo costo, documentar el producto y facilitar las actividades de rastreabilidad, mantenimiento del producto y el reuso. Este modelo tiene la característica de interrelacionar los requerimientos con el diseño físico, con las componentes de software y el modelo de datos. Esta característica permitirá, por otra parte la generación automática de la documentación del software.
Se han definido los distintos elementos de que se dispondrá para la elaboración del Mapa del Sistema, los que permitirán alcanzar la funcionalidad propuesta.
El proyecto contempla el desarrollo de un producto de software que permita la gestión de proyectos basados en este modelo.

Introducción

Durante el año 2004, se realizó un estudio estadístico con el propósito de validar la hipótesis que sosteníamos desde nuestra experiencia como profesionales de sistemas [1]. La misma es que "en desarrollos pequeños y medianos (menos de U$S 20.000 de costo), la documentación del proceso de desarrollo es casi nula, siendo ésta una de las causas de fallas en el producto y del elevado costo de mantenimiento". El estudio reveló que, en esta escala de desarrollo, el grado de informalidad del proceso es de casi del 100%. Los procesos no están documentados, no se encuentran estandarizados y por lo tanto no se sigue una secuencia de pasos similar en cada proyecto, sino que el mismo es totalmente ad-hoc. En la mayoría de los casos, no se mantienen versiones separadas, sino que se actualiza sobre una única versión, la documentación mantenida, en el 80% de los casos estudiados, es solamente el código fuente del software, la estructura de las bases de datos y la memoria del desarrollador.

Otra investigación, respecto de las causas que llevan a los desarrolladores de software a proceder de esta manera, reveló que se piensa que la relación costo / beneficio de ocupar tiempo en la documentación del proyecto respecto de la usabilidad de la misma en etapas posteriores o en otros proyectos similares resulta desventajosa, desde la óptica del desarrollador.

El impacto más fuerte, de esta forma de trabajo, no es en el producto final, como cabía de esperar dadas las investigaciones realizadas en el área de requerimientos [2], sino en la evolución del software. Este hecho se explica fundamentalmente por el tamaño del proyecto y del equipo de desarrollo. Por tratarse de proyectos pequeños o medianos, el equipo de desarrollo es habitualmente unipersonal o está formado por unas pocas personas. El tiempo de avocación al proyecto, por otra parte, es prácticamente del 100%. Con esta dedicación y favorecidos por la alta comunicación que proporcionan los equipos de desarrollo pequeños y con tiempos cortos de desarrollo, los proyectos son habitualmente exitosos en cuanto a la funcionalidad esperada por el cliente y al cumplimiento de los requerimientos del software. Respecto de los costos y tiempos de entrega, no se puede decir lo mismo. En principio, los costos no se calculan, por lo tanto no se tienen en cuenta. Los tiempos comprometidos, por su parte, difícilmente se cumplen pues no fueron planificados ni administrados,

simplemente se estimaron por experiencia sin hacerse un estudio cuidadoso de las actividades del proyecto.

A la hora de la evolución del producto, cuando el tiempo entre el desarrollo y la adaptación del software es lo suficientemente prolongado como para que el equipo de desarrollo haya emprendido otros proyectos, los problemas en el software comienzan a aparecer. El estudio del impacto del cambio se realiza directamente sobre el código fuente, donde localizar todos los módulos del sistema que están relacionados directa o indirectamente con la alteración del producto es prácticamente imposible y depende totalmente de la memoria del equipo de desarrollo. Las pruebas son siempre insuficientes pues no se cuenta con un equipo de testing para realizarlas, son informales y habitualmente son hechas por los mismos desarrolladores. Con este contexto es fácil suponer que los errores se detectan durante la implementación o, en el peor de los casos, mucho más tarde por el usuario final. Se verificó que este procedimiento provoca que, reparar el daño causado por la falla es a veces muy difícil y en otros casos imposible.

Se han propuesto varias notaciones y lenguajes para describir al sistema software desde su arquitectura. Algunos modelos la describen a través de lenguaje estructurado [3] o con diagramas arquitecturales [4][5][6]. Estas descripciones representan la habitualmente llamada Arquitectura del software "idealizada" [7] o "Conceptual" [8]. Otros modelos permiten ver el diseño del software desde otras perspectivas, como la funcional (DFD, DFD Particionado por Eventos, etc.), la dinámica (Diagrama de estados, Redes de Petri, etc.). Pero a pesar de ello, los desarrolladores no los vislumbran como vinculados al código. No sienten que los beneficios a obtener por el uso de tales especificaciones sea significativo ante el costo que se invierte en su generación y administración. Sin embargo, esto no ocurre con el Diagrama Entidad Relación que es el modelo que hemos verificado como más usado. Una de las causas probables es que la distancia existente entre el código y los modelos mencionados en primer término es demasiado grande, mientras que no ocurre lo mismo con el modelo de datos dado su estrecha vinculación con el código.

Con esta motivación se pensó en un modelo que facilite la documentación del software y que a su vez haga de nexo entre el código y el diseño. Este modelo debe tener, además, algunas características claves :

1. Debe ser fácil de construir.
2. Facilitar la "Rastreabilidad" hacia adelante y hacia atrás.
3. Escalable, para que cada equipo de desarrollo utilice las especificaciones que piensen que son útiles para ellos.
4. Debe poder vincular partes heterogéneas de especificaciones (visión funcional con visión dinámica)
5. Permitir el reuso de especificaciones.
6. Ser fácil de interpretar y de manera unívoca.
7. Ser fácil de mantener

A este modelo se lo denominó **Mapa del Sistema** por proveer de una visión integral del mismo y permitir la navegación a través de toda la especificación existente.

Mapa del Sistema

El Mapa del Sistema se basa en un esquema estructural del software con un modelo de datos que permite mostrar a nivel de detalle las relaciones entre :

- los distintos módulos (como un diagrama de estructura);
- los módulos y las entidades del DER;
- los distintos sistemas, a través de los componentes que comparten, y

- las distintas versiones de un producto de software.

Como herramienta de diseño, puede ser usada para modelar la cohesión entre módulos. Como herramienta de traceability, marca las conexiones existentes entre todos los componentes del software, tanto desde los requerimientos como desde el diseño, ayudando a percibir los impactos del cambio en los dos extremos del proceso, desde los requerimientos hasta el modelo físico. También puede ser usada como ayuda para el reuso debido a su fuerte documentación a nivel de módulo, pero apunta **fundamentalmente** a ser un soporte para la **evolución** del producto.

Un Mapa del Sistema está integrado por los siguientes elementos :

Sistema

Define la raíz de la estructura y representa al objetivo del Sistema. De él dependen todos los módulos del producto.

Grupos

Los grupos son básicamente grupos de funcionalidades que comparten alguna característica y que el diseñador identificó como nexo. Un grupo está formado por todos los módulos que los diseñadores concentran en una misma opción del menú. Su función, dentro de la especificación es proveer de una granularidad intermedia entre el sistema y los módulos, permitiendo una lectura más cercana a la integración del software. Por otra parte, su conformación se encuentra directamente relacionada con la lógica del negocio y como los stakeholders visualizan o agrupan sus actividades cotidianas.

Módulos

Son las unidades funcionales del sistema. Están compuestas por un conjunto de procedimientos cuyo objetivo es proveer de una funcionalidad completa y distinguible al sistema (por ejemplo ABM de Clientes o Emisión de Factura de Venta).

Este módulo a su vez puede ser :

- Propio del sistema : En este caso se reconoce que el módulo es utilizado solo por éste sistema. No podría utilizarse en otro sistema pues en su concepción original no se ha garantizado la característica de reutilización.
- Compartido entre sistemas : Es un módulo que está incluido y vinculado a más de un sistema. En este caso, las alteraciones sobre él impactan en todos los sistemas en los cuales está incluido.

Submódulo

Se trata de las partes más pequeñas de los productos cuya función es proveer de cierta funcionalidad adicional a los módulos. Ejemplos de submódulos podrían ser las rutinas de búsqueda que se incluyen en los módulos, las rutinas de generación de informes, etc. Habitualmente estas partes se independizan de los módulos para poder reutilizarlas dentro del mismo sistema. Ej : Una rutina de búsqueda de clientes puede ser utilizada en el módulo de mantenimiento de clientes, en el módulo de facturación, en el módulo de pedidos, etc.

De la misma manera que los módulos, los submódulos pueden a su vez ser :

- Propios del sistema : En este caso se reconoce que el submódulo es utilizado solo en éste sistema, por lo tanto los cambios impactan solamente sobre los módulos de ese sistema.
- Compartido entre sistemas : Es un submódulo que está vinculado a más de un sistema. Sus alteraciones por lo tanto impactan sobre todos los módulos que lo usan en todos los sistemas donde se encuentre vinculado.

Vinculaciones

Los vínculos representan las relaciones entre el sistema, los grupos, los módulos y submódulos. Se marcan con líneas orientadas, indicando quien está requiriendo los servicios de quién. Adicionalmente, se puede incluir sobre la línea de comunicación entre dos elementos del mapa el o los parámetros que se envían desde un elemento hacia otro. Ese símbolo debe ser orientado. Su orientación indica desde donde hacia donde se envían los parámetros.

Proyecto de desarrollo actual

Como apoyo al Mapa del sistema se está diseñando un producto software que permita la gestión de desarrollos, basado en este modelo. La motivación es que una herramienta automática de éste tipo, puede darle valor agregado al modelo, y además permitiría validar la usabilidad del mismo. Este software proveerá de la funcionalidad necesaria para dar respuestas a consultas sobre las relaciones mantenidas en el Mapa del Sistema tales como :

- Sistemas que utilizan el Módulo “mtoCliente” en su versión 5.3.25
- Módulos del Sistema “ProdSys” versión 2005-02-25 que actualizan la tabla “Componentes”
- Módulos del Sistema “ProdSys” versión 2005-02-25 que implementan el Evento 32.
- Módulos del Sistema “ProdSys” versión 2005-02-25 relacionados con el módulo “facturarPedidos”

Además, el software deberá poseer las siguientes características :

1. Permitir la navegación hipervincular entre los diferentes componentes de la especificación.
2. Permitir la navegación hipervincular entre los diferentes módulos del sistema.
3. Mantener el control de versiones de los proyectos
4. Permitir el reuso de documentación.
5. Facilitar el uso de documentación compartida simplificando, de esta manera la gestión del cambio en los documentos.
6. Permitir la vinculación del Mapa con otros documentos del software.

Referencias

[1] W. Panessi, M. Oloriz, J. Peri; *Grado de Documentación del Proceso de Producción de Software*; Jornaciti 2004; Universidad Nacional de Luján; pp. 8-8; 2004

[2] O. Gotel, A. Finkelstein; *An Analysis of Requirements Traceability Problem*; Proc. of 1st Int. Conference on Requirements Engineering (ICRE'94), pp. 94-101, Colorado Springs, IEEE CS Press, April 1994

[3] R. Prieto-Diaz, J. Neighbors; *Module Interconnection Languages*; Journal of Systems and Software; Volumen 6; Pages 307-334; November 1986

[4] M. Shaw; *Procedure Calls Are the Assembly Language of Software Interconnection: Connectos Deserve First-Class Status*; Proceedings of Workshop on Studies of Software Design; Lecture Notes in Computer Science; Springer-Verlag; 1994.

[5] D. Garlan and M. Shaw; *An Introduction to Software Architecture*; In Advances in Software Engineering and Knowledge Engineering (Singapore, 1993); V. Ambriola and G. Tortora, Eds.; World Scientic Publishing Company; pp. 1-39. También aparece como SCS and SEI technical reports: CMU-CS-94-166, CMU/SEI-94-TR-21, ESC-TR-94-021.

[6] G. Abowd, R. Allen, D. Garlan; *Formalizing Style to Understand Descriptions of Software Architecture*; ACM Transactions on Software Engineering and Methodology 4(4) pp. 319-364, October 1995.

[7] D. Harris, H. Reubenstein, S. Yeh; *Reverse Engineering to the Architectural Level*; In Proceedings of the International Conference On Software Engineering; pp. 186-195; Seattle 1995.

[8] D. Soni, R. Nord, C. Hofmeister ; *Software Architecture in Industrial Application*; In Proceedings of the International Conference On Software Engineering; pp. 186-195; Seattle 1995.