Extracción y Visualización de Arquitecturas

Erika Michalczewsky Silvia M. Castro Pablo R. Fillottrani

{emichal,smc,prf}@cs.uns.edu.ar

Departamento Ciencias de la Computación
Universidad Nacional del Sur

1. Introducción

La arquitectura del software es un elemento que influencia fuertemente la capacidad del software para soportar atributos de calidad como modificabilidad, performance y seguridad. Es una herramienta que ayuda en el diseño de alto nivel de sistemas complejos, en el análisis temprano de diseños de alto nivel, particularmente lo relacionado con la satisfacción de atributos de calidad del software, en la reutilización de diseños a un alto nivel y en la mejora de la comunicación entre los participantes en el desarrollo y utilización del sistema [CAR97]. Sin embargo, la falta de documentación o la no actualización de la misma, reflejada en la poca relación entre la especificación y el sistema real, termina por ocultar sus beneficios.

La experiencia del Software Engineering Institute (SEI) en la evaluación de arquitecturas muestra que los nuevos desarrollos requieren de la utilización o compatibilidad con sistemas legacy [KAZ97]. En estos sistemas, las arquitecturas se encuentran rara vez documentadas, por lo que en un análisis de los mismos, generalmente se requiere la extracción total de la arquitectura.

Dali [KAZ99] es un sistema interactivo que asiste al usuario en la extracción, manipulación e interpretación de la información extraída de los fuentes, a través de una infraestructura que permite la integración de una amplia variedad de técnicas y herramientas. Dali no encuentra "la" arquitectura, por el contrario, sirve de soporte al usuario en la definición de patrones y la correspondencia de los mismos con la información extraída, utilizando tres técnicas en esta reconstrucción: la extracción de información, la fusión de la información utilizando patrones definidos por los usuarios y la visualización de los resultados.

Este trabajo ha sido realizado en el marco de la cátedra de posgrado Visualización, dictada en el primer cuatrimestre de 1999, en el Departamento de Ciencias de la Computación de la Universidad del Sur, y con el objetivo de cumplir el plan de estudios del magister, el cual incluye analizar la utilización de la visualización como herramienta en la reconstrucción de arquitecturas de software.

2. Dali

2.1. La extracción de información

Generalmente, la extracción de información relacionada con la arquitectura se realiza utilizando los fuentes, debido a que algunos sistemas no tienen sus arquitecturas documentadas, o sólo se documenta una visión de la misma, o la documentación del sistema no está actualizada con respecto a la implementación.

Como la representación más temprana de las decisiones de diseño, la arquitectura es analizada para asegurar el grado con el cual se soportan un conjunto deseado de atributos o escenarios. Sin embargo, este análisis resulta imposible si la arquitectura implementada no coincide con la diseñada, o si la documentación no incluye la información necesaria para soportarlo. Como el mantenimiento consume mucho tiempo y es propenso a la aparición de errores, es necesario disponer de alguna herramienta

soporte. Sin embargo, las herramientas por sí solas no entienden las abstracciones que utilizan las personas para estructurar y comunicar sus sistemas. Por esta razón, es importante la intervención e interpretación de las componentes extraídas sistemáticamente, ya que los elementos de las arquitecturas rara vez se encuentran directamente en las componentes que se desarrollan, o en los compiladores. Por ejemplo, no existen constructores de *capas* en un lenguaje de programación; no existe una manera sencilla de determinar si un elemento es un *cliente* o un *servidor* y los *subsistemas* se determinan por convenciones de nombres y llamadas. Sin embargo, se utilizan todas estas abstracciones para *definir* las arquitecturas de software [KAZ98].

La variedad de lenguajes, estilos arquitectónicos y convenciones de implementación que existen, determinan que una única colección de herramientas no es suficiente para la extracción y análisis de arquitecturas. Dali es un entorno que ofrece una infraestructura para facilitar la integración de una amplia variedad de herramientas de extracción, manipulación, análisis y presentación. Dali soporta un modelo de reconstrucción de arquitecturas *interpretativo* e *interactivo*. La reconstrucción es realizada por una persona familiar con el sistema, que manipula vistas de la arquitectura interactivamente. Este proceso provee un mecanismo para aplicar una *interpretación* a la arquitectura del sistema. La interpretación refleja como perciben y entienden el sistema aquellos involucrados. Esto es un punto clave: ningún sistema tiene *una* arquitectura, por el contrario tiene muchas vistas de sus muchas estructuras, cada una de las cuáles es apropiada para los diferentes análisis de actividades.

2.2. Vistas de los fuentes

Existen mecanismos—como por ejemplo, los módulos o la herencia que ofrecen un acceso restringido—que pueden *soportar* encapsulación de niveles o subsistemas. Pero generalmente, las abstracciones se encuentran en el diseñador, en las convenciones de uso o de nombre, y no pueden ser extraídas por una herramienta. Por lo tanto, es necesario disponer de múltiples técnicas de extracción que permitan capturar información de la topología del sistema, que incluya relaciones como la alocación de software a diferentes procesos o procesadores, que recuperen la información de estructura proveniente de los fuentes y que detecten vinculaciones tardías, como polimorfismo, punteros a funciones y parámetros, todos elementos que se determinan durante la ejecución.

Para soportar el análisis y evaluación de una arquitectura de software el primer paso necesario es la extracción de un *modelo concreto*, una representación del sistema implementado. Tal representación contiene una colección de *elementos* (por ejemplo, funciones, archivos, variables, objetos), una colección de *relaciones* entre los elementos (por ejemplo, "la función X llama a la función Y", "el archivo X contiene a la función Y") y un conjunto de atributos de estos elementos y relaciones (por ejemplo, "la función X llama a la función Y, *N* veces", "el objeto A es de tipo B"). Un modelo concreto puede reflejar varias *vistas* de un sistema: la estructura estática, la naturaleza dinámica, o la estructura utilizada durante el desarrollo.

Existen varias técnicas y herramientas para la extracción del modelo estático del fuente. Es importante apreciar que ninguna técnica por sí sola logrará extraer con éxito el modelo completo.. Algunas de las herramientas comerciales y de investigación que utiliza Dali son: rigiparse para el lenguaje C, Imagix [IMA] para C y C++, SNiff+ [TAK] para C++ y Fortran, LSME [MUR96] para C++.

1.3. Repositorio Híbrido

Una vez extraído el modelo concreto, el resultado—las *vistas extraídas*—puede ser almacenado en un repositorio, que es una base de datos SQL y que actúa como almacenamiento principal del modelo, junto con formatos de archivos específicos a una aplicación para su intercambio entre herramientas. De este modo las herramientas acceden directamente a la base de datos, utilizando interfaces programadas, o dependiendo de la disponibilidad de archivos de datos en los formatos apropiados. Este esquema permite a los usuarios disponer de las ventajas de tener un repositorio, y aliviar la tarea

del mantenimiento del modelo que sería necesaria para poder utilizar todas las herramientas disponibles. Una vez que la herramienta manipuló el modelo, el repositorio es actualizado.

1.4. Fusión de Vistas

Para crear un visión completa de una arquitectura se requiere que las vistas separadas sean fusionadas. Fusionar vistas de la arquitectura significa establecer conexiones entre ellas. Este punto es muy importante porque las diferentes vistas ofrecen información complementaria sobre el sistema y porque una vista puede ser mejorada con la información provista por otra vista. Asimismo, los usuarios deben poder navegar por las vistas para entender completamente la arquitectura.

Extraída la colección de elementos, las *vistas fusionadas* se definen sobre los elementos extraídos. Algunas fusiones se realizan para mejorar las vistas, otras para resolver las relaciones de "llamadas" ambiguas y otras para obtener la topología de tiempo de ejecución de la arquitectura.

1.5. Visualización e Interacción

La interacción con el usuario se logra con una componente que se encarga de manipular directamente el modelo y guiar el análisis y la manipulación automática. Se plantea a través de Rigi, que provee un balance entre la generalidad (vía un lenguaje programable de comandos, RCL---Rigi Command Language) y la aplicabilidad de dominios. Rigi también ofrece mecanismos para graficar, anotar y manipular directamente elementos de una vista. La principal tarea del usuario al interactuar y manipular estructuras de la arquitectura es organizar la información extraída en vistas que son más significativas para él, es decir más abstractas, y más representativas de su entendimiento acerca de la arquitectura del sistema. Esta tarea la realiza utilizando las facilidades mencionadas anteriormente (anotación, manipulación de gráficos, layout). Por ejemplo, un usuario podría querer agrupar todas los sistemas de llamadas de ventanas en un nivel de "interface de usuario". El resultado de este proceso es la creación de *vistas refinadas.*

1.6. Manipulación Externa

La integración de Dali con otras herramientas, se logra a través de tres pasos:

- exportar el modelo desde Rigi
- aplicar la herramienta apropiada para manipular o analizar el modelo
- (opcional) importar el resultado

Esta integración es posible utilizando una porción de código implementado en RCL, que sincroniza el modelo en Rigi con cualquier cambio al modelo realizado por la aplicación externa.

1.7. Análisis

Muy relacionadas a las técnicas de manipulación externa se encuentran aquellas para el análisis. Existen muchas herramientas para realizar análisis de arquitecturas, como testeo de estructura y medición de la complejidad basada en patrones. Para utilizar estas herramientas Dali aplica un modelo de importación/exportación: el modelo que existe en Rigi se exporta, se aplica la herramienta apropiada y los resultados se visualizan en Rigi o utilizando cualquier otra herramienta externa. Ejemplos de las herramientas utilizadas para el análisis de las arquitecturas son IAPR [KB98] para el análisis de la complejidad de la arquitectura y RMTool [MNS95] para la medición automática de conformance.

Referencias

[CAR97] S. Jeromy Carrière y Rick Kazman. *Assessing Design Quality From a Software Architectural Perspective*, OOPSLA'97 Workshop on Object-Oriented Design Quality, October 5th, 1997.

[IMA] Imagix Corporation, http://www.imagix.com.

[KAZ97] Rick Kazman y S. Jeromy Carrière. *Playing Detective: Reconstructing Software Architecture from Available Evidence.*Technical Report. CMU/SEI-97-TR-010, 1997.

[KB98] Rick Kazman y M. Burth. *Assessing Architectural Complexity*, en Proceedings of 2nd Euromicro Working Conference on Software Maintainance and Reengineering, Florencia, Italia, Marzo 1998, 104--112.

[KAZ98] Rick Kazman y S. Jeromy Carrière. *View Extraction and View Fusion in Architectural Understanding*, 5th International Conference on Software Reuse, Junio 2--5, 1998, Victoria, BC, Canada.

[KAZ99] Rick Kazman y S. Jeromy Carrière. *Playing Detective: Reconstructing Software Architecture from Available Evidence*, Journal of Automated Software Engineering, 6:2, Abril, 1999, 107--138.

[MNS95] G. Murphy, D. Notkin y K. Sullivan. *Software Reflexion Models: Bridging the Gap between Source and High-level Models*, en Proceedings of the Third ACM SIGSOFT Symposium on the Foundations of Software Engineering, Washington D.C., Octubre 1995.

[MUR96] G. Murphy y D. Notkin. *Lightweight Lexical Source Model Extraction*, ACM Transactions on Software Engineering and Methodology, 5(3), Julio 1996, 262--292.

[TAK] TakeFive software, http//www.takefive.com.

[UIM] UIMS Tool Developers Workshop. *A Metamodel for Runtime Architecture of an Interactive System*, SIGCHI Bulletin, 24(1), 32--37.