Atributos de Calidad y Arquitectura del Software

María Cecilia Bastarrica

Departamento de Ciencias de la Computación
Universidad de Chile
cecilia@dcc.uchile.cl

Abstract. La arquitectura del software ha pasado a ser un artefacto esencial dentro del proceso de desarrollo de software moderno debido a sus múltiples usos: documentación de decisiones tempranas de diseño, base para la comunicación con los distintos stakeholders y guía para la implementación y mantenimiento del sistema. Pero llegar a construir una arquitectura para el software que sea a la vez apropiada para dar cabida a las exigencias de las distintas partes interesadas y buena en términos absolutos es una tarea que dista muchísimo de ser sencilla. En este minicurso se expondrán las tendencias actuales para desarrollar un sistema centrándose en su arquitectura, y más específicamente en cómo construir una buena arquitectura para el sistema deseado.

1 Introducción

Como parte de este minicurso de 4 horas se expondrán los conceptos de arquitectura de software, atributos de calidad y modelamiento de la arquitectura para lograr estos requisitos de calidad. Se verá cómo es necesario especificar la arquitectura desde distintos puntos de vista para poder poner de manifiesto las distintas cualidades y tomar decisiones de diseño balanceadas, que armonicen el alcance de una cualidad sin sacrificar las demás en forma desmedida.

2 Arquitectura de Software

La arquitectura del software es una primera aproximación al diseño de alto nivel donde se identifican los principales componentes, su interacción y sus dependencias. Según [1], puede decirse que

> *"la arquitectura de un programa o sistema de cómputo es la estructura o estructuras del sistema, los cuales comprenden los elementos de software, las propiedades externamente visibles de estos elementos, y las relaciones entre ellos"*

Esta definición tiene múltiples implicancias. Primeramente, la arquitectura es una abstracción de un sistema. De hecho, todo sistema tiene una arquitectura, lo cual

no significa ni que ésta sea conocida, ni buena, ni apropiada para los propósitos del proyecto.

2.1. Atributos de Calidad

Si sólo la funcionalidad de un sistema de software fuese importante a la hora de hacer su diseño, cualquier sistema monolítico podría servir. Sin embargo la realidad es otra. Muchos otros atributos de calidad deben ser tenidos en cuenta para determinar la estructura y el comportamiento que tendría el sistema. Hacer un adecuado balance entre la mantenibilidad, la interoperabilidad, la portabilidad, la performance, la seguridad, la disponibilidad, y la reusabilidad, entre otros atributos, es esencial para obtener un sistema que cumpla las expectativas de las distintas partes interesadas.

Esto es algo más fácil de decir que de llevar a la práctica debido a que cualquier cambio en la arquitectura de modo de mejorar un atributo, en general estará afectando negativamente otro atributo.

Y como si esto no fuese poco, el diseño de una arquitectura que tenga en cuenta los atributos deseados, sólo permitirá que el sistema resultante tenga estas característica, pero no lo garantiza.

Podemos clasificar los atributos de calidad del software en dos grandes grupos: aquellos que pueden comprobarse durante la ejecución del software y aquellos que no pueden comprobarse durante la ejecución. Dentro del primer grupo se encuentra la performance, seguridad, disponibilidad, funcionalidad y usabilidad. Dentro de los segundos, y no menos importantes, están la modificabilidad, portabilidad, reusabilidad, integrabilidad y verificabilidad.

Dada la naturaleza diversa de los potenciales atributos de calidad deseados, se ha visto que es necesario contar con distintas "vistas" que pongan de manifiesto las distintas facetas de un sistema de software a la hora de especificar la arquitectura. Es así como la definición de la arquitectura de un software deberá estar constituida por más de una vista, que permitan analizar tanto sus cualidades dinámicas como las estáticas.

2.2. Atributos de Calidad

Según [3], documentar una arquitectura es documentar las vistas relevantes y luego agregar la información que se aplica a más de una vista. Una vista es la representación de un conjunto de elementos del sistema y las relaciones asociadas con los mismos. Las vistas usadas dependen de las necesidades y el uso que se daría la arquitectura, así como también los atributos de calidad requeridos. Existen tres tipos de vistas:

Módulos: estructura en términos de unidades de implementación. Permiten analizar las cualidades estáticas del software.

Componentes y conectores: estructura en términos de elementos interactuantes durante su ejecución. Permiten analizar las cualidades dinámicas del software.

Allocation: relación entre el sistema y las estructuras externas del ambiente. Permite analizar las cualidades de la arquitectura en términos absolutos.

Dentro de cada tipo de vistas existen distintos estilos de arquitectura. Los estilos, al igual que los patrones de arquitectura [2], son restricciones a los tipos de vistas que establecen el tipo de elementos que puede participar y su tipo de interacción. Distintos estilos de arquitectura determinan que el sistema resultante tenga distintos atributos de calidad. Un estilo tal como tubos y filtros es del tipo de componentes y conectores y aplicarlo a un diseño de arquitectura da como resultado un sistema con una alta modificabilidad, pero no necesariamente una alta performance. Asimismo, un sistema con un estilo de arquitectura de cliente-servidor puede contar con una muy alta seguridad y modificabilidad con mínimo impacto si los cambios ocurren en los clientes, pero con un altísimo impacto si estos cambios ocurren en el servidor.

En [4] se establece un método práctico para integrar las distintas vistas de una arquitectura en un único documento que la especifique de manera integral. A pesar de que este método es relativamente nuevo, existe ya evidencia de que ha sido útil en la práctica.

2.3. Documentación

El método consiste básicamente en los siguientes pasos:

1. Identificar los stakeholders del sistema
2. Describir los requerimientos de atributos de calidad
3. Producir una lista de vistas que describan la arquitectura del sistema
4. Crear una tabla jerarquizando la importancia de la información que brinda cada una de las vistas para cada stakeholder
5. Elegir las vistas más relevantes
6. Combinar vistas
7. Documentar las vistas escogidas
8. Documentar la información que se aplica a más de una vista

Durante el curso se expondrá y analizará una aplicación real cuya arquitectura ha sido construida con esta metodología.

3 Conclusiones

Lograr una arquitectura que satisfaga los requisitos de funcionalidad y calidad no es una tarea fácil. Las decisiones que se tomen en la etapa de diseño de la arquitectura tendrán un impacto decisivo sobre las cualidades del sistema resultante. Se expone en este curso una metodología basada en [4] para seguir este proceso y ser capaz de analizar la especificación de modo de comprobar que la arquitectura del software cuenta con las cualidades requeridas.

4 Referencias

[1] Len Bass, Paul Clements, and Rick Kazman. Software Architecture in Practice. SEI Series in Software Engineering. Addison-Wesley, 2 edition, 2003.

[2] Frank Buschmann, Regine Meunier, Hans Rohnert, and Peter Sommerlad. Pattern Oriented Software Architecture: A System of Patterns. John Wiley & Son Ltd., August 1996.

[3] Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Robert Nord, and Judith Stafford. Documenting Software Architectures. Views and Beyond. SEI Series in Software Engineering. Addison Wesley, 2002.

[4] Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Robert Nord, and Judith Stafford. A practical method for documenting software architectures, May 2003. Submitted to the 2003 International Conference on Software Engineering.