Plan de Trabajo Final

Carrera Ingeniería de Sistemas

Facultad de Ciencias Exactas – UNICEN

Tema: Asistencia inteligente para diseñadores de sistemas de software guiada por atributos de calidad.

Alumno/s: Nicolás Cirigliano

Juan Ignacio Pedroche

Director: Dr. Luis Sebastián Berdún

1. Introducción

El desarrollo de grandes y complejos sistemas de software es, en la actualidad, un tema muy importante y posee diversas posturas de cómo abordarlo. El diseño de la Arquitectura de un Software es el proceso por el cual se define una solución para los requisitos técnicos y operacionales del mismo. Este proceso define qué componentes forman el software, cómo se relacionan entre ellos, y cómo mediante su interacción llevan a cabo la funcionalidad especificada.

Según Bass, Clements y Kazman "Una arquitectura es la descripción de las estructuras de un sistema, puede haber varias: descomposición modular, de proceso, de despliegue, de capas, etc. La arquitectura es el primer artefacto que puede ser analizado para determinar cómo son logrados los atributos de calidad, y también sirve como el plan del proyecto. Una arquitectura sirve como el vehículo de comunicación, es la manifestación de las decisiones de diseño tempranas, y es una abstracción reusable que puede ser transferida a nuevos sistemas" [R6]. Partiendo de esta definición, se puede observar que la arquitectura de un sistema de software es una poderosa herramienta a tener en cuenta en cualquier proyecto ya que permite no solo proporcionar una guía para la producción del mismo, sino que también, es un medio que facilita la comunicación entre los distintos integrantes de un grupo de desarrollo y aporta información muy importante a la documentación del mismo.

2. Motivación

Los subsistemas de una arquitectura de software, así como también las relaciones entre éstos, usualmente consisten en varias unidades arquitectónicas menores. Éstas son descritas mediante patrones de diseño.

Un patrón de diseño provee un esquema para refinar subsistemas o componentes de un sistema de software, o las relaciones entre ellos. Describe la estructura comúnmente recurrente de los componentes en comunicación, que resuelve un problema general de diseño en un contexto particular [R8].

La calidad de un software se define como el grado en el cual éste posee una combinación deseada de atributos. Tales atributos son requerimientos adicionales del sistema, que hacen referencia a características que debe satisfacer, diferentes a los requerimientos funcionales. Estas características o atributos se conocen con el nombre de **atributos de calidad**, los cuales se definen como las propiedades de un servicio que presta el sistema a sus usuarios.

Los atributos de calidad deben ser considerados en todo el diseño, implementación y despliegue. Ningún atributo de calidad es totalmente dependiente del diseño, ni es totalmente dependiente de la aplicación o implementación. Un resultado satisfactorio es cuestión de obtener el panorama general (arquitectura), así como los detalles (implementación) correctos [R1][R6][R9][R11].

Los patrones ayudan al diseñador a definir la composición y el comportamiento del sistema de software, y una combinación adecuada de ellos permite alcanzar los atributos de calidad.

Como se desarrolló en la sección anterior, la utilización de patrones de diseño sirve para alcanzar los atributos de calidad deseados, pero a su vez éstos pueden tener un impacto negativo sobre otros atributos. Cada patrón de diseño favorece o perjudica a un conjunto de atributos de calidad, y sumada la gran cantidad de patrones existentes, hace de la tarea del diseño de un sistema una actividad que requiere de conocimiento avanzado.

3. Objetivos

Se propone construir una solución que facilite la aplicación de los patrones de diseño en diferentes situaciones como, por ejemplo, en proyectos de pequeña y mediana escala donde posiblemente no se tenga un diseño detallado del sistema, o con usuarios inexpertos con intenciones de aprender y utilizar los beneficios de los patrones de diseño.

Se plantea realizar una herramienta con integración a un entorno de desarrollo, y de esa forma brindar asistencia para la correcta implementación de los conceptos analizados.[1]

Tomando como entrada de la aplicación un diagrama de clases UML[R5] y los atributos de calidad que el usuario pretenda en el sistema a desarrollar, el asistente recomendará una lista de patrones de diseño que se ajusten a sus necesidades para lograr una mejora en el sistema. Dichos patrones de diseño serán seleccionados de una base de conocimiento experto y luego comparados contra el diseño del usuario para determinar la similitud de cada uno.

La herramienta estará dividida en dos componentes, planteando una arquitectura Cliente-Servidor que permitirá realizar los algoritmos de procesamiento en un lugar centralizado, brindando las mismas sugerencias para un grupo de desarrolladores, así como también la posibilidad de utilizar la herramienta de forma local y particular. En la figura 1 se observa un esquema general de la herramienta.

[1] Utilizando como base una herramienta existente.

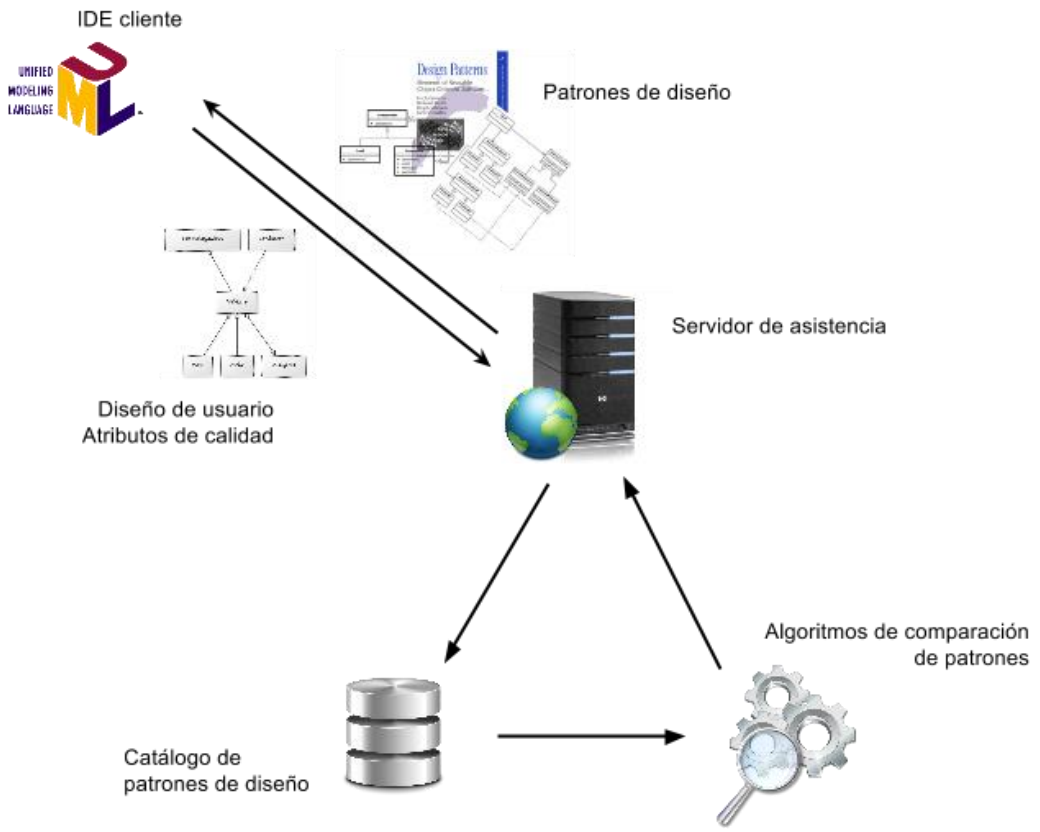

Figura 1 *Esquema general de la herramienta a desarrollar.*

Interfaz de usuario

Se pretende realizar una interfaz de usuario simple e intuitiva para facilitar el diseño de software. El usuario podrá pedir asistencia ingresando los atributos de calidad buscados en su programa y un diagrama de clases UML, ya sea de forma directa a través de su interfaz, o tomando código fuente de un proyecto Java.

La respuesta de la herramienta será una sugerencia de patrones de diseño (diagrama junto a explicación del uso del patrón sugerido) que el usuario podría aplicar a su diseño para mejorar el o los atributos necesarios en su proyecto.

Conexión con el servidor

La lógica del asistente se encontrará ubicada en una aplicación web contenida en un servidor de aplicaciones. La herramienta integrada a un

entorno de desarrollo se conectará a éste mediante un web service, le proveerá el diagrama y los atributos de calidad definidos por el usuario, y luego de realizar los procesos requeridos retornará la recomendación al usuario.

Procesamiento de la información

El asistente realizará comparaciones entre diagramas y, en base a diversos criterios de selección, se elegirá un conjunto solución con los patrones de diseño que más acerquen el diseño del usuario a los atributos de calidad requeridos. Cada patrón de diseño tendrá un valor de éxito asociado luego de que la comparación haya finalizado.

Si bien inicialmente la herramienta trabajará con los patrones de diseño, la dinámica de la misma permite que se incorporen los diseños personalizados que la organización desee, permitiendo reutilizar el conocimiento adquirido en proyectos anteriores.

Las comparaciones entre los diagramas se realizarán utilizando algoritmos computacionales diferentes, dejando la posibilidad de que en un futuro sea posible adicionar nuevas heurísticas para buscar similitudes entre diagramas.

Algoritmos a utilizar

A continuación se presentan los algoritmos analizados para llevar a cabo las comparaciones de diagramas.

Aplanamiento

La forma aplanada de un diagrama de diseño consiste en absorber todas las conexiones entre componentes de forma que se obtenga un diagrama en el cual la existencia de conectores sea nula. Al realizar esta absorción, los componentes incorporarán propiedades que identifiquen y describan toda la información del esquema original, obteniendo una forma compacta de los datos, y manipulando las relaciones entre los componentes del diagrama, de manera de simplificar su representación.

Comparación Topológica

Compara los diagramas en base a la disposición de los elementos en ellos utilizando teoría de grafos dirigidos. De esta forma se podrán buscar similitudes y diferencias entre diagramas estudiando su morfología.

4. Cronograma de actividades

Actividad	Duración
Recopilación bibliográfica.	4 semanas
Comprensión de sistemas de asistencia a usuarios y métricas de comparación de diseños de software.	4 semanas
Implementación de los algoritmos.	5 semanas
Implementación de la herramienta con la inserción de los algoritmos y el modelo de datos.	5 semanas
Evaluación de la herramienta	5 semanas
Generación de informe escrito (en paralelo con las Actividades anteriores).	23 semanas

5. Bibliografía

[R1] Ian Sommerville: Ingeniería de Software - Pearson, Addison Wesley (2005).

[R2] Grady Booch, Ivar Jacobson, Jim Rumbaugh: OMG Unified Modeling Language Specification, (2000).

[R3] Bass, L., Clements, P. And Kazman, R.: Software Architecture in Practice, Addison Wesley (2003).

[R4] F. Buschmann, R. Meunier, H. Rohnert, P.: Pattern-Oriented Software Architecture. A System of Patterns, (1996).

[R5] E. Gamma, R. Helm, R. Johnson, J. Vlissides.: Design Patterns, Elements of Reusable Object-Oriented Software, Addison Wesley (1994).

[R6] SoftwareArchitectures.com (2006). "Intro to Software Quality Attributes".