Propuesta de Trabajo Final de la Carrera Ingeniería de Sistemas

Un Asistente para el soporte del Desarrollador de Software en la Aplicación de Patrones de Diseño

Director: Dr. Luis Sebastián Berdun

Co-director: Dr. Marcelo Gabriel Armentano

Alumnos: Silvina Calderón & Ricardo Gastón Mancini

1. Introducción

Los Patrones de Diseño se han vuelto cada vez más populares e importantes para los Diseñadores de Software. Esto se debió, principalmente, a que los profesionales disponen de distintas soluciones para un mismo problema. Para su aprendizaje, diferentes autores han propuesto catálogos donde los dan a conocer junto a sus características. Si bien esto contribuye a su difusión dentro de la comunidad, genera una gran desventaja: incentiva a los lectores a no interesarse por la práctica.

Los diseñadores novatos expuestos por primera vez a un Patrón de Diseño, deberán primero aprender cómo aplicarlos a problemas reales y, de esta forma, podrán entender cómo un Patrón junto a sus características colaboran para dar o no solución a un determinado problema (Biggs, J., 2003). Un método para enseñarle a una persona los Patrones de Diseño es sentarlo frente a un problema con un tutor a su lado que vaya realizándole recomendaciones siempre que lo considere necesario. Sin embargo, la aplicación de este modelo de enseñanza es muy poco frecuente, debido a que se necesitará de un gran número de tutores y, a su vez, los problemas a solucionar tienden a ser largos y complejos para ser implementados en poco tiempo.

Teniendo en cuenta las dos problemáticas mencionadas anteriormente, la asistencia automatizada surge como una alternativa. Complementando a una herramienta CASE[1] con dicha asistencia, se podrá brindar soporte a los diseñadores novatos y así ayudarlos a entender, aprender y aplicar los Patrones de Diseño a un determinado problema.

Una forma interesante de lograr una asistencia automatizada es a partir del uso de un Agente de Interfaz (Maes, P., 1994). Estos, surgen con la finalidad de proveer a los usuarios asistencia proactiva y reactiva de manera personalizada en el uso de una aplicación de software. Básicamente, la idea principal consiste en tener una entidad de software especializada en el dominio, monitoreando las actividades del usuario, detectando oportunidades para ayudarlo en su trabajo, y luego ofreciendo asistencia para indicar qué actividades deben llevarse a cabo para

[1] CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Computadoras)

lograr un determinado objetivo. En el marco de este trabajo, la actividad que estará desarrollando el usuario será la especificación de un Diagrama de Clases[2] y el objetivo a cumplir uno de los Patrones de Diseño de Software presentados en Gamma, E. et al., 1994.

Un Agente de Interfaz podrá verse involucrado dentro del proceso de modelado de un Diagrama de Clases en dos etapas distintas: durante el desarrollo del proceso o finalizado el mismo. En el presente trabajo se utiliza una técnica que se corresponde a la primera etapa, dado que se tiene como objetivo la detección temprana de un Patrón de Diseño de Software.

Teniendo en cuenta la problemática detallada anteriormente, el objetivo de este trabajo es el desarrollo e implementación de un agente inteligente capaz de asistir al desarrollador de software en la aplicación de Patrones de Diseño. El enfoque que se propone radica en aplicar un algoritmo teórico de modelos de Markov de Orden Variable dentro de una herramienta CASE; de esta manera se intentará detectar la intención del usuario para así sugerirle el Patrón de Diseño más adecuado. Al momento de realizar la sugerencia se prevé no sólo utilizar el patrón cuya probabilidad sea más alta, sino enriquecer al agente con el conocimiento teórico.

En el siguiente gráfico se podrá observar de qué manera trabajará el asistente para poder brindarle ayuda al usuario.

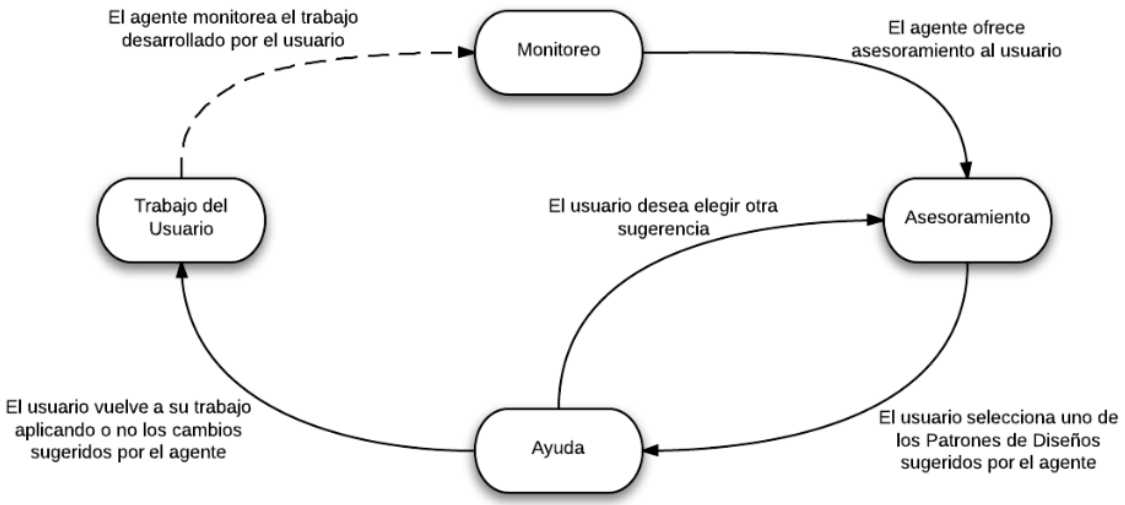

Ilustración 1: Fases del asistente

Como puede observarse en la Ilustración 1, el agente monitorea las acciones que va realizando el usuario hasta que puede ofrecer alguna sugerencia, la cual puede ser aceptada o no.

[2] Se utilizan para describir la estructura de un sistema, mostrando sus clases, atributos y relaciones.

2. Trabajos Relacionados

Para lograr asistir a un usuario, un Agente de Interfaz deberá primero detectar cuál es el objetivo que éste está buscando. Con dicho fin se utiliza una técnica de Plan Recognition (Kautz, H, 1991), que permite detectar cuál será el objetivo del usuario en función de las acciones realizadas por el mismo en un determinado dominio. En nuestro caso, los objetivos son los Patrones de Diseño, y las acciones son las operaciones que permiten definir un Diagrama de Clases.

Para el modelado de los objetivos perseguidos por un usuario, las técnicas de Plan Recognition hacen uso de bibliotecas de planes que permiten modelar las acciones del usuario para lograr un objetivo. De la correctitud y completitud de dichas bibliotecas dependerá el rendimiento de la técnica de Plan Recognition, y por lo tanto del Agente de Interfaz que haga uso de ésta.

En Berdun, et al. 2008, se presenta la herramienta Pattern Advisor. La misma, propone la utilización de Redes de Bayes para analizar las acciones llevadas a cabo por el usuario y luego así proveer al agente de interfaz de la información necesaria para asistirlo. Dichas redes son previamente entrenadas con el conocimiento de expertos, pre-requisito que no será necesario para el trabajo planteado en esta propuesta, donde la información necesaria para entrenar a nuestro algoritmo de Plan Recognition será un Plan Corpus generado de forma puramente computacional.

En Silva Logroño, J. et. Al. 2010, se presenta un enfoque teórico que aplica Modelos de Markov de Orden Variable -como técnica de Plan Recognition- para la detección de la intención del usuario en la aplicación de un determinado patrón.

Cabe destacar que en los trabajos citados anteriormente, la mayoría de los autores no han ido más allá de la detección del patrón o del estudio de la eficacia de diferentes métodos de detección.

3. Propuesta

En este trabajo se propone utilizar el algoritmo desarrollado de Modelos de Markov de Orden Variable con Ventana Deslizante –siendo el que mejores resultados arrojó en Silva Logroño, J. et. Al. 2010, ampliando la cantidad de Patrones de Software soportados y estudiando la posibilidad de incluir la detección de síntomas de bad-smells[3] dentro de un agente inteligente en una herramienta CASE. Además de la ampliación del espectro de trabajo del algoritmo, se debe definir la condición de sugerencia necesaria para el correcto funcionamiento del asistente. Es sabido que una mala asistencia hace que el usuario pierda el interés y confianza en el agente. Para esto último se propone enriquecer la salida del algoritmo con el conocimiento de contexto existente y ayuda teórica de los patrones de diseño.

[3] Síntomas de errores de diseño que pueden indicar un problema mayor.

A continuación, en la Ilustración 2 se puede observar el flujo de conversación propuesto entre el desarrollador y el asistente.

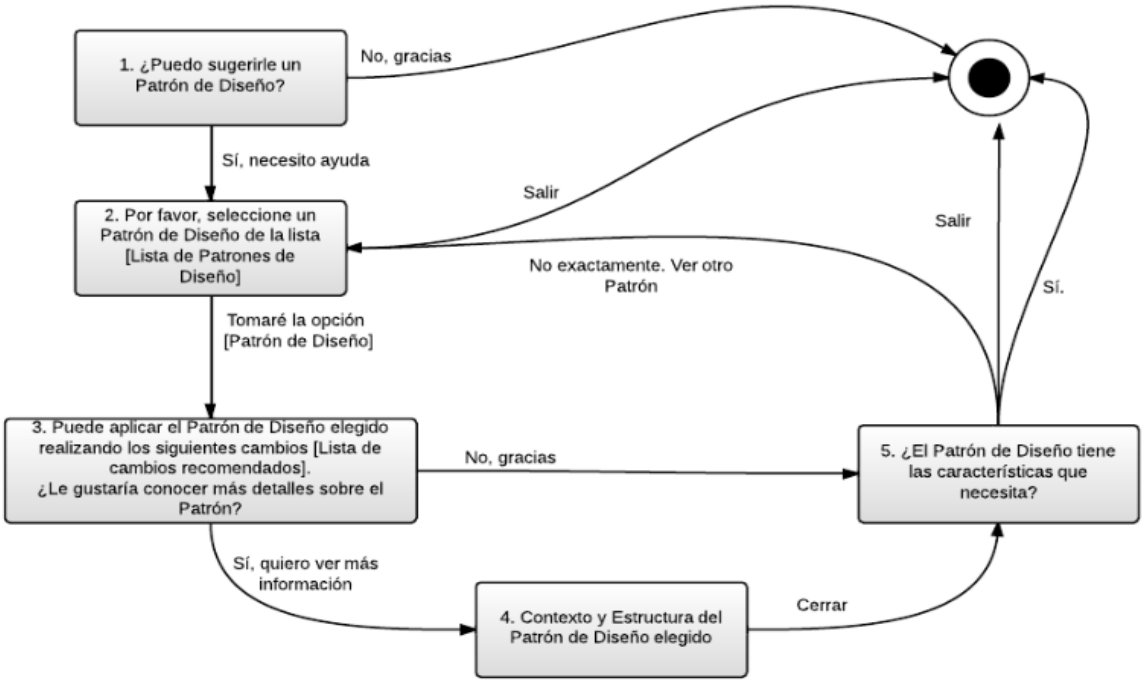

Ilustración 2: Flujo de conversación entre desarrollador y asistente

Para que la propuesta realizada se transforme en una herramienta de utilidad para los programadores, se propone tomar como base de desarrollo una herramienta CASE utilizada dentro del entorno de trabajo Eclipse. Es necesario aclarar que, si bien se toma esta decisión, se busca también independizar el diseño de manera tal de poder exportarlo a distintos entornos de programación logrando llegar a distintos usuarios.

En lo que respecta al diseño del agente y su funcionamiento se buscará un diseño que permita que el mismo se adecúe a cualquier herramienta. Asimismo, y a fin incluso de independizarse de la herramienta CASE, se buscara integrar el asistente de manera tal de poder extraer información del código del usuario y no solo de los diagramas de clase de la herramienta case.

Cómo se detalló anteriormente, al presente trabajo se lo completará con un mayor número de patrones al existente actualmente. Para el entrenamiento de la técnica de modelado, será utilizado un *Plan Corpus* generado de forma computacional.

Con el objetivo de lograr un mejor rendimiento del algoritmo se le dará la posibilidad al usuario de dar feedback a la herramienta. De esta manera el algoritmo seguirá aprendiendo a medida que el usuario lo utiliza. Específicamente se hará que cada vez que se proponga un patrón, y el usuario indique que fue correctamente identificado, la secuencia de acciones realizada hasta el momento se guardará en conjunto con las existentes para que en un futuro también se tenga en cuenta la misma.

4. Cronograma de Actividades

A continuación se detallan las actividades a llevar a cabo para el desarrollo del trabajo propuesto junto al tiempo necesario para las mismas.

Actividad	Duración estimada
Recopilación bibliográfica	4 semanas
Análisis de sistemas de asistencia a usuarios y métricas	4 semanas
Entrenamiento del Algoritmo	2 semanas
Implementación del agente de asistencia	4 semanas
Integración del agente con herramienta CASE y Eclipse	8 semanas
Realización de pruebas y ajuste del asistente	4 semanas
Evaluación del asistente	4 semanas
Generación de informe escrito (en paralelo con las actividades anteriores)	18 semanas

5. Bibliografía

Berdún, L., A. Amandi, and M. Campo. 2011. *An intelligent tutor for teaching software design patterns. Computer Applications in Engineering Education*., p.Articulo en prensa.

Biggs, J. 2003. *Teaching for quality learning at university*. Buckingham: The Society for Research into Higher Education Press, ISBN 0-335-21168-2.

Brown, William J., Raphael C. MALVEAU, Hays W. "SKIP" MCCORMICK et al. 1998. *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis. John Wiley & Sons* - ISBN: 0471197130.

Gamma, E., R. Helm, R. Johnson, and J. Vlissides. 1994. *Design patterns, elements of reusable object-oriented software*. Addison-Wesley - ISBN:0-201-63361-2.

Kautz, H. 1991. *A formal theory of plan recognition and its implementation*. Allen, J. F., Kautz, H. A., Pelavin, R., and Tenenberg, J., editors Reasoning About Plans. Morgan Kaufmann Publishers., pp.69 - 125.

Maes, P. 1994. *Agents that reduce work and information overload*. Commun. ACM. 37(7), pp.30-40.

Silva Logroño, J. Berdún, L. Armentano, M. and Amandi, A. 2010. *Análisis de secuencias discretas para la detección de Patrones de Diseño de Software*. In "Proceedings of ASAI'10", Argentine Symposium on Artificial Intelligence, (39° Argentine Conference on Computer Sciencie and Operational Research). ISSN: 1850-2784, pp. 114-125. Septiembre 2010, Buenos Aires, Argentina.