ANALYSIS PATTERNS: UNA ESTRATEGIA PARA LA REUTILIZACIÓN

Rafael Zancan, Fabricia Carneiro, José Bocanegra
Josejbocanegra@hotmail.com
Estudiantes Doctorado en Tecnología e Ingeniería del Software de la ETSII. Universidad de Sevilla.

RESUMEN

Normalmente los desarrolladores de software han centrado sus esfuerzos en aplicar estrategias de reuso en la fase de implementación, olvidando que existen otras estrategias para las demás fases del proceso, principalmente, para las anteriores. Esta situación origina que la mayoría de los proyectos sean abordados desde cero, incrementando los plazos del ciclo de desarrollo, maximizando los costos y disminuyendo la calidad de las soluciones finales. Como alternativa de solución a este problema, plantearemos el uso de analysis patterns como una de las estrategias que ha de tenerse en cuenta para la fase de análisis. Analysis patterns aumentaría la calidad y disminuiría el tiempo del análisis, proporcionando reuso desde las fases iniciales y sugiriendo, muchas veces, transformaciones de los modelos de análisis para la fase siguiente.

Palabras Claves: Reutilización, patterns, analysis patterns.

INTRODUCCIÓN

El desarrollo de software, tradicionalmente, está dividido en cinco fases: Ingeniería de requisitos, análisis, diseño, implementación y pruebas. Según Fowler [4], en la etapa de análisis la principal tarea es intentar entender el problema y crear 'modelos mentales' (mental models) para explicar o entender el problema y no solo para definir los requisitos en Casos de Uso. Estos modelos pueden ser representados y probados a través de un lenguaje de programación. Esto trae algunos beneficios como el poder ejecutar y probar el modelo, pero, también puede llevar a preocupaciones con cuestiones específicas de lenguaje, desviando el objetivo, que para esa etapa es el de entender el problema. Por esta razón son usadas técnicas de análisis y diseño independientes de tecnologías y lenguajes para poder hacer un modelado conceptual. Frecuentemente ocurren problemas comunes y recurrentes en la fase de diseño. Para ayudar a resolver estos problemas existe una serie de design patterns. Sin embargo la fase que antecede el diseño es la fase de análisis y es sabido que existe una serie de modelos (mentales/conceptuales) que pueden ser utilizados en el entendimiento del problema en diversas áreas de negocio/proceso y que también son recurrentes en diferentes proyectos. A raíz de esto fueron creados y documentados algunos patterns orientados exclusivamente a la fase de análisis, los cuales son llamados analysis patterns.

La motivación del trabajo es la posibilidad de reutilización, no sólo en las fases de diseño e implementación, sino en la fase de análisis. Concientes de la necesidad de desarrollar software con más calidad, con menor coste, en menor tiempo y teniendo en cuenta que el mantenimiento de los mismos consume gran parte de su vida útil, es necesario buscar técnicas para ayudar en este sentido. Entonces ¿Cómo conseguir ésto? ¿Cuáles son las técnicas? La adopción de patterns no es la panacea que tanto se espera, pero puede ayudar bastante en este proceso. Buscar formas de reuso, también, en las fases iniciales y anteriores a la fase de diseño del software es algo extremadamente importante. Es justamente en este punto donde patterns, más específicamente analysis patterns entran en escena.

El objetivo de este artículo es mostrar el reuso desde analysis patterns y presentar las ventajas que supone su uso en proyectos de desarrollo de software. El artículo está basado en los trabajos de Gang of Four (Erich Gamma, Richard Helm, Raph Johnson y John Vlissides), Gang of Five (Frank Buschmann, Regine Meunir, Hans Rohnert, Peter Sommerlad y Michael Stal), Martin Fowler, Kent Beck y Grady Booch y está dividido en tres partes. En la primera se presentarán algunas estrategias para el reuso y se definirán los conceptos de frameworks y de modo mas detallado, el de patterns. En la segunda serán presentados los analysis patterns, las ventajas que supone su uso en proyectos de desarrollo de software y se mostrará un ejemplo concreto de su utilización. En la última parte se listan las conclusiones y las referencias bibliográficas utilizadas para estudio e investigación sobre el tema.

ALGUNAS ESTRATEGIAS PARA EL REUSO

Frameworks. Un framework es un conjunto de clases, muchas de ellas abstractas, pensadas específicamente para atender a un dominio de problema y su énfasis está mas ligado al reuso de diseño que al reuso de código, constituyéndose entonces en una especie de mini-arquitectura para una determinada categoría de software [2]. Un framework debe ser personalizado para una aplicación específica a partir de la creación de clases concretas que heredan de las clases abstractas del framework. Con la utilización de un framework en un determinado proyecto, el analista perderá un poco de libertad en el proceso de toma de decisiones sobre su proyecto, pues siendo el framework una especie de arquitectura, mucho de este trabajo de toma de decisiones sobre el proyecto ya fue hecho por quien elaboró el framework. Se debe poner en claro que un

framework no llega a ser una aplicación completa, es apenas una especie de esqueleto para un tipo específico de aplicación.

Patterns.

1) Antecedentes: El concepto de pattern surgió a través de los trabajos del arquitecto Christopher Alexander. En sus libros el término pattern hace referencia a estructuras que pueden ser utilizadas para solucionar problemas que ocurren continuamente en el área donde él actúa [1]. La aplicación de las ideas presentadas por Alexander, en el área de desarrollo de software, fueron inicialmente difundidas por Kent Beck y Ward Cunninghan. En 1987 Kent y Ward trabajaban con Smalltalk y proyectos GUI en Textronix y decidieron aplicar algunas ideas presentadas por Alexander, en una ocasión en que estaban enfrentando problemas para terminar un proyecto. Crearon cinco patterns para guiar a nuevos programadores en Smalltalk en los proyectos GUI. Esto permitía que los jóvenes programadores pudiesen aprovechar el poder del lenguaje y no cometiesen fallos ya conocidos por programadores más diestros. El resultado de esta experiencia fue muy interesante para Kent y Ward, llevándolos a presentar estos resultados en 1987 en el evento OOPSLA'87 [7] en Orlando, a través del artículo titulado: Using Pattern Languages for Object-Oriented Programs [2].

2) Definición: Los patterns se definen como la 'documentación sobre la mejor manera de resolver un problema que es recurrente en un determinado contexto' [3]. Esto permite documentar la experiencia de los profesionales para que otros puedan partir de ellas y no tengan que preocuparse por la solución de problemas ya resueltos. La utilización de patterns en el desarrollo de software mejora la documentación, facilita el mantenimiento y aumenta el reuso. Christopher Alexander [1], define un pattern como 'una construcción de tres partes. Un contexto: sobre cuáles condiciones el pattern es empleado, las 'fuerzas del sistema': los elementos del sistema que influyen en el problema y la solución: la configuración que equilibra el sistema de fuerzas o resuelve el problema'. Para Fowler [4], pattern es 'una idea que fue útil en un contexto práctico y, probablemente, será útil en otros'. La mayoría de los autores concuerdan que un pattern está formado, por lo menos, por: el contexto en el cual el pattern es válido, las fuerzas envueltas y la solución del problema.

ANALYSIS PATTERNS

.Definición

Los analysis patterns capturan modelos conceptuales de un determinado dominio de aplicación que puedan ser útiles en una o más aplicaciones de un mismo dominio o reaprovechados en aplicaciones de otros dominios [4]. No es normal el tener reuso en la ingeniería de requisitos, donde el trabajo nunca es duplicado y cada proyecto es iniciado cual si fuese un proyecto único. Utilizar analysis patterns en esta fase permite que el proyecto ya sea empezado a partir de una consulta y de la aplicación de algunas soluciones de calidad para problemas recurrentes de análisis en el dominio del problema en cuestión. Por esto, no se está perdiendo tiempo creando una solución que ya existe para un problema determinado. Eso permite comenzar proyectos ganando así tiempo y aumentando la calidad final del análisis.

En la figura 1 es posible ver que en el espacio del problema, al tomar el problema de un nivel concreto y llevarlo para un nivel abstracto, se puede hacer uso de analysis patterns [6]. El uso de este tipo de pattern ayuda a entender el problema en cuestión. En el espacio del problema no hay preocupación con la tecnología que será utilizada, por tanto no hay dependencia alguna de los modelos empleados con cualquier tecnología de

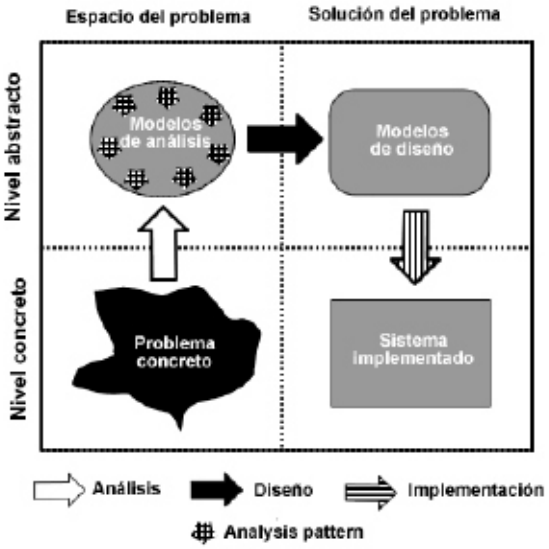

Fig. 1. Analysis patterns en el proceso de desarrollo de sofrware. Basada en [6]

software. El uso de analysis patterns irá a disminuir el tiempo de análisis para entender el problema en cuestión y se habrá aplicado una estrategia de reutilización antes de la fase de diseño. Otra ventaja de los analysis patterns es que facilitan la transformación de los modelos de análisis a modelos de diseño para crear soluciones de calidad a problemas comunes.

Beneficios

Andreas Shultz en [6] aplicó analysis patterns para el desarrollo de varios sistemas de información en el proyecto denominado Universidad Virtual de la Vienna University of Economics and Business Administration. Shultz tomó el analysis pattern virtual library with active agents y lo reusó para construir varios de los sistemas de información del proyecto mencionado. Para cuantificar el beneficio del uso de analysis patterns, comparó el esfuerzo necesario para implementar dos sistemas de información reusando patterns con la estimación del esfuerzo necesario para construir los sistemas partiendo de cero. Para la estimación fue utilizado el Organic

Mode of Software Development de COCOMO obteniendo de esta forma la siguiente ecuación, dónde

Proyecto	Tamaño KDSI	Tecnicos	MM
Calendario de eventos	4.02	1	0.5
Biblioteca digital	7.61	2	6

Tabla 1. Proyectos Analizados

2.4 y 1.05 son los parámetros del modelo, KDSI son las líneas de código sin comentarios divididas entre 1000, y MM es la cantidad estimada de hombres mes necesaria para desarrollar el sistema: MM = 2.4KDSI1.05. La información analizada del proyecto como el tamaño, el número de desarrolladores y el esfuerzo necesario para

	Unidades	LOC	LOC reusadas	%
PERL	LOC	3743	3502	93.56%
HTML	LOC	278	192	69.06%
Total LOC	LOC	4021	3694	91.67%
Total MM	MM	10.35	9.46	91.40%

Tabala 2. C odigo Reusado En El Calendario De Eventos

su implementación son mostrados en la tabla I.
El primer sistema desarrollado fue un calendario de eventos, creado a partir del analysis pattern virtual library. En la tabla II se detalla la cantidad de código que fue reutilizada. Un total de 93% de líneas de código en PERL y 69% en HTML fueron reutilizadas. El modelo estimó una reducción superior al 90% tanto en hombres/mes como en líneas de código. El segundo sistema desarrollado fue la biblioteca digital, creada tomando como base, nuevamente, el analysis pattern virtual library. Para este sistema, dadas sus características fue necesario incluir algunos cambios no contemplados en el pattern. La tabla III muestra el total de código reutilizado en este proyecto. Más de 53% de líneas de código en PERL y más de 33% en HTML fueron reutilizadas. Esto significa un reuso total de código de 51% y una reducción estimada del esfuerzo de hombres mes cercana al 50%.

Ejemplo de un Analysis pattern: a simple pinboard
1) Contexto: Una compañía multinacional tiene grupos de trabajo dispersos alrededor del mundo. Los grupos deben trabajar juntos para cumplir los objetivos de la empresa, por tanto, la comunicación entre ellos es vital. No es posible que los miembros de los grupos se reunan a menudo porque esto consumiría tiempo y recursos, pero es necesario que los grupos intercambien mensajes,

	Unidades	LOC	LOC reusadas	%
PERL	LOC	6954	3697	53.02%
HTML	LOC	662	224	33.84%
Total LOC	LOC	7616	3911	51.35%
Total MM	MM	20.23	10.05	49.68%

Tabla 3. Código Reusado En la Biblioteca Digital

se comuniquen asíncronamente y construyan juntos una base común de conocimiento. El problema radica en que los medios tradicionales de comunicación como cartas, teléfonos, faxes y correos electrónicos no satisfacen esas necesidades.

2) Fuerzas de sistema: La comunicación de forma eficiente del grupo es vital. No es viable la realización de reuniones presenciales periódicas. Los medios tradicionales de comunicación no son lo más idóneo

3) Solución: Usar un pinboard para la comunicación del grupo. Esto permitiría almacenar información de todos los miembros del grupo para que siempre esté disponible. Los mensajes estarían compuestos por un cuerpo y el nombre del autor. Será posible la búsqueda de mensajes específicos de acuerdo a su autor o a una palabra del texto.

Los siguientes actores interactuarán con el sistema:
Usuarios: Recuperarán información del pinboard
Proveedores de información: Agregarán información al pinboard
Administrador: Será el responsable de la operación del pinboard.
Consecuencias con el uso del pinboard:
Comunicación asíncrona: El trabajo no será interrumpido permanentemente debido a llamadas telefónicas.
Generación de una fuente común de conocimiento: Los mensajes serán archivados para construir una base de datos colectiva.
Como comentario al anterior ejemplo podemos decir que la comunicación entre los grupos de trabajo es un problema recurrente en las empresas. Ocurre continuamente que cada empresa crea su propia solución. Esto conlleva que no exista un patrón de solución conocido que pueda ser reutilizado. El analysis pattern a simple pinboard es una propuesta de solución a este problema.

Como puede verse en [6], simple pinboard sugiere patterns para la transformación del modelo de análisis del nivel abstracto al modelo de diseño en el espacio de solución del problema.

CONCLUSIONES

El conocimiento de técnicas de reuso en las diversas fases del proceso de desarrollo de software, es extremadamente importante. El reuso hace mucho tiempo es explorado, de manera más intensa, en la fase de implementación, y de un tiempo acá, en la fase de diseño a través del uso de design patterns. Pocos conocen o aplican técnicas (patterns) que puedan ser utilizadas en las fases anteriores al diseño. Los analysis patterns cuando son utilizados, proporcionan reuso desde antes de la fase de diseño e implementación, generando ganancia al proceso. Un mismo analysis pattern puede ser utilizado en los más variados sistemas, trasciende dominios de aplicaciones y acelera y califica

la fase de análisis. Algunos analysis patterns llegan a sugerir design patterns para ser utilizados en la fase de diseño.

Los proyectos generalmente parten de cero. Esta cultura necesita ser modificada de modo que los profesionales conozcan y busquen patterns de los más variados tipos (architectural patterns, analysis patterns, design patterns, idioms, etc.), para generar un sistema de mayor calidad, mas flexible, con un menor tiempo de desarrollo, y ciertamente menos costoso.

REFERENCIAS

[1] ALEXANDER, Christopher. The Timeless Way of Building. Oxford University Press. 1979.
[2] APPLETON, Brad. Patterns and Software: Essential Concepts and Terminology. Disponible en http://www.enteract.com/bradapp. 2005.
[3] BUSCHMANN, Frank at Al. Pattern-Oriented Software Architecture: A System of Patterns. Wiley. 1996.
[4] FOWLER, Martin. Analysis patterns: Reusable Object Models. Addison Wesley. 1996.
[5] GAMMA, Erich et Al. Design patterns: Elements of Reusable Object- Oriented Software, Addison Wesley.2002
[6] SCHULZ, Andreas et Al. Software engineering with analysis patterns. Disponible en http://wwwai.wuwien.ac.at/ hahsler/research/virlib working2001/virlib/virlib.html. 2006.
[7]Object-Oriented Programming, Systems, Languages and Applications 87