Soporte a la actividad de diseño basado en patrones de diseño

Luis Roqué Fourcade, Liliana Arakaki
Departamento de Informática – Facultad de Ciencias Físico Matemáticas y Naturales
Universidad Nacional de San Luis
Ejército de Los Andes 950, CP D5700HHW, San Luis, Argentina, +54 (0266) 4520300
email: araroq@unsl.edu.ar, liliana.arakaki@yahoo.com

Resumen

Una de las direcciones en la que se trabaja intensamente en el proceso evolutivo de la ingeniería de Software intenta alcanzar un estado similar al de otras ingenierías más desarrolladas. Éstas han logrado un alto grado de reusabilidad con un rango amplio de granularidad que va desde componentes atómicos hasta productos finales terminados, pasando por componentes estructurados a partir de otros. El gran desafío es avanzar en el conocimiento sobre diferentes dominios (tanto de aplicación como tecnológicos) de manera tal que éste pueda ser traducido en componentes atómicos y/o estructurados susceptibles de ser reutilizados en los diferentes procesos de desarrollo.

El presente trabajo, describe líneas de investigación en la dirección mencionada, que intentan soportar el proceso de diseño mediante el uso de patrones de diseño, a partir de la hipótesis de que éstos, además de cumplir con los requerimientos mencionados, pueden cumplir un rol fundamental para el objetivo de soportar al proceso de diseño si consideramos que un patrón de diseño es a la vez un objeto y una regla que indica cuándo y cómo reutilizarlo.

Palabras clave: Ingeniería de software - Diseño – Patrón de diseño - Derivación de Diseño - OOD - UML – Modelado - Transformación - MDA - QVT - XSLT

Contexto

Las líneas de investigación presentadas en este Trabajo, se desarrollan en el marco del proyecto de investigación 'Ingeniería de software: aspectos de alta sensibilidad en el ejercicio de la profesión de ingeniero de software', de la Facultad de Ciencias Físico-Matemáticas y Naturales de la Universidad Nacional de San Luis, bajo el número 22F222.

Introducción

Desde sus inicios, la Ingeniería de Software, en su proceso evolutivo, ha recorrido un camino zigzagueante en las diferentes dimensiones por las que ha transitado. En ese camino, se han producido innumerables métodos, técnicas, estándares y especificaciones de modelos de aplicación, de arquitectura y del proceso de desarrollo.

En la variedad de resultados producidos a lo largo de este proceso y en particular, en la dirección mencionada en el resumen, se destaca el enfoque de patrones de diseño. Sin embargo, un asunto al menos curioso, radica en el hecho de que, a pesar de que pareciera ser el recurso ideal en la dirección de representar conocimiento de manera altamente reusable y mediante mecanismos aptos para especificar una arquitectura de software, no ha alcanzado aún un uso generalizado. Encontrar la razón de esta situación no es una tarea fácil. Sin embargo, examinando el proceso evolutivo, podemos encontrar allí uno de los principales motivos que, por lo menos, no alienta el uso masivo y estándar de este enfoque. Después de una revisión amplia en publicaciones de todo tipo, comprobamos que no existe, a esta altura, una definición precisa y estándar acerca de qué es Diseño de Software. Es decir, si bien existen bastantes definiciones, la mayoría adolece de diferentes problemas: conceptuales, de completidud, etc. También, presentan en común el ser demasiado abstractas y el hecho de no ser aceptadas como un estándar de uso generalizado, lo cual impide que puedan ser utilizadas como fundamento de un posible soporte a la actividad.

Siguiendo con el enfoque propuesto en nuestros trabajos presentados en WICC 2012 [1] y en 41JAIIO [2], reafirmamos aquí el rol esencial del enfoque de patrones en la actividad de diseño de software, especificando la correspondencia entre ambos y planteando líneas de investigación en la dirección de soportar la actividad con patrones de diseño. Lo hacemos a partir de tal correspondencia y del hecho de que los patrones de diseño son tanto, un objeto como la regla que especifica cuándo y cómo reutilizarlos. Esta característica los convierte en el recurso ideal para el objetivo de desarrollar soportes a la actividad de diseño de software.

Para este fin, desarrollamos a continuación, los conceptos que fundamentan la propuesta y luego

presentamos de manera detallada, las líneas de investigación sugeridas.

Diseño de Software

Contar con una buena definición de la actividad de diseño de software resulta clave para el objetivo de proveer un soporte a la actividad de diseño, particularmente a la ejecución de la actividad. Sin embargo, después de haber realizado una búsqueda exhaustiva en la literatura existente acerca de una definición de Diseño de Software, sólo nos ha sido posible encontrar definiciones en publicaciones de libros y artículos cuyo objetivo no es esencialmente esta definición en sí misma sino que, al igual que en este trabajo, les resulta por lo menos necesario considerar una. Además, prácticamente ninguna de las definiciones encontradas resulta totalmente satisfactoria para lo que requerimos aquí. Algunas adolecen de ambigüedad, otras de falta de completitud, otras son sólo una expresión abstracta que abarca un gran número de posibilidades pero que en ningún caso lo hace con el grado de precisión que requerimos.

Sin embargo, hemos podido encontrar también algunas realmente interesantes, como las que referimos más adelante, y algunas coincidencias generalizadas que son de interés en este caso. Por ejemplo, buena parte de los autores coinciden en definir al Diseño de Software desde dos perspectivas: como actividad y como objeto (el resultado de la actividad) y, la mayoría, sobre todo en la actualidad, prefieren referirse al resultado de la actividad como 'Arquitectura de Software' en lugar de 'Diseño de Software', para significar que no se trata sólo de la descomposición funcional de un sistema de software en un conjunto de componentes cooperando sino que, durante la actividad de diseño se construye una 'Arquitectura de Software'. Es decir, la complejidad propia de los requisitos funcionales combinada con la resultante de incluir componentes y diseños específicos requeridos para la satisfacción de requisitos no funcionales, impone la necesidad de enfocarse en el diseño cuidadoso de la estructura interna de los componentes, sus responsabilidades, funcionalidad e interfaces, y en las múltiples relaciones que existen entre ellos y formas en que colaboran.

Dada la coincidencia de intereses en ambos casos con el trabajo, incluimos a continuación las definiciones proporcionadas por los autores, Paul Ralph y Yair Wand [3] y Frank Buschmann, et al. [4], en las cuales basaremos nuestra propuesta. En el caso de Ralph y Wand, éstos sintetizan conclusiones basadas en una investigación muy completa en la literatura existente sobre diseño y enfocan su propuesta, con un enfoque formal e independiente del dominio, en la estructura y organización del proceso de diseño. En el caso de Buschmann, además de enfocar su definición específicamente en el dominio de la Ingeniería de Software, lo hace desde la necesidad de contar con una definición para presentar un Sistema de Patrones de Diseño de Software y su relación con la arquitectura de software [4].

<u>Propuesta de Paul Ralph y Yair Wand</u>

Paul Ralph y Yair Wand proponen una definición del concepto de diseño en un nivel de abstracción independiente del dominio y lo hacen de manera explícita desde ambas perspectivas [3]:

- *Diseño como objeto*: especificación de un objeto (el objeto de diseño), desarrollada por un agente, destinado a cumplir objetivos, en un entorno particular, utilizando un conjunto de componentes primitivas, satisfaciendo un conjunto de requisitos, sujeto a restricciones.

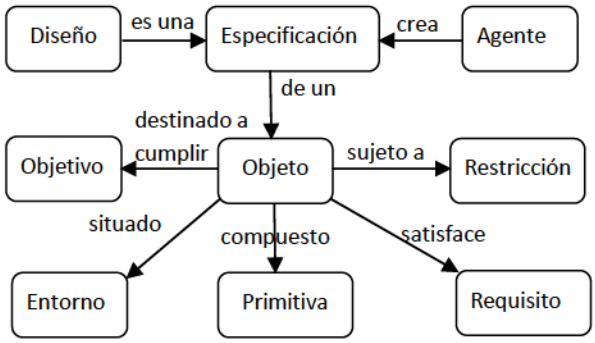

Fig. 1 Modelo conceptual de diseño ***[3]***

Esta definición de diseño como objeto introduce una dependencia de éste respecto de cinco elementos claves: objetivos, entorno, componentes primitivas, requisitos y restricciones.

- *Diseño como actividad*: actividad, desarrollada en un entorno (en el que se desempeña el agente), cuyo objetivo consiste en crear un diseño.

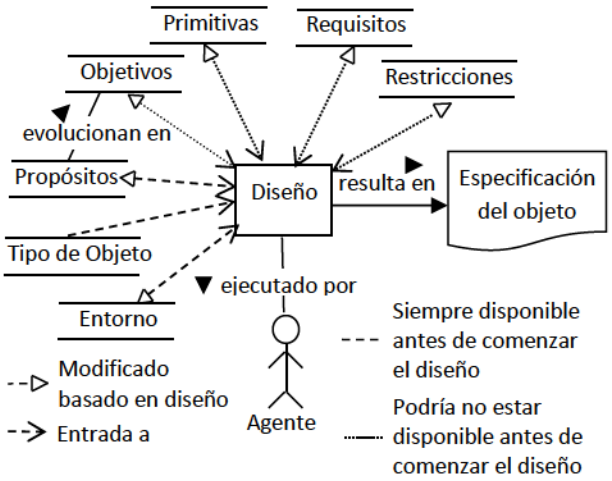

Fig. 2 Nivel contextual del modelo conceptual de diseño ***[3]***

Esta definición de diseño como actividad, hace lo propio, al establecer una dependencia absoluta respecto de la primera definición como objeto.

Propuesta de Frank Buschman et al.

Frank Buschman, en su libro Pattern-oriented software architecture: A system of patterns [4], presenta un enfoque muy interesante en la misma dirección del trabajo. Alienta el desarrollo de aplicaciones cuyas arquitecturas hereden una impronta definida por patrones de diseño. Para este objetivo, en su propuesta, se distinguen tres líneas:

- una definición de Diseño de Software (compatible con la propuesta por Ralph y Wand) que destaca el resultado de la actividad como una arquitectura de software,
- un sistema y un catálogo de patrones en el que, además de la extensión y variedad del mismo, destacan una categorización en dos niveles (abstracción y área funcional a la que aplican) y la capacidad (no descripta formalmente) de composición de patrones.
- el intento (aunque no formal) de establecer una correspondencia entre las estructura de patrones de diseño y de arquitectura de software.

Estas líneas revisten interés en el presente trabajo. La primera, porque utilizamos esta definición junto a la de Ralph y Wand, para especificar nuestra propuesta de definición. La segunda, porque destaca correspondencias con el diseño de software desde un punto de vista estructural y estático, el de la Arquitectura de Software, y el trabajo, lo hace desde un punto de vista de la actividad de diseño, cuyo resultado es precisamente una Arquitectura de Software. Y, la tercera, porque esa correspondencia (a pesar de la informalidad) cumple un rol importante para sustentar nuestra propuesta de soporte a la actividad.

A continuación presentamos de Buschmann, que tendremos en cuenta en el trabajo.

- *Diseño de Software* es la actividad ejecutada por un desarrollador de software que resulta en la arquitectura de software de un sistema. Se ocupa de especificar los componentes de un sistema de software y las relaciones entre ellos dentro de propiedades funcionales y no funcionales dadas.
- *Arquitectura de Software* es una descripción de los subsistemas de un sistema de software y de las relaciones entre ellos. Subsistemas y componentes son típicamente especificados en diferentes vistas para mostrar las propiedades funcionales y no funcionales relevantes de un sistema de software. Es el resultado de la actividad de diseño.

Estas definiciones identifican tres componentes esenciales: la actividad de diseño, un desarrollador de software, y la arquitectura de software, consistentes con la actividad de diseño, el agente y el objeto de diseño en la definición de Ralph y Wand.

Luego continúa la definición refiriéndose al diseño como actividad enunciando, que 'se ocupa de especificar los componentes de un sistema de software y las relaciones entre ellos, dentro de propiedades funcionales y no funcionales dadas'. Este enunciado, es consistente con las dependencias que Ralph y Wand identifican del diseño respecto de los requisitos. También es consistente con el enfoque orientado a componentes de Ralph y Wand, circunscribiendo la especificación de diseño como una colaboración entre componentes.

Si bien estas definiciones no especifican de manera explícita correspondencia o especialización respecto del resto de dependencias especificadas en la propuesta de Ralph y Wand, éstas son consideradas de manera implícita en la formulación tanto de los patrones de diseño como de la correspondecia sugerida entre éstos y la Arquitectura de Software.

Patrones de Diseño de Software

Según Christopher Alexander, "un patrón describe un problema que ocurre una y otra vez en nuestro entorno, y describe la esencia de la solución a ese problema, de modo tal que pueda utilizarse esta solución un millón de veces más, sin siquiera hacerlo de la misma manera dos veces" [5].

Si bien Alexander hablaba de 'Ciudades y edificaciones' estas afirmaciones son válidas también en el dominio de la ingeniería de software. Difieren el lenguaje y los términos que utilizamos para especificarlos, los cuales son necesaria y convenientemente dependientes del dominio, en este caso, del dominio de la ingeniería de software.

Precisamente, en el dominio de la ingeniería de software, Buschmann define patrones de diseño de software de la siguiente manera [4]:

- *Un Patrón de arquitectura de software* describe un problema particular y recurrente de diseño que surge en contextos de diseño específicos, y presenta un esquema genérico y probado para su solución. El esquema de solución se especifica mediante la descripción de sus elementos constitutivos, responsabilidades y relaciones, y las formas en las que éstos colaboran.

Las definiciones de diseño y de patrones aportadas por Buschmann, coinciden en estar construidas en torno a la definición de Arquitectura de Software, también aportada por él. La de diseño, describiendo al diseño como una manera concreta de obtenerla y, la de patrones, como una manera abstracta de obtenerla. Esta coincidencia, constituye un aspecto clave en nuestro trabajo.

Líneas de Investigación y Desarrollo

Revisitando el resumen, rescatamos el objetivo de *soportar a la actividad de diseño de software,* mediante el uso de Patrones de Diseño como recurso apto para afrontar el desafío de avanzar en el conocimiento sobre diferentes dominios, tanto de aplicación como tecnológicos, de manera tal que éste pueda ser traducido en componentes atómicos y/o estructurados, susceptibles de ser reutilizados.

Comparando además las definiciones de Diseño y de Patrones de Diseño de Software consideradas, surge que un patrón actúa como un modelo que:

- anticipa un conjunto de decisiones de diseño y
- acota el universo de evaluación para otro cuyas decisiones son esencialmente dependientes del dominio de la aplicación para la cual se está ejecutando la actividad de diseño.

En consecuencia, podemos considerar a un patrón como una abstracción cuya concretización depende del dominio de reutilización. Es decir que, para su concretización, será necesario identificar en el dominio, especializaciones de los componentes incluidos en el esquema genérico de solución y valores que condicionan el equilibrio entre requisitos y las fuerzas direccionadas por el patrón.

De lo expresado hasta aquí podemos concluír, que es posible desarrollar soportes basados en patrones de diseño de software que, dado algún estado en el proceso de diseño y a partir de especificaciones válidas, sean capaces de anticipar algunas decisiones, y de asisitir al diseñador con universos acotados, en otras. Proponiendo inclusive, valores por omisión basados en reglas y valores heurísticos. Claramente, por los soportes mencionados, nos referimos al proceso de concretización de la asbtracción representada por el patrón.

Así, en primer lugar, reformulamos las definiciones de diseño que hemos considerado, a través de los modelos conceptual de diseño y contextual de la actividad que exponemos a continuación.

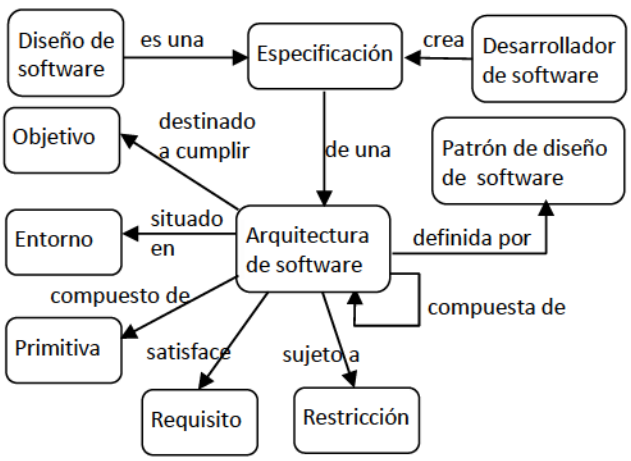

Fig. 3 Modelo conceptual de diseño de software

Como se observa en el modelo (ver *Fig. 3*), nuestra propuesta especializa la definición de Ralph y Wand, desde la perspectiva de la definición propuesta por Buschmann, vinculando el modelo al dominio de Ingeniería de Software y expresando el objeto de diseño como una Arquitectura de Software. También, se puede observar la extensión que agrega la dependencia de la Arquitectura de Software respecto de patrones de diseño, la cual, especializa la definición de la misma como una composición de arquitecturas de software donde cada una está definida por algún patrón de diseño.

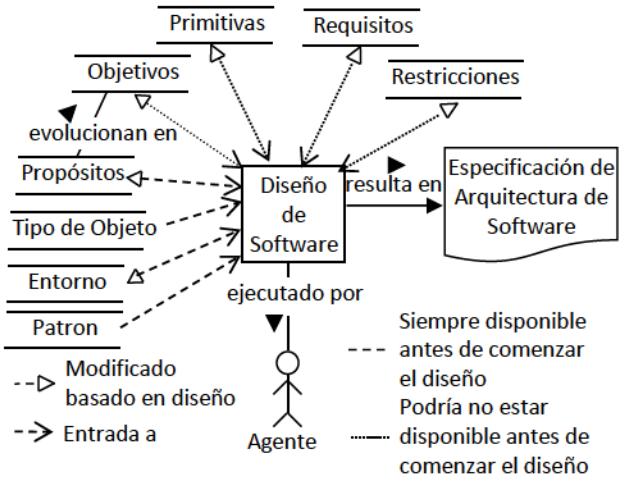

Fig. 4 Nivel contextual del modelo conceptual

En el modelo de nivel contextual (ver *Fig. 4*), extendemos el de Ralph y Wand, adicionando la entrada de patrones de diseño y especializando la expresión de salida de la actividad como la especificación de la Arquietctura de Software. Por simplicidad, el modelo no exhibe la composición ni la definición a partir de patrones. Lo hacemos así, porque tanto la composición como la dependencia están explicitadas en el modelo conceptual y en la entrada de patrones de diseño a la actividad.

Alexander afirma también, que un patrón de diseño es a la vez una cosa, la cual ocurre en el mundo, y la regla que nos dice cómo y cuándo debemos crearla. Es decir, es tanto la descripción de una cosa como la descripción del proceso que generará esa cosa [6].

Claramente, 'la descripción de una cosa' que refiere Alexander, en nuestro dominio es la especificación de una arquitectura y, 'la descripción del proceso', no es otra cosa que la especificación de la actividad.

A partir entonces, de los modelos conceptual de diseño y contextual de la actividad de diseño propuestos y, de la traspolación de la afirmación de Alexander al dominio de la Ingeniería de Software, proponemos las siguientes líneas de investigación:

1) Desarrollar especificaciones de patrones de amplia aceptación como Model/View/Controller

(MVC), Composite, Observer, etc., utilizando estándares como MDA [7] o XMI [8].

2) Especificar mecanismos de recolección de especializaciones de componentes abstractos de patrones y de valores que condicionan la resolución de las fuerzas direccionadas por éstos. Inclusive, reglas y valores heurísticos que permitan definir valores por omisión.
3) Desarrollar proptotipos de soporte a la actividad de diseño de software mediante el desarrollo de transformaciónes basadas en recursos como QVT [9] o XSLT [10], o basadas en lenguajes tradicionales de programación como Java o C++.
4) Especificar soportes a la actividad de Diseño de Software a partir de los prototipos desarrollados.

Resultados y Objetivos

Tanto las líneas de investigación y desarrollo propuestas como la formulación de una definición de Diseño de Software a través de los modelos conceptual y de contextual presentados aquí, forman parte de actividades de investigación que se desarrollan en el marco del proyecto mencionado previamente. En ese contexto, estas y otras actividades que se están desarrollando, se encuentran en diferentes estadíos. Algunas ya finalizadas, como por ejemplo transformaciones entre instancias de modelos a partir de sus especificaciones MDA y/o XMI, análisis de correspondencias entre especificaciones de diferentes modelos, tanto semánticas (en función del modelo) como sintácticas (en función de las construcciones sintácticas), etc. Otras, se encuentran en curso, como por ejemplo más experimentos de transformaciones, investigación y especificación de reglas y valores heurísticos para la toma de decisiones de diseño, extensiones a meta-modelos MDA de los diferentes modelos para soportar transformaciones, etc.

En el transcurso del presente año, esperamos desarrollar casos de prueba de refactorizaciones de diseño apoyadas en patrones de diseño de complejidad acotada como Composite, Observer, MVC, etc. y publicar resultados para su evaluación.

En el mediano plazo, esperamos avanzar en la interpretación del enfoque de Patrones de Diseño de Software en el ámbito de otros dominios, como Orientación a Objetos o Gramáticas de Lenguajes, experimentar traspolando propiedades válidas de estos dominios y también publicar resultados para someterlos a evaluación.

También, se encuentran en curso, trabajos de maestría y tesis de licenciatura (algunos finalizados), que experimentan aspectos como los mencionados.

Formación de Recursos Humanos

En el área de la temática planteada, se desarrollan diferentes actividades que van desde la inclusión de trabajos prácticos, laboratorios y ejercicios de investigación, en materias de las carreras afines hasta el desarrollo de tesis y trabajos finales, algunos finalizados y otros en curso, en las mismas.

En particular, durante el año 2014, finalizó una tesis de licenciatura [11] que experimentó con este enfoque durante el desarrollo de un prototipo en el marco de la tesis y se inició una tesis de maestría cuyo objetivo es central a la temática planteada.

Referencias

[1] L. Roqué Fourcade y L. Arakaki, “Derivando el diseño a partir de especificaciones de requisitos basadas en Casos de Uso”, WICC 2012, 2012.

[2] L. Roqué Fourcade, “Derivando el Diseño a Partir de Especificaciones de Requisitos Basadas en Casos de Uso”, 41JAIIO, 2012.

[3] P. Ralph y Y. Wand, “A Proposal for a Formal Definition of the Design Concept”, Design Requirements Workshop, Lecture Notes on Business Information Processing, Berlin, 2009.

[4] F. Buschmann et al., “Pattern-Oriented Software Architecture: A System of Patterns”, Wiley, 2001.

[5] C. Alexander et al., “A Pattern Laguage - Towns . Buildings . Construction”, Oxford University Press, 1977.

[6] C. Alexander, “The timeless way of building”, Oxford University Press, 1979.

[7] Object Management Group, “OMG Model Driven Architecture (MDA)”, http://www.omg.org/mda/

[8] Object Management Group, XML Metadata Interchange (XMI), http://www.omg.org/spec/XMI/.

[9] Object Management Group, Query/View/Transformation(QVT), http://www.omg.org/spec/QVT/index.htm.

[10] World Wide Consortium W3C, XSL Transformations(XSLT), http://www.w3.org/TR/xslt.

[11] E. Bernardis, Desarrollo de una Herramienta para la Administración y Control de Proyectos Extendiendo el Concepto de Indicadores, UNSL, 2014.