Artefactos de diseño en el paradigma orientado a aspectos

Design artifacts in the aspect-oriented paradigm

LSC. María Zavala Hurtado*

M.C Rene Edmundo Cuevas Valencia**

Fecha de recepción: 14 de abril de 2013
Fecha de aprobación: 5 de mayo de 2013

Resumen

La Ingeniería de Software Orientada a Aspecto es un enfoque de la Ingeniería de Software que se ha diseñado para hacer frente al problema de requerimientos-componentes.

El desarrollo orientado a aspectos toma las principales preocupaciones de un problema y son consideradas como aspectos transversales, los cuales son agrupados en módulos separados.

En la actualidad la investigación y el desarrollo en la orientación a aspectos se enfoca, en su mayor parte, en la programación orientada a aspectos olvidando las fases de análisis y diseño. Por tal motivo, el presente trabajo pretende describir la fase de diseño que modela los artefactos para el diseño de las competencias transversales.

Palabras clave: Artefactos, competencias transversales, Programación, ingeniería de software, paradigma, UML, OOP, POA.

* Universidad Autónoma de Guerrero Unidad Académica de Ingeniería Av. Lázaro Cárdenas S/N, CU. Correo electrónico: mzh27@hotmail.com

** Universidad Autónoma de Guerrero Unidad Académica de Ingeniería Av. Lázaro Cárdenas S/N, CU. Correo electrónico: reneecuevas@hotmail.com

Abstract:

The Engineering Aspect Oriented Software is an approach to software engineering that is designed to deal with the problem of requirements-components.

The development is oriented to aspect that takes the main problems and are considered as aspects cross-cutting issues, which are grouped in separate modules.

At present the research and development in the aspect orientation is focuses, for the most part, in the aspect-oriented to the programming of the analysis and design phases. Therefore, the present study describes the design stage modeling artifacts for the design of generic skills.

Key words: Artifacts, transferable skills, programming, software engineering paradigm, UML, OOP, AOP.

1. Introducción

La ingeniería de software orientada a aspectos (AOSE) es un paradigma de desarrollo de nueva creación, que aún no ha alcanzado su máxima implementación en las diferentes etapas que integra el desarrollo de proyectos.

El presente artículo describe el problema, la justificación y objetivos que permitirán el logro de diseño de artefactos capaces de apoyar en la fase de diseño del paradigma orientado a aspectos. Así como también hace mención de la hipótesis y variables, marco teórico y la estructura capitular que la investigación estará estructurada.

No olvidando el tipo de investigación a realizar y las conclusiones del presente protocolo.

2. Delimitación del problema

2.1. Problema de investigación

2.1.1. Enunciado

Para Sommerville (2011) los grandes sistemas sufren de complejidad entre los requerimientos y los componentes, ya que un componente puede estar formado por varios requerimientos o viceversa, un requerimiento está integrado por varios componentes.

Lo antes mencionado causa conflictos para la reutilización de componentes, ya que éstos, al querer reutilizarlos, en la mayoría de las ocasiones, deben modificarse, lo que implica comprender y modificar varios componentes, provocando pérdida de información, pérdida de tiempo o que la actividad sea más laboriosa.

La ingeniería de software orientada a aspectos (AOSE- Aspect-Oriented Software

Engineering) es un enfoque al desarrollo de software que se ha diseñado para hacer frente al problema de requerimientos-componentes, con esto se logran desarrollar programas más fáciles de mantener y reutilizar.

La AOSE se basa en abstracciones llamadas aspectos, esto quiere decir, según Fuhrer Patrik y Pasquier Jacques (2006) en su caso de estudio, que el desarrollo orientado a aspectos toma las principales preocupaciones de un problema y son consideradas como aspectos transversales, enseguida los agrupa en módulos separados, los cuales, son llamados automáticamente por el sistema; a diferencia de la programación orientada a objetos (OOP- Objetc Oriented Programming) que encapsula las preocupaciones para ser compartidas por otros componentes, elementos o partes que integran la aplicación, el problema de éste encapsulamiento es que no todos los "accesorios" no necesitan todas las operaciones de ese negocio.

En la actualidad, la investigación y el desarrollo en la orientación a aspectos se enfoca, en su mayor parte, en la programación orientada a aspectos (AOP- Aspect Oriented Programming), lo que ha logrado el diseñando de lenguajes como son Aspectj, Spring AOP, JBoss AOP, entre otros. Colyer y Clement (2005) mencionan que las grandes compañías ya usan la programación orientada a aspectos en sus procesos de producción de software, sin embargo, las competencias transversales han ocasionado conflictos en otras de las etapas del proceso de desarrollo de software.

Por tal motivo, los investigadores indagan sobre cómo utilizar la orientación a aspectos en la ingeniería de requerimientos y diseño de sistemas, la forma de poner a prueba y verificar programas orientados a aspectos.

Enfocados en el área de diseño, hasta el momento, se han utilizado técnicas y artefactos de la programación orientada a objetos (OOP). Pero aún no se cuenta con los artefactos que describan, precisamente, las competencias transversales.

2.1.2. Formulación del problema

¿Cómo favorece la creación de artefactos de competencias transversales en la fase de Diseño en Ingeniería de Software para la programación orientada a aspectos?

¿Las compañías u organizaciones cuentan con los conocimientos necesarios para la identificación de las competencias transversales?

2.2. Objetivos de la investigación

Los siguientes son los objetivos que orientan esta investigación:

2.2.1. Objetivo general

Diseñar artefactos capaces de describir las competencias transversales en la Ingeniería de Software Orientada a Aspectos.

2.2.2. Objetivos específicos

Para llevar a cabo el logro del objetivo general es necesario cumplir lo siguiente:

1. Conocer cómo trabaja la Ingeniería de Software Orientada Aspectos (AOSE).
2. Describir los elementos necesarios para trabajar con la AOSE.
3. Clasificar los diferentes tipos de competencias.
4. Conocer el lenguaje de Programación Orientado a Aspectos.
5. Diseñar artefactos que describan los tipos de competencias.

2.3. Justificación y delimitación de la investigación

La presente investigación busca ser un aporte a la Ingeniería de Software Orientada a Aspectos y así mismo convertirse en una propuesta de formulación de artefactos orientados a apoyar a la ingeniería de software tanto al área de diseño como de programación para el desarrollo de aplicaciones de software.

El estudio se orienta al paradigma de la ingeniería de software orientada a aspectos, en la fase de diseño.

2.4. Alcances

La ingeniería de software orientada a aspectos (AOSE) es un paradigma de desarrollo de nueva creación, que aún no ha alcanzado su máxima implementación en las diferentes etapas que integra el desarrollo de proyectos.

En la ingeniería de software, las etapas de análisis, diseño, construcción y pruebas son de vital importancia para entregar un software de calidad, que cumpla con los requerimientos que el cliente solicita. Por eso, cuando una de estas etapas no se lleva a cabo o no se realiza de manera adecuada, el software tiene el riesgo de no cumplir con el objetivo específico.

Como ya se mencionó, la AOSE ha resuelto problemas enfocados en la programación (AOP), pero no en el diseño, por tal motivo, con base a la programación ya existente se pretende desarrollar una propuesta de artefactos, en la fase de diseño, para la modelación de las aplicaciones (competencias transversales) orientadas a aspectos.

2.5. Tipo de investigación

Esta investigación será de tipo Documental, debido a que analiza información escrita sobre el tema objeto de estudio. Además, será de tipo Estudio de Caso porque se analiza una unidad específica.

2.6. Hipótesis de la investigación

La utilización de artefactos en el desarrollo de sistemas, en la fase de Diseño, ayudará al ingeniero de software a interpretar las competencias transversales en diseños que facilitarán la programación y utilización de éstas.

2.7. Variable

- *Variable independiente:* la utilización de artefactos en el desarrollo de sistemas.
- *Variable dependiente:* ayudará a interpretar las competencias transversales en diseños que facilitarán la programación.
- *Variable interviniente:* capacidad de diseñar o programar del ingeniero de software, tipo de software a desarrollar, lenguaje de programación.

3. Marco teórico

3.1. Estudio del arte

Para [1] la ingeniería del software es una disciplina basada en la ingeniería, en la tecnología y en la administración, dedicada a la producción sistemática de productos de software. Por tal motivo, su meta es el desarrollo costeable de sistemas de software confiable que funcionen de modo eficiente y comprende todos los aspectos de la producción del software.

Dentro de la historia de la Ingeniería del software, se puede observar que los progresos más significativos se han obtenido gracias

a la aplicación de uno de los principios fundamentales a la hora de resolver cualquier problema, la descomposición de un sistema complejo en partes que sean más fáciles de manejar.

En los primeros estudios de desarrollo de los lenguajes de programación, se tenía un código en el que no había separación de conceptos, datos y funcionalidad que se mezclaban sin una línea divisoria clara.

En la siguiente etapa aparecen las definiciones de Descomposición Funcional, orientado a identificar las partes más importantes y manejables como funciones que se definen como requerimientos de un sistema, aun así surgen grandes problemas ya que las funciones quedan, en ocasiones, poco claras debido a la utilización de datos compartidos, y los datos están esparcidos por todo el código, provocando que si un dato se modifica, éste se tendrá que modificar en varias partes del código. A pesar de la problemática existente aparecen lenguajes de programación que permiten la codificación de este estilo.

Continuando con la evolución, surge la programación orientada a objetos (POO), en 1960 apareció la primera implementación usable de los conceptos de orientación a objetos con el lenguaje Simula-68, que supuso un gran paso en la ingeniería del software, ya que presentaba un modelo de objetos que parecía encajar de manera adecuada con los problemas reales. La cuestión era saber descomponer de la mejor manera el dominio del problema al que se enfrentaría, encapsulando cada concepto en lo que se dio en llamar objetos y haciéndoles interactuar entre ellos, habiéndoles dotado de una serie de propiedades.

En 1980, veinte años después, se empezó a usar de manera comercial la POO en proyectos reales. Fue el crecimiento de C++ frente a C y COBOL Surgieron numerosas metodologías para ayudar en tal proceso de descomposición y aparecieron herramientas que incluso automatizaban parte del proceso. Esto no ha cambiado y se sigue haciendo en el proceso de desarrollo del software.

Sin embargo, frecuentemente la relación entre la complejidad de la solución y el problema resuelto hace pensar en la necesidad de un nuevo cambio. Por lo antes mencionado surge el término Programación Orientada a Aspectos (POA). El concepto de POA fue introducido por Gregor Kiczales y su grupo, aunque el equipo Demeter, según [2] había estado utilizando ideas Orientadas a Aspectos antes incluso de que se acuñara el término. Éste mismo autor menciona que el trabajo del grupo Demeter estaba centrado en la Programación Adaptiva (PA), que puede verse como una instancia temprana de la POA. Dicha metodología de programación se introdujo alrededor de 1991. Consiste en dividir los programas en varios bloques de cortes. Inicialmente, se separaban la representación de los objetos del sistema de cortes; luego se añadieron comportamientos de estructuras y estructuras de clases como bloques constructores de cortes, además estaba basada en el uso de autómatas finitos y una teoría formal de lenguaje para expresar concisamente y procesar eficientemente conjuntos de caminos en un grafo arquitectónico, como por ejemplo los diagramas de clase en UML.

La relación entre la POA y la PA surge de la Ley de Demeter, según Larman Craig (1999):

"Solo conversa con tus amigos inmediatos".

Esta ley inventada en 1987 en la Northeastern University y popularizada en libros de Booch, Budd, Coleman, Larman, Page-Jones, Rumbaugh, entre otros, es una simple regla

de estilo en el diseño de sistemas orientados a objetos.

Para poder escribir código respetando la Ley de Demeter observaron que los conceptos que se entrecruzan entre varias clases deberían y tendrían que ser claramente encapsulados. Esto resultaría en una clara separación de los conceptos de comportamiento y de aquellos conceptos de la funcionalidad básica. Por lo tanto, la separación completa de conceptos fue área de interés de este grupo aún antes de que la POA existiera como tal.

En [4] se menciona que en 1995 dos miembros de este grupo, Cristina Lopes, actualmente integrante del grupo Xerox PARC, y Walter Huersch, presentaron un reporte técnico sobre separación de conceptos incluyendo varias técnicas como filtros composicionales y PA para tratar con los conceptos que se entrecruzan. Este reporte identificó el tema general de separación de conceptos y su implementación, y lo propuso como uno de los problemas a resolver más importante en el diseño y desarrollo de software.

La definición formal de PA puede considerarse como una definición inicial y general de la POA: en PA los programas se descomponen en varios bloques constructores de corte (crosscutting building blocks). Inicialmente separaron la representación de los objetos como un bloque constructor por separado. Luego agregaron comportamiento de estructura (structure-shy behavior) y estructuras de clase como bloques constructores de corte.

La primera definición del concepto de aspecto fue publicada en 1995 según [4], también por el grupo Demeter, y se describía de la siguiente forma:

"Un aspecto es una unidad que se define en términos de información parcial de otras unidades".

La definición de aspectos ha evolucionado desde entonces, ha cambiado con el tiempo y se hace más comprensible y precisa. Actualmente es más apropiado hablar de la siguiente definición de Gregor Kiczales:

> Un aspecto es una unidad modular que se dispersa por la estructura de otras unidades funcionales. Los aspectos existen tanto en la etapa de diseño como en la de implementación. Un aspecto de diseño es una unidad modular del diseño que se entremezcla en la estructura de otras partes del diseño. Un aspecto de programa o de código es una unidad modular del programa que aparece en otras unidades modulares del programa. [3]

Cristina Lopes y Karl Lieberherr empezaron a trabajar con Gregor Kickzales y su grupo. A Lopes y Kickzales no les gustó el nombre

"Programación Adaptativa" e introdujeron un mejor término "Programación Orientada a Aspectos (POA) o Aspect Oriented

Programming en su denominación inglesa." con su propia definición y terminología. Los conceptos de orientación a aspectos nacieron en 1996.

En la literatura se le considera a Gregor Kickzales como el creador de este nuevo paradigma.

A partir de entonces, la POA y otras técnicas y tecnologías, que se centraban en la modularización del código que atraviesa varios componentes para realizar una cierta función (crosscutting code), se agruparon bajo el nombre de Advanced Separation of Concerns. Después de varios años, en 2002, se adoptó la denominación actual Desarrollo de

Sistemas Orientado a Aspectos para referirse a las técnicas y tecnologías que pretenden la modularización de los crosscutting concerns a lo largo del ciclo de vida. Desde entonces, los investigadores que trabajaban en torno a la orientación a aspectos comenzaron a estudiar las relaciones entre aspectos así como las propiedades de la composición aspectual.

La identificación, especificación e implementación de los aspectos se realizaba en la fase de implementación, por tal motivo, los programadores tienen la necesidad de rediseñar el sistema para evitar el código entrelazado (crosscutting code).

3.2. Trabajo de investigación

El tema del trabajo de investigación se denomina:

"Diseño de artefactos para el modelo de aplicaciones bajo el Paradigma Orientado a Aspectos"

3.3. Desarrollo de capítulos

Para la elaboración de la presente investigación se ha estructurado de la siguiente manera:

- Capítulo I. Marco teórico e hipótesis.
- Capítulo II. Ingeniería de Software Orientada a Aspectos. Capítulo IV. Diseño de artefactos para las competencias transversales.
- Capítulo V. Caso de Estudio Conclusiones y trabajos futuros Referencias.
- Anexos

4. Conclusiones

Contar con un buen diseño dentro del ciclo de vida del desarrollo del software permite el logro de los objetivos y disminuye los riesgos que llevarían al fracaso al desarrollo de un buen software o en su caso la entrega de software que no cumple con los estándares mínimos de calidad.

En general, la utilización de aspectos en el proceso de desarrollo proporciona un soporte avanzado para la separación de conceptos introduciendo una nueva unidad modular: Aspecto.

La separación de aspectos es una herramienta de ingeniería del software que reduce la complejidad de las aplicaciones a niveles manejables para las personas y permite a los desarrolladores centrarse en problemas concretos, ignorando otros que en determinado momento no sean tan importantes.

Es por eso que el presente trabajo pretende diseñar artefactos que permitan al Ingeniero de software (Diseñador) crear aspectos que faciliten la identificación de competencias transversales dentro del ciclo de vida del desarrollo de un software.

5. Referencias

[1] I. Sommerville, *Ingeniería de Software*, Addison-Wesley, México. 2011.

[2] A. Colyer and A. Clement, A, *Aspect-oriented programming with AspectJ*, IBM System J. 2005.

[3] B. Bruegge y A. Dutoit Allen, *Ingeniería de software orientada a objetos*, Prentice-Hall. 2002.

[4] K. J. Lieberherr. *Adaptive Object-Oriented Software*, The Demeter Method, Northeastern University Boston. 1996.

[5] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J. Loingtier and J. Irwin, *Aspect-Oriented Programming*, Xerox Palo Alto Research Center, 1997

[6] C. Lopes, *A Language Framework for Distributed Programming*, Ph.D.thesis, Northeastern University, noviembre 1997.

[7] A. F., Contreras B, *Programación Orientada a Aspectos, Análisis del paradigma*, Tesis (2002), Universidad Nacional del Sur

[8] J. M. Nieto Moreno, "Introducción a la Programación Orientada a Aspectos", Artículo, Escuela Superior de Ingeniería Informática

[9] O. Hernández, J. Octavio y K. Cortés, "Documentando arquitecturas orientadas a aspectos para líneas de productos de software", Artículo, Facultad Estadística e Informática, Universidad Vreacruzana, (CIMAT)

[10] A. Navasa Martínez, *Marco de trabajo para el desarrollo de arquitecturas software orientadas a aspectos*. Tesis doctoral (2008), Departamento de Ingeniería de sistemas informáticos y telemáticos, Universidad de Extremadura

[11] Página del grupo Demeter: http://www.ccs.neu.edu/research/demeter/