MÉTRICAS APLICADAS A LA PROGRAMACIÓN ORIENTADA A ASPECTOS

(Workshop de Investigadores en Ciencias de la Computación WICC-2006)

Lorena S. Baigorria, Germán Montejano
Departamento de Informática, Universidad Nacional de San Luis
San Luis- CP 5700- Argentina

RESUMEN

Encontramos muchos problemas de programación para los cuales las técnicas de programación procedural o orientadas a objetos (OO) no son suficientes para capturar algunas de las decisiones importantes de diseño, que el sistema debe implementar.
La Programación Orientada a Aspectos (POA) es una técnica que ha surgido para mejorar la separación de las competencias en la programación de software.
La Orientación a Aspectos (OA) se basa en tecnologías existentes, como la orientación a objetos.
Existe gran cantidad de investigaciones acerca de las métricas de software orientadas a objetos y procedurales, pero no para software orientado a aspectos.
Las métricas de software son formas de calificar los diseños de software. Decimos entonces que las métricas aplicadas a la OA son cruciales para determinar la efectividad de este paradigma como también de su uso en el diseño de sistemas de software.
Dado que como dijimos la OA es una extensión de OO podemos ver como la aplicación de esta metodología afecta las métricas utilizadas para OO y cuales métricas deberían considerarse para realizar un análisis mas objetivo del diseño de sistemas de software que apliquen este nuevo paradigma.

Keywords: Métricas, Orientación a Aspectos, Métricas C&K, Métricas Orientadas a Aspectos.

1.INTRODUCCIÓN

Las métricas de software intentan medir la complejidad del software para predecir el costo total del proyecto y la evaluación de la calidad y la efectividad del diseño. Las métricas de software tienen múltiples aplicaciones en las distintas tareas de la ingeniería de software tales como pruebas, refactoring, gerenciamiento y mantenimiento. [1]
La investigación de las métricas se va adaptando a las necesidades y nuevas propuestas de desarrollo de software, nuevos lenguajes y nuevos paradigmas de programación, como es el caso de la POA.
La *programación orientada a aspectos* (*POA*) es una nueva metodología de programación que aspira a soportar la separación de competencias para los aspectos tanto en el diseño como en la implementación de software.. Los lenguajes orientados a aspectos se utilizan para lograr esta separación y así obtener las ventajas proporcionadas por esta metodología como: el fácil desarrollo y mantenimiento del software
El software desarrollado con esta metodología es más fácil de desarrollar y mantener, debido a la separación de competencias entre las componentes y los aspectos del software.
Al aplicar métricas al diseño de sistemas OA podemos obtener una visión objetiva de la calidad del diseño.

2.ORIENTACION A ASPECTOS

Los aspectos son propiedades de un software que tienden a atravesar sus funcionalidades principales. Consideramos un *aspecto a una unidad modular que se disemina por la estructura de otras unidades funcionales. Los aspectos existen*

tanto en la etapa de diseño como en la de implementación[2].
La sincronización, el compartimiento de recursos y distribución son algunos ejemplos de aspectos.. Normalmente los aspectos están entrelazados en el corazón de los componentes del sistema causando el problema de tener código desordenado.
La Orientación a Aspectos trata de modularizar este entrecruzamiento de competencias y encapsularlos en módulos en vez de tenerlos dispersos en los componentes del sistema.

Para lograr programas OA utilizamos los lenguajes OA. Los lenguajes orientados a aspectos definen una nueva unidad de programación de software para encapsular las funcionalidades que cruzan todo el código. Además, estos lenguajes deben soportar la separación de aspectos como la sincronización, la distribución, el manejo de errores, la optimización de memoria, la gestión de seguridad, la persistencia. De todas formas, estos conceptos no son totalmente independientes, y está claro que hay una relación entre los componentes y los aspectos, y que por lo tanto, el código de los componentes y de estas nuevas unidades de programación tienen que interactuar de alguna manera. Para que ambos (aspectos y componentes) se puedan mezclar, deben tener algunos puntos comunes, que son los que se conocen como *puntos de enlace,* y debe haber algún modo de mezclarlos.
El encargado de realizar este proceso de mezcla se conoce como *tejedor*. El tejedor se encarga de mezclar los diferentes mecanismos de abstracción y composición que aparecen en los lenguajes de aspectos y componentes ayudándose de los puntos de enlace.[2]

3. EFECTOS DE LA POA EN LAS METRICAS DE CHIDAMBER Y KEMERER METRICS (C&K)

Como dijimos la POA puede verse como una extensión de los sistemas orientados a objetos. Pero en un sistema orientado a aspecto la unidad básica es el aspecto en vez de una clase o procedimiento.
Las métricas utilizadas para la evaluación de sistemas OO son aplicables a los sistemas OA, como es el caso de las métricas C&K definidas por Chidamber y Kemerer [3].
Veremos como estas métricas se ven afectadas al aplicarse a sistemas Orientados a Aspectos.

3.1. PESO DE LOS MÉTODOS POR CLASE (WMC)

Los aspectos pueden ayudar a reducir el numero de métodos por clase de la siguiente manera:

- Los aspectos combinan el entrecruzamiento de funcionalidades en unidades modulares y encapsuladas. Sin el diseño orientado a aspectos, entrecruzamiento de funcionalidades estaría desparramado en la clase.
- En algunos casos las subclases deben re definir una función de la clase padre de manera tal que pueda definir el comportamiento de su aspecto.

3.2. PROFUNDIDAD DEL ÁRBOL DE HERENCIA (DIT)

Las subclases pueden definirse con el propósito de aplicar su propia implementación del comportamiento del aspecto, éstas no existirían en sistemas desarrollados con el paradigma de OA. Esto ayuda a reducir la profundidad del árbol de herencia de una clase.

3.3. NUMERO DE HIJOS (NOC)

En este caso nos encontramos con la misma justificación que en el caso anterior. Ya que solo las subclases especializadas serían definidas, es decir las subclases que

modelaran el comportamiento de los componentes del sistema.

3.4. ACOPLAMIENTO ENTRE OBJETOS (CBO)

La presencia de aspectos decrementa el acoplamiento entre clases, aunque aumenta el acoplamiento entre clases y aspectos.
Esto se debe a que los aspectos son nuevas entidades de las cuales las clases dependen.
Dado que un diseño puede involucrar acoplamiento entre clases, seria mejor que este se diera entre clases y aspectos.

3.5 RESPUESTA POR CLASE (RFC)

RFC incrementa ante la presencia de aspectos. Esto se debe a que el numero de entidades con las que se comunica una clase se incrementa, y las clases tienen que comunicarse con los aspectos. El punto a favor en este caso es que los aspectos pueden diseñarse de manera tal que encapsule la lógica y los objetos con los cuales la clase se comunica en forma modular.

4. METRICAS APLICADAS A LA ORIENTACION A ASPECTOS

Como se dijo en las secciones anteriores la unidad básica de POA son los aspectos por esto surge de forma natural la medición los mismos. Lo que se mostrará en esta sección es la medición de los aspectos intentando encontrar métricas que nos ayuden a conocer la complejidad del sistema que se está desarrollando aplicando esta metodología.
Al interiorizarse en la POA [4][5] encontramos diferentes conceptos relacionados a ésta. Los cuales nos indican las posibles métricas necesarias desde el punto de vista del diseño, para poder obtener una apreciación mas objetiva del diseño del sistema realizado. Se propone entonces, trabajar con métricas desde el diseño del sistema aplicándolo a UML, para realizar esto; es decir, se trabajara con las extensiones realizadas a UML para soportar OA [6][7]. Luego se aplicaran las métricas a esas extensiones. Algunas de las métricas sobre las cuales se trabaja son:

- **Relación entre Clase y Aspectos:** una clase es una componente cuya funcionalidad puede ser encapsulada. Como la POA separa las clases de los aspectos un aspecto puede relacionarse con una o varias clases. Esta relación es la que muestra que una clase se ve afectada por dicho aspecto.

- **Clase Tejida**: Al utilizar un tejedor de aspectos, el código de las clases y los aspectos se mezclan y se genera como resultado una clase tejida. La estructura de la clase tejida depende del tejedor de aspectos y del lenguaje de programación que se haya utilizado para generarla.

- **Puntos de enlace**: son una clase especial de interfaz entre los aspectos y los módulos del lenguaje de componentes. Son los lugares del código en los que éste se puede aumentar con comportamientos adicionales. Estos comportamientos se especifican en los aspectos.

- **Clases:** unidad básica que encapsula el comportamiento del sistema a través de las distintas funcionalidades y no contiene el comportamiento de los aspectos que la afectan.

Además de los aspectos, estos conceptos afectan el diseño del sistema por lo que sería deseable conocer algunas mediciones de los mismos como:

- **Cantidad de Aspectos**: al ser el aspecto la unidad básica de la POA

es deseable conocer la cantidad de aspectos que afectan al sistema total o a parte del mismo. Esto nos puede ayudar a justificar el uso de esta metodología ya que si la cantidad de aspectos es muy alta; esta metodología nos beneficia; en cambio si la cantidad es baja nos puede complicar el diseño en vez de mejorarlo.

- **Cantidad de relaciones existentes entre aspectos y una clase**: una clase puede ser atravesada por mas de un aspecto, aumentando así la complejidad de la clase.

- **Cantidad de clases relacionadas con un mismo aspecto**: dado que varias clases se pueden relacionar con un mismo aspecto, es importante conocer esta cantidad para conocer así la complejidad del sistema. Ya que el diseño del comportamiento de aspectos puede verse afectado por esto.

- **Cantidad de puntos de enlace en una clase:** al ser los puntos de enlace los lugares en los que se agrega el comportamiento de los aspectos, esta cantidad varia de la cantidad de aspectos que atraviesan una clase ya que dicha clase puede ser atravesada por un mismo aspecto en diferentes oportunidades.

- **Cantidad de Clases Tejidas:** la generación de la clase tejida depende del lenguaje OA que se utilice. En el caso que se generen clases tejidas el número de éstas me indica la complejidad y reutilización de la clase que intervino para generar la misma.

- **Reutilización de una clase**: sabemos que una clase se puede relacionar con mas de un aspecto, esto causa un incremento de la misma, por lo que disminuye su reutilización.

También podemos encontrar algunas relaciones entre las medidas anteriores como:

- **La cantidad de Puntos de enlace con la reutilización de la clase:** cuanto mas es afectada una clase por un aspecto menor es la reutilización de la clase.

- **Cantidad de relaciones existentes entre un aspecto y una clase con la Reutilización de la misma**: mayor es la cantidad de aspectos que atraviesan una clase menor será la reutilización de la misma.

- **Cantidad de Clases Tejidas con la cantidad de aspectos**: ya que la clase tejida es generada a partir de una clase y un aspecto puede ser que una misma clase tejida contenga el comportamiento de uno o más aspectos dependiendo de las decisiones de diseño.

5 CONCLUSIONES

La POA es una nueva metodología para el desarrollo de sistemas de software la cual permite un manejo más fácil del seguimiento y mantenimiento del software desarrollado. Como toda metodología debe ser evaluada a través de distintas herramientas para poder así calificar el uso de la misma. Una de las herramientas que permiten medir de manera mas objetiva el diseño realizado mediante la aplicación de distintas técnicas, en particular de los sistemas OA, son las métricas. Esto nos lleva a profundizar el análisis de las métricas para poder aplicarlas a la POA y encontrar además de los posibles efectos, nuevas métricas que ayuden a calificar objetivamente un sistema OA.

En este trabajo se muestra tanto las consecuencias de la aplicación de la POA en el desarrollo de sistemas como las posibles métricas que deberían considerarse para medir de forma mas objetiva el diseño realizado aplicando esta metodología.
la mayor parte del diseño Orientado a Aspectos es subjetivo por lo que la definición formal de estas métricas obteniendo una vista mas objetiva de la calidad del diseño. En futuros trabajos se realizara la definición formal de métricas para sistemas Orientados a Aspectos. Definiendo métricas aplicables a la unidad básica de la OA, los aspectos; así como para diferentes artefactos que intervienen en la POA.

7 REFERENCIAS

[1] Zhao J.;Measuring Coupling in Aspect-Oriented Systems; Department of Computer Science and Engineering; Fukuoka Institute of Technology

[2] Quintero A.; Visión General de la Programación Orientada a Aspectos. Departamento de Lenguajes y Sistemas Informáticos. Universidad de Sevilla.

6 FUTUROS TRABAJOS

Sabemos que las métricas nos permiten calificar el diseño de software, pero aun así

[3] Zakaria; Hosny; Metrics for Aspect-Oriented Software Design. The American University in Cairo.

[4] Nieto Moreno; Introducción a la Programación Orientada a Aspectos, Escuela Técnica Superior de Ingeniería Informática, Universidad de Sevilla, España

[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] J. Suzuki, Y. Yamamoto. Extending UML with Aspect: Aspect Support in the Design Phase. 3er Aspect-Oriented Programming (AOP) Workshop at ECOOP'99.

[7] OMG. Unified Modeling Language Specification. v. 1.3, 1ª Edición. Marzo, 2000.http://www.omg.org/technology/documents/formal/unified_modeling_language.htm