Énfasis: Un Lenguaje Orientado a Aspectos con Corte sobre Variables Locales

Ulises Juárez Martínez, José Oscar Olmedo Aguirre
Departamento de Ingeniería Eléctrica, CINVESTAV-IPN, México D. F.
Teléfono (01)555 5061-3758 Fax (01)555 5061-3757
E-mail: ujuarez@computacion.cs.cinvestav.mx, oolmedo@cs.cinvestav.mx

Abstract

La programación orientada a aspectos (POA) ofrece mecanismos para encapsular requisitos no funcionales (aspectos) los cuales se coordinan con la aplicación mediante tres conceptos fundamentales: puntos de unión (join points), corte sobre puntos (pointcuts) y avisos (advices). Los lenguajes de aspectos ofrecen corte sobre puntos basados en clases, objetos, métodos y campos. En áreas como depuración, verificación de programas, visualización de programas y monitoreo, el corte sobre puntos necesita extenderse a variables locales para permitir flexibilidad en la definición de aspectos de grano fino. Énfasis es un lenguaje de aspectos específico para corte sobre puntos de variables locales que incluye un compilador y un entorno de ejecución dinámico.

Keywords — Programación orientada a aspectos, corte sobre puntos, variables locales, entrelazado dinámico.

1. Motivación

La corrección de errores y depuración de programas son actividades que se realizan para garantizar un desarrollo de software con calidad. La simplificación de algoritmos, el incremento de eficiencia y la velocidad de programas, la supervisión de valores en entornos de graficación y aprendizaje, optimización de código, así como el monitoreo de variables críticas a tiempo de ejecución son solo algunos ejemplos donde la utilización de variables locales juega un papel importante.

Algunas técnicas para reducir el impacto de errores en el software son el manejo de aseveraciones (assertions) para definir precondiciones y postcondiciones, la definición y uso de excepciones generales, hasta técnicas formales mediante evaluaciones parciales. Las herramientas de depuración son una opción para subsanar errores, sin embargo, no son una herramienta que permita fácilmente incorporarse como modelo de programación ni como software de programación para encapsular requisitos funcionales durante todo el ciclo de desarrollo del software. La POA ofrece los mecanismos necesarios para establecer cortes sobre variables locales e implementar atributos de grano fino en forma de aspectos.

Los lenguajes orientados a aspectos no ofrecen soporte para definir puntos de corte sobre variables locales. El corte más fino que definirse es para campos o variables (de instancia o de clase). Nuestro trabajo propone a Énfasis, un lenguaje orientado a aspectos con un modelo de puntos de unión específico para definir corte sobre puntos de variables locales. Énfasis está basado en el modelo de puntos de unión de AspectJ.

2. Trabajos previos en el área

En un lenguaje orientado a aspectos, la expresividad del modelo de puntos de unión define las capacidades de corte que pueden definirse y al mismo tiempo cuánto puede extenderse el comportamiento original del programa [1]. Estas extensiones están directamente asociadas al proceso de entrelazado (weaving), el cual es responsable de coordinar las interacciones entre aspectos y componentes (objetos o procedimientos). El proceso de entrelazado puede realizarse desde el código fuente (estático) donde predominan AspectJ, Eos [10] y CaesarJ [11]. El entrelazado estático aplica también en tiempo de carga de clases donde se destaca nuevamente AspectJ. A tiempo de ejecución, el entrelazado se vuelve dinámico y el soporte se realiza principalmente con monitoreo de eventos como PROSE [3] [8] y hasta con máquinas virtuales [4].

Concretamente, AspectJ es el lenguaje con el modelo de puntos de unión más expresivo y utiliza los tres modelos de entrelazado gracias a su fusión con AspectWerkz [6], PROSE utiliza una interfaz de depuración (JVMDI) para soportar corte sobre puntos de métodos, campos y excepciones, similarmente Axon incorpora conceptos de reglas ECA [9] y Eos–U considera el mecanismo de avisos como análogo a las redefiniciones de métodos [2].

Aunque todos estos lenguajes manejan diferente expresividad en sus modelos de puntos de unión, ninguno apoya el corte sobre variables locales.

3. Objetivos de la investigación

Los objetivos de la investigación alcanzados son:

- Definir un modelo de puntos de unión que permita definir corte sobre puntos de variables locales.
- Integrar un enfoque de entrelazado dinámico para trabajar con esquemas de monitoreo.
- Definir la expresividad que debe tener dicho modelo de puntos de unión.

- ¿Qué primitivas de corte son importantes definir para trabajar con variables locales?

Los objetivos restantes son:

- Desarrollar un compilador para soportar el lenguaje de aspectos para variables locales.
- Incorporar el entorno dinámico de Énfasis para soportar el entrelazado dinámico para variables locales.

4. Metodología

La definición del modelo de puntos de unión se basó en el modelo de AspectJ. Particularmente, el tipo de expresividad para trabajar con campos se utilizó para la parte de variables locales.

Para la especificación de los patrones de firmas sobre variables locales se consideró la ocurrencia de las variables en diferentes contextos como las estructuras de control y ambientes definidos por paréntesis arbitrarios.

El diseño del entorno dinámico de Énfasis se basó en un esquema de monitoreo de eventos mediante una interfaz de depuración mejorada, la cual está disponible en la versión 5.o de Java: JVM Tool Interface (JVMTI) [7]. Funciones callback y breakpoints controlan y ofrecen información del entorno de ejecución de las variables locales.

5. Estado de la investigación

El avance de la investigación se considera en un 75%, el examen predoctoral se presentó en el mes de abril del presente año, el resultado del examen fue satisfactorio y las recomendaciones se enfocan principalmente en dos rubros: la justificación de corte sobre variables locales y la publicación en congresos especializados.

6. Conclusiones intermedias

El avance de la investigación ha permitido desarrollar el entorno dinámico de ejecución para ofrecer información no solo de variables locales, sino además de puntos no alcanzabas desde el nivel de código fuente, como lo es el inicio y término de hilos previos y posteriores a la ejecución de la aplicación, inicio y término del colector de basura, ejecución de métodos nativos y la especificación de avisos de tipo *before* y *after* para estos casos particulares.

El modelo de puntos de unión de Énfasis soporta la especificación de tres primitivas: *haslocal* para preguntar por la existencia de una variable, *getlocal* para avisar cuando se lee una variable local y *setlocal* para avisar cuando se escribe o cambia de valor una variable local. En este caso, se tiene soporte completo a los tres tipos de avisos: *before*, *around* y *after*.

En el caso de la especificación de puntos de unión mediante patrones de firmas, se pueden identificar tanto los argumentos de un método como las variables locales explícitamente declaradas en dicho método.

El trabajo pendiente consiste en la construcción del compilador. En este sentido se está utilizando Polyglot, un compilador extensible para Java y las extensiones para soportar variables locales se basan en Aspect–Bench Compiler.

7. Referencias

[1]. G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, W. G. Griswold. “An Overview of AspectJ”. *Proceedings of European Conference on Object–Oriented Promgramming*. Lecture Notes in Computer Science 2072, pp. 327-353. Springer-Verlag. 2001.

[2]. H. Rajan, K. J. Sullivan. “Classpects: Unifying Aspect– and Object–Oriented Lenguaje Design”. *27th International Conference on Software Eengineering*. ACM. 2005. In press.

[3]. A. Popovici, T. Gross, G. Alonso; “Dynamic Weaving for Aspect–Oriented Programming”. In: *1st International Conference on Aspect-Oriented Software Development*, Enschede, The Netherlands, April 2002.

[4]. C. Bockisch, M. Haupt, M. Mezini, K. Ostermann. “Virtual Machine Support for Dynamic Join Points”. In: *3th Aspect-Oriented Software Development Conference*. 2004.

[5]. H. Rajan, K. Sullivan. “Need for Instance Level Aspect Language with Rich Pointcut Language”. In: Proceedings of the Workshop on Software Engineering Properties of Languages for Aspect Technologies (SPLAT) held in conjunction with AOSD. 2003.

[6]. AzpectWerkz Home Page http://aspectwerkz.codehaus.org/.

[7]. JVMTI: http://java.sun.com/j2se/1.5.0/docs/guide/jvmti/jvmti.html

[8]. A Nicoara, G. Alonso. “Dynamic AOP with PROSE”. In: Proc. of *International Workshop on Adaptive and Self-Managing Enterprise Applications (ASMEA'05)* in conjunction with CAISE'05, Porto, Portugal, June 2005. In press.

[9]. S. Aissman, M. Haupt. Axon – “Dynamic AOP through Runtime Inspection and Monitoring”. *Advancing the State-of-the-Art in Run-Time Inspection (ASARTI) Workshop*. 2003.

[10]. Rajan, H. and Sullivan, K., “Eos: Instance-Level Aspects for Integrated System Design”, Proceedings of the *9th European software engineering conference* held jointly with 11th ACM SIGSOFT international symposium on Foundations of software engineering (ESEC/FSE 03), Helsinki, Finland, Sep 2003, pp291-306.

[11]. Mezini, M., and Ostermann, K., “Conquering Aspects with Caesar”, Proceedings of the *2nd International Conference on Aspect–Oriented Software Development (AOSD 03)*, Mar 2003, Boston, MA, USA, pp. 90-100.