PTOLEMY: APLICACIÓN DEL DBC EN LA POA

*Guadalupe Isaura Trujillo Tzanahua**
*Ulises Juárez Martínez**
*Beatriz A. Olivares Zepahua**
*Ignacio López Martínez**
*Celia Romero Torres**
** Instituto Tecnológico de Orizaba*

Resumen
La Programación Orientada a Aspectos (POA) es un paradigma de programación que permite separar componentes y aspectos permitiendo su abstracción y composición para producir todo el sistema. Las principales ventajas que ofrece son: separación de asuntos, facilidad para razonar conceptos, alto nivel de reutilización y mejor mantenimiento del software.
Actualmente, existen varios Lenguajes Orientados a Aspectos entre los que destacan: AspectJ, CaesarJ, ECaesarJ, EOS, y Ptolemy. Cada uno proporciona mecanismos que contribuyen al desarrollo o mantenimiento del software dependiendo de lo que se desee realizar.
El Diseño por Contrato (DbC) es una técnica muy conocida para el diseño de software de calidad. Un contrato es la especificación de cómo los elementos de un sistema colaboran entre sí, con base a obligaciones (precondiciones) y beneficios (postcondiciones). Los beneficios más importantes del DbC son: facilitad de pruebas unitarias, manejo de excepciones de manera eficiente y construcción de componentes robustos.
Las principales herramientas que implementan el DbC en Java son: jContractor, Jass, Modern Jass, Contract4j, JML y Ptolemy. Sin embargo, no se reporta alguna guía para el desarrollo de software con Ptolemy. El propósito de este artículo es ofrecer una guía básica para el desarrollo de software orientado a aspectos utilizando Ptolemy, lenguaje de programación orientado a aspectos que combina el DbC y la programación basada en eventos mediante un mecanismo denominado contrato traslúcido.

Diseño por Contrato (DbC)

El DbC es una técnica diseñada por Bertrand Meyer y característica central del lenguaje de programación Eiffel, que él desarrolló [1]. El DbC se utiliza con cualquier lenguaje de programación para desarrollar software correcto.
Un contrato es una especificación que establece las condiciones de uso e implementación (obligaciones y beneficios) que clientes y proveedores de un componente deben cumplir.
Una aseveración es una condición que debe cumplirse. Su incumplimiento invalida totalmente el software. En el DbC se utilizan tres tipos de aseveraciones [1]:

- **Precondición:** restricciones bajo las cuales una rutina funcionará correctamente.
- **Postcondición:** describe el efecto de la rutina.
- **Invariante de clase:** restricciones que deben satisfacerse por cada objeto tras la ejecución de los métodos y constructores.

Ptolemy

Ptolemy es un lenguaje de programación cuyo objetivo es mejorar la habilidad de un ingeniero de software de separar los asuntos conceptuales [2]. Ptolemy se distingue de Lenguajes Orientados a Objetos como Java debido que ofrece el soporte para el manejo de eventos e introduce un nuevo concepto denominado contrato traslúcido [3]. A diferencia de otros Lenguajes Orientados a Aspectos como AspectJ y CaesarJ solventa dificultades como:

- **Cuantificación (quantification):** referenciar los lugares en el código donde un evento es disparado utilizando su propio identificador. Esto se hace sin tener que nombrar a las clases que disparan el evento [4].
- **Inconsciencia (obliviousness):** liberar al programador de las especificaciones puntuales donde una sentencia o conjunto de sentencias del programa tendrá efecto.
- **Fragilidad en los cortes:** dependencia de los cortes respecto a cómo se estructura el código sobre el que actuarán los aspectos.

Primeros pasos

Para comenzar, lo primero que se necesita es descargar el compilador de Ptolemy de los siguientes sitios:

- http://ptolemy.cs.iastate.edu/download/
- http://sourceforge.net/projects/ptolemyj

Recomendaciones

La carpeta *bin* de Ptolemy contiene el script para utilizar el compilador de Ptolemy, sin embargo el archivo *bat* requiere de ciertos ajustes. Los errores detectados son:

- Sintaxis incompleta al asignar un valor a la variable de entorno PYC_HOME. El valor asignado debe situarse entre *"%"* como: *"set PYC_HOME=%~dp0%\..".*
- Ruta incorrecta de *javac.jar.* El jar de javac se localiza dentro del subdirectorio *\bootstrap\lib\javac.jar.*

Configuración de variables de entorno

Antes de utilizar pyc para compilar y ejecutar programas, es necesario configurar las variables de entorno del sistema *Path* y *PYC_HOME*. La variable de entorno *Path* tendrá como valor el directorio de la carpeta *pyc\bin y PYC_HOME* tendrá como valor el directorio que contiene el compilador *pyc*. Compilación y ejecución

Ptolemy añade las características específicas compatibles con los programas Java. Cada programa válido en Java es un programa válido en Ptolemy. El compilador Ptolemy se encarga de traducir el archivo fuente a bytecode para su ejecución en la máquina virtual de Java. La compilación de un archivo de código fuente *.java* se realiza a través del comando *pyc*. Tras ejecutar este comando, se generarán los archivos *.class* de cada clase existente en el código fuente. En caso de que existan errores sintácticos en el código fuente, el compilador informa de ello y por lo tanto el archivo *.class* no se genera.

Para ejecutar un programa en Ptolemy, se utiliza el comando java seguido del nombre de la clase que contiene el método *main()* .

Habilitar contratos traslúcidos durante la ejecución

Para verificar el cumplimiento de un contrato traslúcido es necesario habilitar las aseveraciones con el modificador *-ea* seguido de la clase principal.

Contrato traslúcido

Un contrato traslúcido para un tipo de evento es un algoritmo abstracto que describe el comportamiento de los aspectos que se aplican a una interface orientada a aspectos [2]. El algoritmo es abstracto porque suprime muchos detalles de la implementación. Esto permite a la especificación decidir qué detalles ocultar y cuáles revelar.

Terminología Orientada a Aspectos

A continuación se describen los conceptos de aspectos desde la perspectiva del lenguaje Ptolemy.

Evento (Punto de unión)

Un punto de unión es un evento identificable en la ejecución de un programa. Ptolemy provee el soporte para programación basada en eventos. La declaración de un evento en Ptolemy consta de 3 elementos [2]:

- **Nombre del evento:** Se utiliza por el controlador para declarar sus intereses y por el código base para anunciar las instancias de ese evento.
- **Lista de variables de contexto:** Permite que el código base esté disponible para los manejadores de eventos.
- **Tipo de retorno:** El anuncio de los eventos puede retornar valores para los manejadores del código OO.

Para definir un evento en Ptolemy, se utiliza la palabra reservada *event*, seguido del nombre del evento como se muestra en la línea 1 del listado 1.

Método controlador del evento (Aviso)

A diferencia de AspectJ, Ptolemy no tiene una sintaxis especial para los aspectos o avisos. En su lugar, tiene la capacidad de reemplazar todos los eventos en un conjunto específico con llamadas al método controlador. El método controlador (aviso) debe ser público, tener un argumento del tipo del evento y lanzar *Throwable* como se muestra en la línea 15 del listado 1.

Cuantificación (Patrón de firmas)

La palabra reservada *when* en Ptolemy asocia un método de una clase a un conjunto de eventos identificados por un tipo de evento. El nombre de un tipo de evento permite que los programas hagan referencia a un conjunto de eventos y así cuantificar sobre un conjunto de eventos. Un beneficio importante de referirse a elementos de tal conjunto de eventos es que no requiere explícitamente la enumeración (inconsciencia). En la tabla 1 se muestra la diferencia sintáctica al cuantificar en los lenguajes AspectJ y Ptolemy.

Concepto	AspectJ	Ptolemy
Cuantificación	execution(void u01.EjemploContrato.setX(..));	when Evento do aviso;

Tabla 1. Cuantificación AspectJ vs Ptolemy.

Como se observa la identificación de puntos de unión en AspectJ se consigue mediante patrones de firmas y en Ptolemy se realiza mediante la llamada al evento a identificar con las palabras reservadas *when y do* [5]. Ptolemy permite cuantificar de manera inconsciente, al no depender del patrón de firmas, garantizando que el sistema reaccione a los eventos registrados.

Variables de contexto (Exposición de contexto)

Los cortes permiten exponer el contexto de ejecución en sus puntos de unión a través de una interfaz compuesta de parámetros formales en la declaración de los cortes definidos por el programador.
La exposición de contexto permite conocer cuáles son los argumentos del corte. Las primitivas para exponer el contexto en AspectJ son: *target, this y args.* Por otro lado, Ptolemy no provee primitivas especiales para exponer contexto, en cambio permite definir n variables de contexto de x tipos, es decir las variables disponibles para los manejadores de eventos.

Activación dinámica de aspectos *(announce)*

En CaesarJ, la activación dinámica de aspectos se realiza mediante la palabra reservada *deploy*. Ptolemy posee un mecanismo similar que realiza el anuncio al evento mediante la palabra reservada *announce*.

A continuación en el listado 1 se presenta un breve ejemplo de cómo aplicar los conceptos orientados a aspectos en Ptolemy anteriormente mencionados.

Clic aquí para ver el código

En la figura 1 se muestra la ejecución de la clase principal Entero, donde se muestra como salida el cumplimiento del contrato traslúcido al anunciar el evento *IntEvento* (linea 24) y se obtiene como salida el valor de *x* y *y*. El incumplimiento del contrato traslúcido (línea 27) genera la excepción *AssertionError.*

Figura 1. Ejecución de la clase *Entero* y su contrato traslúcido.

Conclusión

Para garantizar el desarrollo de software correcto, es decir que se cumpla con la funcionalidad expresada en su especificación (contrato) se requiere de mejores prácticas de diseño y programación como la aplicación del DbC en la POA. Al desarrollar aplicaciones en Ptolemy es posible aplicar ambos enfoques mediante los contratos traslúcidos, los cuales son implementados de acuerdo al evento identificado. Otras ventajas que proporciona Ptolemy son: mejorar la modularidad de los asuntos de corte, preservando la encapsulación del código orientado a objetos, resolver problemas como: fragilidad de los cortes y cuantificación, control de avisos y razonamiento modular.

Bibliografía

[1] M. Bertrand, Object Oriented Software Construction, Prentice Hall, 2002.
[2] R. Hridesh y T. L. Gary, «Ptolemy PROGRAMMING LANGUAGE,» 2013. [En línea]. Available: http://ptolemy.cs.iastate.edu/.
[3] R. Hridesh, M. Sean, L. Gary T., D. Robert, D. F. Rex, A. D. D. Mohammad y W. Bryan, «Modularizing Crosscutting Concerns with Ptolemy».
[4] B. Mehdi, R. Hridesh, L. Gary T. y M. Sean, «Translucid Contracts for Modular Reasoning about Aspect-oriented Programs».
[5] M. Bagherzadeh, «Enabling Expressive Aspect Oriented Modular Reasoning by Translucid Contracts».