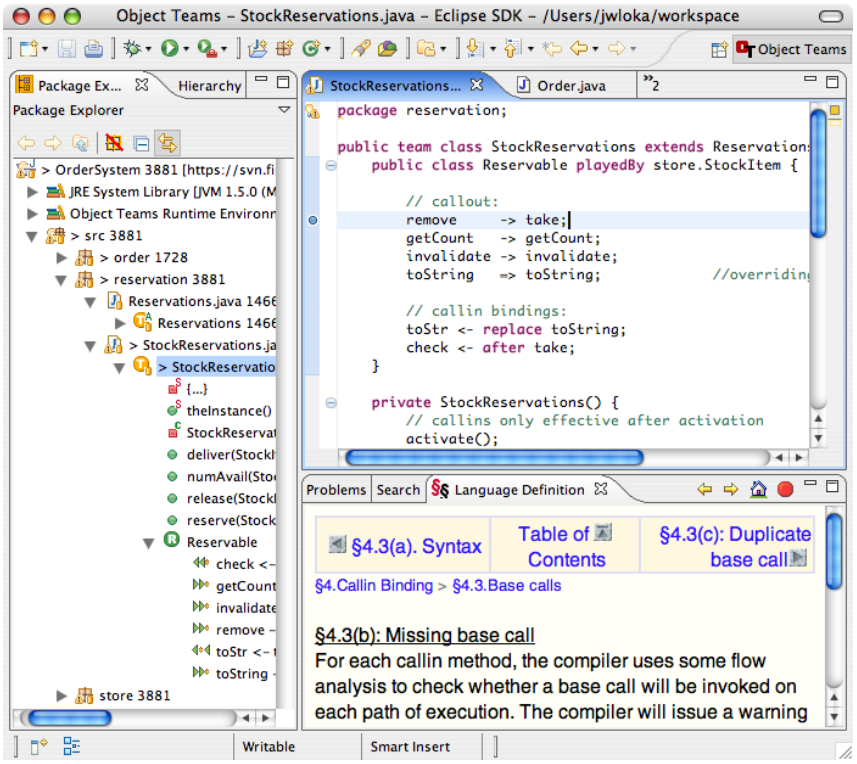
Das Object Teams Development Tooling

http://www.objectteams.org

Stephan Herrmann Christine Hundt Marco Mosconi Jan Wloka Carsten Pfeiffer
Technische Universität Berlin Fraunhofer FIRST



1 Object Teams

Seit 2001 wird an der Technischen Universität Berlin das Programmiermodell Object Teams entwickelt. Dieses Modell greift die ursprünglichen Verheißungen der Objekt-Orientierung auf, in denen eine exzellente, natürliche Modularisierung jeglicher Software versprochen wird. Die Praxis hat jedoch mittlerweile einige Ernüchterung bereitet, die darauf hindeutet, dass die Klasse als alleiniges Strukturierungsmittel für große Softwaresysteme nicht ausreicht. Die Komplexität von Entwürfen und ihren Umsetzungen legt außerdem nahe, dass es nicht ausreicht, Module nur durch zwei Arten von Beziehungen zu Systemen zusammensetzen zu können: Assoziation und klassenbasierte Vererbung.

Neue Module und Beziehungen

Wir führen deshalb zwei neue Arten von Modulen und eine Reihe neuer Beziehungsarten ein: *Teams* und *Rollen* sind besondere Arten von Klassen, wobei ein Team als Modul für eine Gruppe interagierender Rollen entworfen wird. Eine Rolle kann, außer Mitglied eines Teams zu sein, auch eine existierende (Basis-)Klasse erweitern bzw. adaptieren. Diese so genannte *playedBy* Beziehung zwischen einer Rolle und ihrer Basis erlaubt es, das Verhalten existierender Klassen zu adaptieren und eröffnet somit neue Flexibilität für den Entwurf. Teams bieten verbesserte Möglichkeiten der Kapselung, und durch Team-Vererbung können ganze Subsysteme konsistent spezialisiert werden.

Sprache und Werkzeug

Mit der Entwicklung der Programmiersprache ObjectTeams/Java (OT/J) wurden die o.g. Konzepte in die Sprache Java integriert. Eine Umfassende Definition dieser Sprache ist unter [www.objectteams.org/def] verfügbar.

Im Rahmen des vom BMBF geförderten Verbundprojektes TOPPrax [www.topprax.de] haben das Fraunhofer Institut FIRST und die Technischen Universität Berlin eine Entwicklungsumgebung für OT/J entwickelt: das Object Teams Development Tooling, das in seiner aktuellen Version auf Eclipse 3.2 und seinem Java Development Tooling (JDT) basiert. Innerhalb des TOPPrax-Projektes wurden beim Projektpartner GEBIT Solutions praktische Fallstudien durchgeführt.

Außerdem werden die Sprache und ihre Werkzeuge seit 2003 in verschiedenen Lehrveranstaltungen und Diplomarbeiten der TU Berlin eingesetzt. Die im September 2006 veröffentlichte Version 0.9.17 ist durch eine Testsuite von nahezu 27000 Testfällen getestet.

Dank der Förderung des BMBF ist es somit gelungen, ein Werkzeug zu entwickeln, das einerseits auf aktuellen Konzepten und Theorien (z.B. *aspect oriented programming, virtual classes, family polymorphism*) basiert, andererseits eine für Forschungsprojekte eher untypische Praxisreife erlangt hat.

2 Object Teams Development Tooling

Um mit einer neuen Programmiersprache effizient arbeiten zu können, reicht ein neuer Compiler allein nicht aus. Vielmehr müssen für den Alltag der Entwickler vielfältige Tätigkeiten unterstützt werden, die alle nahtlos integriert aus einer einzigen Entwicklungsumgebung heraus durchgeführt werden sollen. Für OT/J wurde die bewährte, quelloffene Entwicklungsumgebung Eclipse adaptiert und erweitert.

Sprachimplementierung

Die Sprache OT/J ist implementiert durch einen inkrementellen Compiler (basierend auf dem Compiler des JDT) und eine Laufzeitumgebung. Letztere ist dafür verantwortlich, die Adaptierungen, die eine Rolle an ihrer Basisklasse vornimmt, in diese Basisklasse hineinzuweben. Dies geschieht als Byte-code Transformation zum Zeitpunkt des Programmstartes. Beim Compiler wurde zusätzliches Augenmerk auf hilfreiche Fehlermeldungen gelegt, wobei alle Fehlermeldungen innerhalb der Umgebung direkt mit dem entsprechenden Absatz der integrierten Sprachdefinition verlinkt sind. Dies ist besonders beim Erlernen der Sprache sehr hilfreich.

Editorunterstützung

Der Editor für OT/J unterstützt nicht nur syntaktisches sondern auch semantisches highlighting. Navigation, Vervollständigung (completion), Suche sowie Übersichtsdarstellung (outline) sind alle an die neuen Strukturen angepasst. Die neuen Elemente sind durchgängig in der gesamten Umgebung durch spezielle Icons gekennzeichnet.

Entlang der *playedBy* Beziehung können Methodenbindungen neue Kontrollflüsse erzeugen, die im Quelltext nicht direkt sichtbar sind. Deshalb ist es wichtig, dass diese Bindungen auch auf der textuell nicht veränderten Basisseite durch spezielle Marker im Editor visualisiert werden. Für die ersten Schritte der Programmierung bieten wir spezielle Wizards für die Erstellung von Teams und Rollen an.

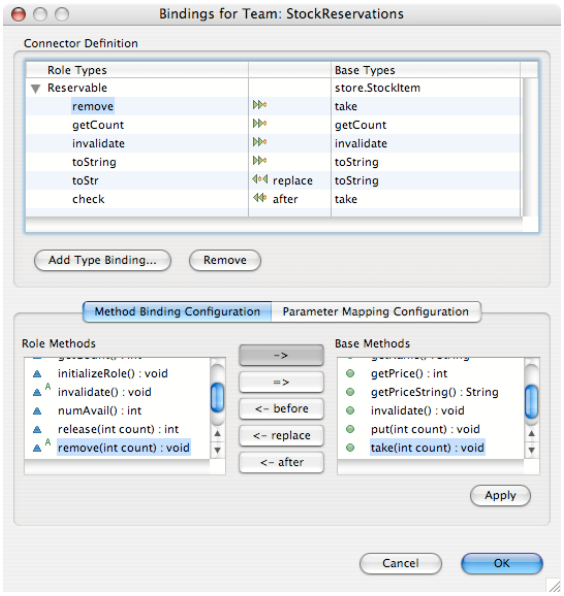
Neue Ansichten

Die Teamvererbung erweitert die normale Vererbung zwischen Klassen. Hier sorgt eine neue Hierarchieansicht für Übersicht.

Eine besondere Verwendung von Teams besteht darin, ein Team als *Konnektor* zu entwerfen, was bedeutet, dass dieses Team keine Implementierung enthält, sondern nur Bindungen zwischen existierenden Elementen (Klassen und Methoden). Ein solcher Konnektor kann in dem so genannten Bindingeditor (s. Abb.) komplett tabellenbasiert erstellt werden.

Refactoring

Da die neuen Beziehungen neue Abhängigkeiten innerhalb eines Programms darstellen, musste auch die Implementierung der Refactoring-Unterstützung



angepasst werden. Für die gängigsten Refactorings stellt das OTDT sicher, dass die Semantik eines OT/J Programms durch das Refactoring nicht verändert wird.

Debugging

Durch die Byte-code Transformationen der Laufzeitumgebung gibt es strukturelle Unterschiede zwischen dem Quelltext und dem von der Java Virtual Machine ausgeführten Bytecode. Ein erweiterter JDT-Debugger berücksichtigt diese Diskrepanz, um das Setzen von Breakpoints, schrittweises Ausführen sowie das Inspizieren des Programmzustandes zu ermöglichen.

3 Weitere Werkzeuge

Neben der Kernfunktionalität des OTDT gibt es eine Reihe von zusätzlichen Werkzeugen.

Durch eine spezielle Integration der Laufzeitumgebung mit dem Eclipse-Framework ist es möglich, OT/J auch zur Entwicklung von Eclipse Plugins einzusetzen. In der Tat sind einige Teile des OTDT selbst mit OT/J entwickelt.

Ein Modellierungswerkzeug erlaubt das Modellieren mit einer speziellen Erweiterung der UML (UFA). Aus diesen Modellen kann direkt OT/J Quelltext generiert werden. Ein Metrikwerkzeug berechnet quantitative Eigenschaften der Software. Eine Erweiterung des Visual Editor (VE) [http://www.eclipse.org/vep] erlaubt das interaktive Erstellen von grafischen Benutzungsoberflächen mit Erzeugung von OT/J Quelltext, der die speziellen Fähigkeiten der Sprache ausnutzt, um eine besonders klare Umsetzung des Model-View-Controller Konzeptes zu realisieren.

Software, Publikationen, ein Mailforum sowie weiterführende Links:

http://www.objectteams.org