Automated Refactoring of Java Contracts

A dissertation proposal for an MSc in Advanced Software Engineering by Iain Hull

Supervisor	Dr Joseph Kiniry
Subject Area	Software Engineering
Pre-requisite	Good knowledge of Java
Co-requisite	Knowledge of the Eclipse plugin development, JML, formal software engineering processes, and verification will be obtained during the course of this project
Subject Coverage	Integrated Development Environments, Formal Specifications, Verification, Logic
Project Type	Design & Implementation
Hardware	PC or workstation (on nearly any operating system)

Description

Over the past decade the software industry has crossed a rubicon in how it deals with the prospect of change. Previously it tried to manage or control change, it put a lot of effort into plans and designs to avoid it. Recently the industry has started to accept change as inevitable, and moved its focus onto how to live with it and even embrace it. Agile processes and practices have been developed and reworked to help software organisations cope with change.

One of these, Refactoring (1), is now so ubiquitous that modern integrated development environments include automated support for common Refactoring operations. These automated operations guarantee that the code's behaviour will not change as a result of the refactoring.

One of the first tools to offer support for refactoring in the Java language was The Eclipse IDE (2). This has had support for Refactoring since version 2.0. Now the refactoring support has been generalised to support the development of new in Refactorings and new Languages (3).

Design-by-Contract is a slightly older concept pioneered by Meyer and the language Eiffel (4). Its use and importance are growing beyond its traditional domain of mission critical software and formal methods. This is especially true as support of Design-by-Contact has spread to the tools and languages most developers use. This includes the Java Modeling Language, JML (5), which adds support for Design By Contract to the Java language using Java annotations in special comments (6).

Design By Contract controls how software components collaborate with one another by explicitly specifying the rules they must follow to talk to one another. These rules include:

- The preconditions a client agrees to before initiating an interaction.
- The postconditions the supplier agrees to after completing the interaction
- Invariants that both can expect to always hold true.

These Contracts, or Specifications, are used to document the design and aid its implementation. Once in place they aid testing, debugging and reasoning about the implementation.

While Design-by-Contract reduces the flaws in a system, thereby reducing the subsequent changes it requires, it cannot eliminate the need for other types of changes. As a result these systems must be refactored as they evolve. Since the Specifications describe the design, any changes to the design require changes to these Specifications. While Refactoring support has been added to other Java Specification Languages these tools have not been released (7), currently there are no tools that can Refactor JML Specifications.

Mandatory

- Extending the Eclipse IDE, with a new plug-in, to add support for Refactoring JML Specifications. JML is currently evolving quite rapidly to incorporate the latest Java features, so to remain current these refactorings will operate on the new Java Contracts (7) representation of JML, (a plain Java representation of the classic JML constructs)
- Initially this will only be concerned with Specification only Refactorings, involving:
    - Pull-up Specification
    - Push-down Specification
- All the possible combinations of, specification type, visibility, source type, target type will be investigated and documented by hand. These will then make up a suite of automated tests to prove the refactoring functionality.
- The plug-in will use the Eclipse JDT's Java parser and AST to separate and comprehend the JML Specifications and surrounding the Java code to be refactored.
- The refactorings will use preconditions to test the suitability of the refactoring before it is performed. Postconditions will be used to test the results of the refactoring.

Discretionary

- Additional Refactorings could include:
    - Change Specification visibility
    - Introduce Model

Exceptional

- The modifications made to the Java code and JML specifications will be formally specified and proven to be equivalent to the original.

Works Cited

1. **Fowler, M.** *Refactoring. Improving the Design of Existing Code.* s.l. : Addison-Wesley, 1999.
2. The Eclipse Project. [Online] 2009. http://www.eclipse.org.
3. **Widmer, T.** Unleashing the Power of Refactoring. [Online] 2009. http://www.eclipse.org/articles/article.php?file=Article-Unleashing-the-Power-of-Refactoring/index.html.
4. **Meyer, B.** *Eiffel: The Language.* s.l. : Prentice Hall, 1990.
5. The JML Project. [Online] 2009. http://www.cs.ucf.edu/~leavens/JML/.
6. **G. Leavens, Y. Cheo.** Design by Contract with JML. [Online] 2009. http://www.jmlspecs.org/jmldbc.pdf.
7. **Goldstein, M. Feldman, Y.A. Tyszberowicz, S.** Refactoring with contracts. *Agile Conference.* 2006.
8. **Patrice Chalin, Robby.** Java Contracts: A Quick Overview. [Online] 2009. http://jmlspecs.svn.sourceforge.net/viewvc/jmlspecs/org.jmlspecs.javacontract/trunk/doc/overview.doc.