CS 298 Proposal : Designing a Programming Contract Library for Java

Guided by Dr. Thomas Austin

By :
Neha Rajkumar
San José State University

June 3, 2015

Abstract

As technology is advancing there has been a significant growth in software systems. Programmers are now developing large and complex software systems, so it is important to have software that is consistent, efficient, and robust. Software must work correctly; it must not fail giving erroneous outputs; It must be able to restore to a consistent phase even when an error occurs while informing the programmer about the cause of the error [4]. To improve software reliability a programmer must be able to define what actions each software element is supposed to do, and determine who is at fault when things break down. This can be achieved by programming contracts. Design by contract is a principle used in various programming languages to define formal specifications for software components. The design by contract principle [3] was first used in the Eiffel programming language [2]. Programming contracts includes preconditions, postconditions and invariants [5]. The contract denotes the relation between the client and the supplier. The contract is said to be broken if the client does not meet the preconditions of the supplier or the supplier has not met the postconditions. The software is correct when the preconditions, postconditions, and the invariants are true [1]. So the design by contract helps to build bug-free software leading to safe exceptional handling. The purpose of my project is to design a programming contract library for Java.

1 Project Deliverables

The list of project deliverables includes the following

- Library supports a set of preconditions and postconditions that are specified in Java annotations
- Incorporates contract checking for objects of subclasses
- Support for writing contracts in scripting languages
- Support contracts for lambdas in Java 8

2 Challenging aspects of Project

The main challenge involved in implementing the library is to check for the preconditions and postconditions using AspectJ. The contracts are specified through annotations and the conditions are checked at run time through the before and after advice in AspectJ.

Other challenges involved in the project are the use of different scripting languages for writing contracts. The contract library supports scripting languages like Jython, Ruby, and Javascript where the programmer could write the conditions based on these languages. These scripting languages allow the programmer to write more complex conditions that can be checked during program execution.

3 Schedule

Table 1: Timeline

Week 1: May 4th - May 8th	Research about programming contracts
Week 2-3: May 11th - May 22nd	Writing sample programs using assert, and Java libraries
Week 4: May 25th - May 29th	Learn about defensive programming
Week 5: June 1st - June 5th	Research about custom annotations
Week 6-7: June 8th - June 19th	Implement quicksort using programming contracts
Week 8-9: June 22nd - July 3rd	Developing similar applications using programming contracts
Week 10-11: July 6th - July 17th	Survey on existing programming contracts
Week 12-13: July 20th - July 31st	Research on ways to improve the library
Week 14-15: Aug 1st - Aug 15th	Implement contract checking on objects
Week 16-17: Aug 17th - Aug 28th	Research on scripting languages to implement as conditions
Week 18-19: Sept 1st - Sept 11th	Implement contract conditions using Jython
Week 20-21: Sept 14th - Sept 25th	Implement contract conditions using Javascript
Week 22-23: Sept 28th - Oct 9th	Implement contracts for lambdas in Java 8
Week 24: Oct 12th - Oct 16th	Implement contract conditions using Ruby
Week 25: Oct 19th - Oct 23rd	Write test cases for the library
Week 26-27: Oct 26th - Nov 6th	Write report
Week 28-29: Nov 9th - Nov 20th	Prepare for defense

References

[1] Bertrand Meyer. Applying design by contract. *Computer*, 25(10):40–51, 1992.

[2] Bertrand Meyer. *Eiffel: the language*. Prentice-Hall, Inc., 1992.

[3] Richard Mitchell, Jim McKim, and Bertrand Meyer. *Design by contract, by example*. Addison Wesley Longman Publishing Co., Inc., 2001.

[4] Jeffrey E Payne, Michael A Schatz, and Matthew N Schmid. Implementing assertions for java. *Dr. Dobb's Journal*, 23(1):40–44, 1998.

[5] Richard S Wiener. *Software development using Eiffel*. Prentice-Hall, 1995.