Final Year Project

BON Software Model Consistency Checker for Eclipse

Eva Darulová

Project Management Deliverable submitted in part fulfilment of the degree of

BA/BSc (hons) in Computer Science

Supervisor: Dr. Joseph Kiniry

Moderator: Dr. Barry Smith

UCD School of Computer Science and Informatics
College of Engineering Mathematical and Physical Sciences
University College Dublin
November 26, 2008

Abstract

Modelling languages specifying a system independently from its implementation are particularly useful during analysis and design stages of a system. An implementation-specific formal language translates those requirements directly into code, annotations and possibly assertions. Both approaches have their advantages and should ideally be used hand in hand. In practice however, mostly due to lack of tool support, the model of a project will not be updated once a formal specification is in place, thus rendering it useless. This project interlinks the BON and JML modelling languages by creating a mapping between their structure and assertion features and by providing tool support for consistency checking.

Chapter 1: **Introduction**

Business Object Notation (BON) has been developed for designing and analysing object-oriented programs. It is designed to enable a seamless and reversible development process as well as software contracting. It provides a textual, graphical and an informal representation of the system to be developed. Thus it can close the communication gap between technical and non-technical people (e.g. programmer and project manager) involved in the design and implementation of software.

The Java Modelling Language (JML) is a formal specification language for Java. It follows the Java syntax closely and its annotations are inserted directly into code. By doing so, it is easy for developers to learn and convenient to apply. It employs the design by contract approach by specifying preconditions, post-conditions and invariants. With its extensive tool support it is suitable for use in the development of commercial software.

To make the most of a software model it has to be used throughout the development process, not just for the initial design. For example, it can be used to update high-level manager on possible changes in the implementation without confusing them with technical details. This can only be achieved when the model is always up-to-date during the whole of the development process. This project aims to provide such synchronisation by checking the consistency between a BON model and the corresponding Java source code with JML annotations.

Mandatory:

1. Familiarisation with software modeling and the BON method/language
2. Familiarisation with design by contract and the Java Modeling Language (JML)
3. Familiarisation with development of Eclipse plugins
4. Create a mapping between BON features and JML features and identify possible issues where the two notations may not be (fully) compatible
5. Implement an Eclipse plugin that reads in a BON file (format to be decided) and highlights differences in the corresponding Java/JML souce code
6. Design and implement a test program Discretionary

Discretionary:

1. Design a GUI that is clearly arranged and flexible
2. Include the option to generate Java skeleton code with JML annotations from BON model
3. Make checks possible in both directions (BON - Java and Java - BON)

4. Internationalize the plugin
5. Make the plugin effective (e.g. it will only check lines of code that have changed since the last check)
6. Allow for JML and BON to be extendable (e.g. custom tags, keywords)

Exceptional:

1. Write, submit, and publish a paper on the results.

Reading:

- Seamless Object-Oriented Software Architecture—Analysis and Design of Reliable Systems by Kim Walden and Jean-Marc Nerson. Available online.
- JML Homepage,
  http://www.eecs.ucf.edu/ leavens/JML/
- Developing Eclipse Plugins (IBM),
  http://www.ibm.com/developerworks/library/os-ecplug/
- Java tutorial on Internationalization,
  http://java.sun.com/docs/books/tutorial/i18n/intro/index.html

Chapter 2: **Project Management**

The following sections give a brief outline of the realisation of the project.

2.1 Tasks

The detailed project time plan from December to the beginning of May is given in Figure 1.1. The tasks are divided into three main parts: Research, Implementation and Documentation. It is aimed that all background research will be completed by February. The implementation of the fundamental functionality is phased from December to mid March, since the individual parts depend on each other. The presentation, such as the implementation of the plug-in and the GUI can be developed in parallel and are planned for January until April. These also include optional features, which, if delays should occur, will be abandoned. Finally, the documentation will be written towards the end of the project in March and April. This includes the final report as well as the documentation of the actual application. Testing and code maintenance are ongoing processes and are thus planned after each major task.

2.2 Milestones

The project's progress is measured on the following five milestones:

1 - Basic functionality A basic implementation that can compare BON and Java source code. Simple output is communicated to the user.

2 - Eclipse plug-in The basic functionality has been inserted into an Eclipse plug-in, so that the application can be directly used from within Eclipse without need for a command-line. Results are also presented in Eclipse.

3 - Graphical User Interface A GUI is implemented which makes the use of the application comfortable and straightforward. If options can be selected, they will be selected here.

4 - Advanced functionality Advanced functionality is implemented that also includes consistency checking between BON logic constructs and JML.

5 - Finished application A fully working Eclipse plug-in, which integrates basic and advanced functionality into the Eclipse IDE.

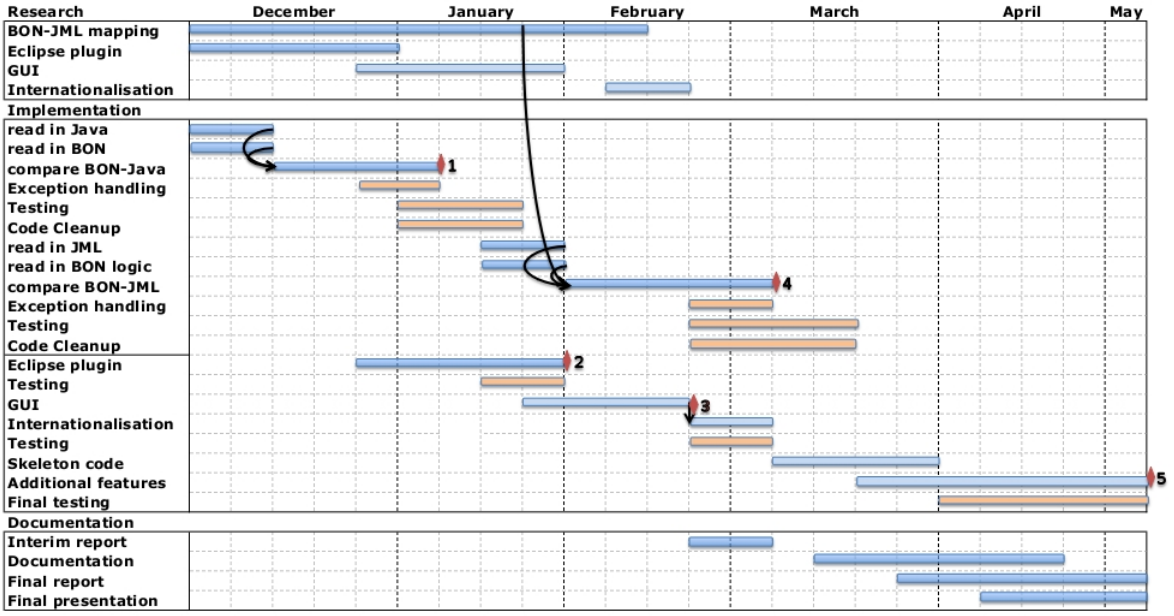

Figure 2.1: Tasks and Milestones. Priorities are denoted by the following colours: blue - mandatory, light blue - optional, orange - not in project description, but nevertheless graded. Milestones are indicated by red diamond shapes.

2.3 Risks

2.3.1 Software

The project will be implemented in Java. Since this is a widely used programming language, no complications are expected. Furthermore, the project will use and so will rely on the open source projects OpenJML, BONc and Eclipse so that issues and incompatibilities may arise.

- OpenJML has already been tested on a small scale and deemed utilisable for this project.
- Eclipse plug-ins are being developed frequently, thus there exists are extensive community, so if problems should arise, there are resources for finding solutions.
- BONc is being developed by a PhD researcher in UCD, who can be contacted directly if trouble shooting should be needed. Problems may arise here, since BON logic constructs are not being fully parsed at the moment. A separate parser will thus have to be written for reading in BON logic.

2.3.2 Content

Since BON and Java/JML are two independent languages, it is to be expected that they will not be (fully) compatible in all features. These features have to be identified and, if possible, workarounds should be found, so that these incompatibilities will have as little effect on the overall application as possible.