Raj Dandage December 6, 2001

6.191 Final Paper

The project that I have chosen is for the Software Development Group (SDG) at LCS. I intend to work with Daniel Jackson on software modeling, a research field that he is interested in. This project will begin with a UROP over IAP and continue throughout next spring and next year.

I. Research Group Overview

SDG focuses on improving software. Improving means making the software cycle more reliable and efficient in all stages of development. There are several parts of the software cycle that can be improved.

SDG looks at software design, by defining languages and methodology for specifying the workings of a piece of software before it is written. To this end, they do research in specifications, which includes writing specs that clearly define the input and output of programs. They also do research in modeling and invariants.

During the programming phase, they look at how to write code with fewer bugs, more specific to what was requested. They also look at testing, and how to correctly glass-box test and black-box test. This is extended after the program is complete. SDG looks into how to ensure a program was written to its specification, and whether it works in different executions, with different inputs or environmental factors.

Finally, they look at what to do with a program after it is written. That is, how to maintain it, how to update it, how to fix bugs that may appear, and more. There are several different areas of research in this field, but they all focus on making sure that the end product works as the user expects.

II. Specific Technology

SDG has several technologies that pertain to my project; however, the most important ones are Alloy, the object modeling language developed by Daniel Jackson, and its extensions, Jalloy and Womble. These technologies all help define how a program works and its output states.

Alloy is an object modeling language, similar to UML, but it is much more expressive. It is both a language to express models and a program to execute these models. The language has precise semantics, unlike other languages for expressing object models, that makes it well-suited for creating models. In a short amount of time, a designer can express a complete system, and then he can run it through the constraint analyzer to check certain properties of the system.

The constraint analyzer works by using a solver to try to find conditions in which a predicate holds. For example, if you wanted to check whether the program ever tries to divide by zero, you could write that as a predicate and run the analyzer to check if that condition ever happens. Although the constraint analyzer is not a proof-checker, and therefore is not guaranteed to find an answer if it exists, it increases the likelihood of finding a bug in a design by many factors.

Jalloy and Womble, meanwhile, are systems for extracting models from Java code. Jalloy reads Java specs and builds a Alloy model from them. This can be used to check code for bugs or invariants. Womble does a similar thing from the Java .class file.

This means that existing programs can be checked with the Alloy analyzer without any changes to the program. This makes it easy to check whether a piece of software met its requirements and specifications.

III. The Problem

One of the problems with Alloy, and with object modeling languages in general, is that they are not versatile enough to model web applications. Web applications differ from traditional software in that they are not homogeneous. There are many reasons for this:

1. We applications are transactional. Their execution cycle lasts only an instant when they are computing something. They begin, are passed a request, compute something based on that request, output a result, and then close. This means that they cannot rely on a complex object hierarchy to store data, because that hierarchy will be destroyed as soon as the request is served.
2. They are tightly bound to the application server. That is, they often use many components that are integrated in the server, such as session maintenance, load balancing, and connection pooling. This means that their state is very closely tied to the state of the server and other external software.
3. They often use external data sources for temporary and long-term storage. Most web applications, for example, use a database to store information between transactions. Because most databases do not handle objects, this means the object hierarchy must be translated to relational, table-based information for storage and then translated back.

4. Many span multiple languages and operating environments. They may use Java on the server, SQL for database connectivity, JavaScript on the client browser, Perl for scripting, etc. Data must be correctly translated and state must be correctly maintained across these languages, which is often very difficult.

Because of these reasons, traditional object modeling techniques, including Alloy and the Alloy constraint analyzer, are not well suited to this problem.

IV. The Solution

The solution to this problem is the topic of my M. Eng. thesis. Basically, what I intend to do is come up with an extension to Alloy to allow developers to simulate web applications easily. This will be done in two parts.

The first of these parts is to design an Alloy web application framework. What this will do is provide software designers Alloy tools with which to do this. These tools will come in the form of Alloy models of common web application components. I will create a model of a Java 2 Enterprise Edition application server, such as Weblogic or Dynamo, that designers can plug their designs into and test with. This model will include several sub-components. These components will model session management (which is probably the most common place for bugs in a web application). They will also model load balancing, connection pooling, and bean delivery.

I will also create models of other components of web applications. For example, an SQL-based relational database. While these are extremely complicated systems, they are fairly easy to model in Alloy because we do not need to worry about performance or intricate details. The database model will allow designers to test the translation of their data to and from the database. I will also model the communication between a web browser and the server. This includes the HTTP protocol and the content handling.

The other part of my solution will be to extend Jalloy or Womble to be able to extract useful models from a web application that is already written. The modifications will rely on the models of the server and database and other components described above. It will use these components wherever they appear in the application. With this tool, developers will be able to take their web application and use the constraint analyzer to test it for bugs or errors.

V. Timeline and Resources

The timeline for this project is the following:

1. Fall 2001 – the proposal will be completed and necessary resources will be secured
2. Spring 2002 – I will do a UROP over IAP and spring to familiarize myself with Alloy and the various tools used by the group
3. Fall 2002 – I will complete the Alloy models and begin work on the Jalloy extension
4. Spring 2002 – I will complete the Jalloy extension and write the thesis

The resources required for this project are minimal. At most I will need a computer and a workspace and access to the software code developed by the group. All of this is fairly easy to acquire.