David Pick
CM2403

Inside of Praxis software engineers use Z and Spark to ensure that their software operates according to specifications. Z is a formal specification language based on standard mathematical notation. Once a program is described in the Z language it can be reviewed either by hand or using a theorem-proving software. Praxis uses Z in order to ensure that their model of the software system to be built is correct. This is similar to the way in which the alloy modeling language is used. The underlying mathematical concepts behind alloy were heavily influenced by Z, but its syntax is much closer to Object Orient languages than Z. By combining mathematics and object oriented languages, Alloy gives programmers a way to describe their systems that is much closer to the way in which they will actually be written. However since its underlying core was based on mathematics and the Z language, it can be used to accurately test a model. This makes Alloy a perfect tool for programmers who are not as familiar with the mathematical concepts used in Z as they should be. It can used to create models of systems before coding even begins, to ensure that the model is correct, and assuming it is translated into actual code correctly, will behave as the developers intended.

Spark is a programming language designed to ensure that software operates as it was intended to. Spark was based on the Ada programming language. In order to ensure that the Ada code was written correct, Spark allows a developer to add additional annotations to the comments of the program. These comments define what the input and output of a function should be. These annotations allow a developer to include formal specifications in the comments of the code. When the code is run Spark checks all the functions according to the Spark definitions in the comments. This Spark code ensures that functions do not effect any parts of memory they should not, as well as ensures that inputs and outputs of functions meet the specification. The operation of Spark is very similar to the Java Modeling Language. Like Spark JML allows developers to add annotations to normal code. The only difference being that the language is java instead of Ada.

The goal of these formal modeling languages is to create bug-free code. By combining tools like Z or Alloy with Spark or JML developers have the ability to create software that while not bug free, is much better than software developed in a traditional manor. However, when switching to a new process there are always trade-offs. The most glaring problem with using these new tools is the time that must be invested when using them. While writing Z or Alloy code before the project can help a developer understand the problem before they actually begin coding, it is still developing a solution to the problem. Along with the time trade-offs, it is also possible to add bugs into the model, as languages like Alloy and Z can be just as difficult or even more difficult to work in than traditional programming languages. Despite the additional possibility for errors and the fact that using formal methods will add to the total amount of time spent developing the software, formal methods most defiantly have their place in software development. They should most certainly be used in any mission critical application that is control of people’s lives or something equally important.