

# **Tools for the One-Time Program**

**An analysis of the quality of informal projects and the effectiveness of tools used to debug them**

Paul DiPalma and Elise Hewett

## **PROJECT SUMMARY**

Though many programs engineered today seem bullet-proof, there are often dangerous constructs that cause security weaknesses as well as many unseen bugs such as memory leaks, null pointers, and dangerous aliasing. We hope to analyze and uproot these problems by filtering completed programs through several analysis tools that are freely available such as Valgrind, Splint, ITS4, Electric Fence, and possibly others. These tools may give us an insight into the origin of any bugs and security vulnerabilities. The results from these tools will allow us to compare their relative effectiveness in finding problems.

These tools will be used to analyze both the Reliability Block Diagrams (RBD) developed by the Fall 2003 Stochastic Models class and Coppit and Painter's formally developed RBD. Since the class projects were developed by a single engineer, under extreme time constraints, and for a one use purpose, we expect to find more bugs in these informally developed RBD's than in the formally developed RBD. This hypothesis will be used to compare the tools' effectiveness in finding bugs and in helping the programmer correct the problems.

As a further analysis of the RBD program design structure, we hope to go behind the scenes and interview the programmers themselves in search of a correlation between the bugs and the design approach used. If any design techniques exist they should be compared to the formal design approach.

### **Intellectual Merit**

Classroom programmers often complete projects that perform the task at hand, but are often buggy and plagued by memory leaks. It seems that this is often due to time constraints and the plain fact that the novice student programmer is unaware of these problems and how to find them. We want to find user-friendly tools that can be easily acquired and learned by the student, so that the quality of the final project can be enhanced. We also hope that using these tools will increase the productivity of the student by finding various errors that may be overlooked and by decreasing the amount of time used in tracking down bugs whose symptoms are not indicative of their origin. As a result, the student will have more time to dedicate to overall quality. These tools provide an important learning medium for students since they also give suggestions on how to correct problems that are encountered.

### **Broader Impact**

We seek to address the problem of bugs in one-time use student programs. In sorting through all of the available tools we hope to recommend the ones that are most useful for everyday use in our department. Installing these tools on the network along with departmental encouragement of the use will be most effective in increasing quality of students' programs in our computer science department.