Estimating Software Reliability in a Haskell Programming Environment

Mark Sherriff, Laurie Williams
Department of Computer Science, North Carolina State University, Raleigh, NC 27695
{mssherri, lawilli3}@ncsu.edu

1. Introduction

Some profess that functional languages offer a good balance between productivity, reliability, maintainability, and efficiency. Recently, research and projects have been undertaken to take advantage of the benefits that functional languages present, specifically the Haskell language. This work adds to the body of reliability knowledge for systems built with the Haskell functional programming language. *Our research objective is to construct, validate, and automate two internal, in-process methods to provide an early indication of system reliability.*

The two methods that we propose in this work utilize testing and static code metrics that are gathered on a given program. An Eclipse[1] plug-in will be created to automate the gathering of these metrics and the computation, analysis, and presentation of the reliability estimates within the same development environment where code is developed. If reliability information can be presented early in the development process, preferably while a developer is working on the code, affordable corrective action can be taken to rectify reliability concerns as they appear.

2. STREW-H

Davidsson and Nagappan began work on the "Good Enough" Reliability Tool for Java (GERT-J[2]) in 2003. Their tool utilizes in-process testing metrics to estimate reliability. The metrics underlying GERT-J are those in the Software Testing Reliability Early Warning for Java systems (STREW-J) metric suite. GERT-J automates the collection and analysis of the STREW-J metrics and provides visual feedback to programmers on the reliability of various parts of the system and on the thoroughness of the test effort. Early results from a feasibility study and a structured experiment have shown that a regression equation can be formed to provide a practical estimate of software reliability.

Based on Davidsson and Nagappan's work, we propose the STREW-Haskell (STREW-H) metric suite. We utilized the STREW-J metric suite as a starting point for the STREW-H. We eliminated the metrics that were not applicable for functional languages and made additions based upon a review of the literature and upon expert opinion. Expert opinion was gathered via interviews with 12 Haskell researchers at Galois Connections, Inc. and with members of the Programatica team at The OGI School of Science & Engineering at OHSU (OGI/OHSU). From these sources, we propose an initial set of metrics for the STREW-H version 0.1. Through validation with multiple industrial projects, we will refine the proposed metric suite by adding and deleting metrics until we feel we have the minimal set of metrics needed to accurately predict and explain product reliability.

A feasibility study involving a subset of the metrics was conducted to analyze the potential of the STREW-H metric suite. The open source Glasgow Haskell Compiler (GHC) was chosen as the system under study since there were multiple versions available with detailed documentation and defect logs. No time-dependent reliability was available so the study focused on estimating a related measure, defect density. The results showed that while there were not enough test cases to denote statistical significance, this method is an efficient indicator of the defect density. More studies are in progress.

3. COPPER

The Programatica team at OGI/OHSU is working on a method for high-assurance software development. Programmers can create *certificates* on individual functions or lines of code in a program. The certificates are tied to a specific type of evidence that shows that the functions or lines of code are of high-assurance. The evidence is based on the verification and validation practices (testing, formal proofs, and development practices) used on that part of the code.

We build upon OGI/OHSU's work and propose a certification-based method of estimating reliability. We call this method the Certificates with Operation Percentages for Providing an Estimate of Reliability (COPPER) method. COPPER extends OGI/OHSU's work by associating a reliability measure with each certificate that is placed on a per-function basis. Then, a program profiler calculates the *operation percentage*, the percentage of processor time each function consumes during normal operation, for each certified function when the system is run with a representative set of test cases. The reliability of the system as a whole can be estimated based on the reliability of the certified functions multiplied by their operation percentage.

4. Eclipse Plug-In

Eclipse is an open-source development environment built around the concept that every component of the system is a plug-in, each building off another to increase functionality. Despite Eclipse being primarily a Java tool, developers have successfully implemented plug-ins to allow other languages to be used in the environment, including Haskell. Work began earlier this year by Frenzel et. al. to create a Haskell development plug-in for Eclipse. While still in its early stages, this plug-in provides syntax highlighting, a build system, module viewer, basic error checking, and others. Later versions of the plug-in will expand on this and include more functionality. Our efforts will build upon this system.

We began incorporating STREW-H and COPPER into an Eclipse plug-in earlier this year to embed these methods directly into the programmers' development process. The Eclipse framework can facilitate the automatic gathering of the various metrics needed along with calculating the required estimates. The graphical nature of Eclipse will also allow us to create a certificate structure so that the certificates described in the COPPER model will be able to be shown in-line with the code, providing visual feedback to developers.

---

[1] Eclipse is an open source integrated development environment. For more information, go to http://www.eclipse.org/.

[2] GERT-J is available at http://sourceforge.net/projects/gert/.