Tool Support for Estimating Software Reliability in Haskell Programs

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

Abstract

In late-stage phases of development, action to correct defects can be cost prohibitive. Effective, efficient, and expressive measures of reliability during the development cycle could aid developers by showing early warning indications of where the system might require modification or more testing. In this paper, we present initial research in creating an Eclipse plug-in that utilizes two methods for estimating reliability in-process in a functional programming environment. One method is based on testing and static code metrics that can be gathered automatically during the coding process. A feasibility study involving a subset of these metrics was performed. The other method is based on creating certificates for individual sections of code. These certificates are created from information gathered by validation and verification practices as evidence of reliability. They can be used together with the operational profiles of the system to estimate overall system reliability.

1. Introduction

Some profess that functional languages offer a good balance between productivity, reliability, maintainability, and efficiency [1]. Recently, research and projects have been undertaken to take advantage of the benefits that functional languages present [5], specifically the Haskell language. Haskell can be used to create nearly any type of program, but is most often used for creating system-critical, back-end functionality. This paper adds to the body of reliability knowledge for systems built with the Haskell functional programming language. *Our research objective is to construct and validate an Eclipse*[1] *plug-in that will utilize two different, internal, in-process methods to provide an early indication of system reliability.*

The two methods that we propose in this paper utilize testing and static code metrics that are gathered on a given program. An Eclipse plug-in will be created to automate the gathering of these metrics and the computation 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 any reliability concerns as they appear.

2. STREW-H

Davidsson et. al. 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. This estimation provides early feedback to developers so that they can correct faults during the development process and can increase the testing effort, if necessary, to provide added confidence in the software. The metrics underlying GERT-J are those in the Software Testing Reliability Early Warning for Java systems (STREW-J) [2, 6] 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. While the STREW-J metric suite is still in development, early results from a structured experiment [2, 6] have shown that a regression equation can be formed to provide a practical estimate of software reliability.

Based on Nagappan's work [6], 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.[2] and with members of the Programatica team[3] 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, including the following:

- *number test case asserts / source lines of code*
- *IO monadic lines of code / source lines of code*
- *number of type signatures / source lines of code*
- *number of overlapping patterns / source lines of code*
- *number of duplicate exports / source lines of code*
- *number of incomplete patterns / source lines of code*
- *number of missing signatures / source lines of code*
- *number of name shadowing / source lines of code*
- *number of test cases / number of requirements*
- *test lines of code / source lines of code*

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)[4] 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

[1] http://www.eclipse.org/

[2] http://www.galois.com/
[3] http://www.cse.ogi.edu/PacSoft/projects/programatica/
[4] http://www.haskell.org/ghc/

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.

A limitation of the STREW-H method is that it is not based on any operational profile of the system. While utilizing operational profiles to estimate reliability would be beneficial, it is often cost and time prohibitive. Rivers and Vouk have shown that non-operational testing is related to field quality [7], and thus there is potentially value in this method.

3. COPPER

Research has shown that breaking up a larger system into components is an effective means of estimating reliability [5]. However, most of these methods look at system components as being individually-running programs or classes unto themselves. This concept does not translate well to functional languages, and thus adjustments have to be made for the idea of components to be valid in this environment.

The Programatica team at OGI/OHSU is working on a method for high-assurance software development [4]. 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. The difference with this method is that most other component-based reliability methods require that each component is a working system unto itself. The proposed COPPER method goes to more atomic levels in the code, utilizing several different means of ensuring that a piece of code is reliable.

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. [3] 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.

5. Summary and future work

Having an early warning system to estimate reliability would aid developers by giving them an indication as to potential problems in the system. We can leverage metrics that can be gathered in any system with some effort to help provide this reliability estimate. An Eclipse plug-in utilizing two methods for estimating the reliability of software written in a functional language environment have been proposed in this paper. The plug-in gathers metrics and records certificates and calculates the system reliability estimate based on the two methods presented in this paper. Reliability growth information is provided to developers while they are still implementing code and can affordably make corrective actions.

Currently, further feasibility studies are underway for both methods using code from an ASN.1 compiler system developed in Haskell. Twenty different development versions with defect information are available and will be used to help verify the findings in the original feasibility study. The initial feasibility study will also be performed on the COPPER method using this data set.

Acknowledgements

This work is supported by the National Science Foundation.

References

[1] Breazu-Tannen, V., Buneman, O. P., and Gunter, C. A. "Typed functional programming for rapid development of reliable software." In J. E. Gaffney, editor, Productivity: Progress, Prospects, and Payoff. June, 1988. pp. 115-125.

[2] Davidsson, M., Zheng, J., Nagappan, N., Williams, L., and Vouk, M., "GERT: An Empirical Reliability Estimation and Testing Feedback Tool," International Symposium on Software Reliability Engineering, 2004. To appear.

[3] Frenzel, L., "Haskell support for Eclipse." http://eclipsefp.sourceforge.net/ . 2004.

[4] Halgren, T., "Tools from the Programatica Project," presented at ACM SIGPLAN Haskell Workshop, 2003.

[5] Hamlet, D., Mason, D., and Woit, D., "Theory of Software Reliability Based on Components," International Conference on Software Engineering, 2001, pp. 361-370.

[6] Nagappan, N., Williams, L., Vouk, M.A., "Towards a Metric Suite for Early Software Reliability Assessment," International Symposium on Software Reliability Engineering, FastAbstract, Denver, CO, pp.238-239, 2003.

[7] Rivers A. and Vouk, M.A., "Resource Constrained Non-Operational Testing of Software," International Symposium on Software Reliability Engineering 1998, Paderborn, Germany, 4-7 Nov., 1998.