Certification of Code During Development to Provide an Estimate of Defect Density

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

Abstract

In industry, information on defect density of a product tends to become available too late in the software development process to affordably guide corrective actions. Our research objective is to build a parametric model which utilizes a persistent record of the validation and verification (V&V) practices used with a program to estimate the program's defect density. The persistent record of the V&V practices are recorded as certificates which are automatically recorded and maintained with the code.

1. Introduction

The defect density of a software system is calculated by measuring the number of failures divided by the size of the system, using a size measure such as lines of code. In industry, post-release defect density of a software system cannot be measured until the system has been put into production and has been used extensively by end users. Actual post-release defect density information becomes available too late in the software lifecycle to affordably guide corrective actions to software quality. Correcting software defects is significantly more expensive when the defects are discovered by an end user compared with earlier in the development process [3].

Therefore, it would be beneficial if software developers could receive any indication of the defect density of the system during development. Further, if this defect density information can be presented during the development process within the environment where developers are creating the system, more affordable corrective action can be taken to rectify defect density concerns as they appear.

During software development, a development team will use several different methods to ensure that a system is of high-assurance [7]. However, the verification and validation (V&V) practices used to make a system reliable might not always be documented or this documentation may not be maintained. This lack of documentation can hinder other developers from knowing what V&V practices have been performed on a given section of code. Further, if code is being reused from an earlier project or code base, developers might spend extra time re-verifying a section of code that has already been verified thoroughly.

Research has shown that parametric models [4] using software metrics, such as the Software Testing and Reliability Early Warning (STREW) [5] suite, can be an effective means to predict product quality. *Our research objective is to build a parametric model which utilizes a persistent record of the V&V practices used during development and testing to estimate the defect density of that program.* To accomplish this objective, we are developing a method called Defect Estimation with V&V Certificates on Programming (DevCOP). This method includes a mechanism for creating a persistent record of V&V practices as *certificates* stored with the code base, a parametric model to provide an estimate of defect density, and tool support to make this method accessible for developers. A DevCOP certificate is used to track and maintain the relationship between code and the evidence of the V&V technique used. We will build the parametric model using a nine-step systematic methodology for building software engineering parametric models based on work developed at the Center for Software Engineering at the University of Southern California [1]. This method has previously been used to build other successful parametric models [2, 5].

2. DevCOP

We propose a parametric model which uses non-operational metrics to estimate defect density based upon records of which V&V practices were performed on sections of code during development. We also wish to integrate our estimation directly into the development cycle so that corrective action to reduce defect density can take place early in the development process. This is the basis for creation of the DevCOP method.

A V&V certificate in DevCOP contains information on the V&V technique that was used to establish the certificate. Different V&V techniques will provide a different level of assurance as to how reliable a section of code is. For example, a desk check of code would be, in general, considered less effective than a formal proof of the same code.

We envision the defect density parametric model to take the form of Equation 1. For each certificate type, we would sum the product of a size measure (perhaps lines of code or number of functions/methods) and a coefficient produced via regression analysis of historical data. The calibration step of the regression analysis would yield the constant factor (a) and a coefficient weighting (c_j) for each certificate type, indicating the importance of a given V&V technique to an organization's development process.

$$DefectDensity = a + \sum_{j=1}^{certificate\_type} (c_j * Size_j) \qquad \textbf{(1)}$$

The goal of the model is to provide an estimate of defect density based on V&V certificates and the coefficient weights. We anticipate that a model would need to be developed for each programming language we would study. Our current work involves the Java (object-oriented) and Haskell (functional) languages.

We are working with industry partners to gather expert opinion and our initial data sets. Developers on a small Java team using Eclipse are recording their V&V efforts using the DevCOP plug-in as the project progresses. During defect removal and bug fixes, the team will also record these efforts as a different type of certificate.

4. Limitations

In the creation of certificates, we are not assigning more importance to certain functions or sections of code over others, as is done with operational profile means of estimation. Nor are we using the severity of defects detected to affect the importance of some certificates over another. While this level of granularity could be beneficial, one of our initial goals is to make this method easy to use during development, and at this time, we think that adding this level of information could be a hindrance. Another granularity limitation is the granularity of certificates. Based on the Programatica Team's work [7] and expert opinion, it was decided that methods (rather than statements) would be the proper level of granularity for certificates.

5. Tool Support with Eclipse

We are in the process of automating the DevCOP method with little additional overhead for developers. Ease of use, along with the added benefit of being able to calculate V&V and defect information with a defect density estimate, should make the DevCOP method practical for practicing engineers. We have created the DevCOP Eclipse[1] plug-in v. 1.1 to handle the creation and management of V&V certificates during the development process[2][6]. The plug-in allows developers to create certificates during the development process within the integrated development environment (IDE) so that this information can be utilized throughout the code's lifetime.

Programmers can select one or more functions for certification through the Eclipse Package Explorer. They assign the type of certificate (i.e. Code Inspection, Pair Programming, Bug Fix, etc.) and the weight coefficient associated with it. The certificate information is then stored in an XML document that is saved in the project's workspace. The Eclipse plug-in reads and writes to this XML document as certificates are created and edited.

We have made the certificate creation process as easy and transparent as possible, and will continue to improve it in later iterations as we receive more developer feedback. The primary method in which we accomplish this method of certificate creation is through what we call Active Certificates. An *Active Certificate* is a means by which Eclipse will automatically identify changed code during a programming session to be certified by the developer. For example, if two programmers were about to start pair programming on a piece of code, they would click the Active Certificate button before they began. Eclipse would then actively record non-trivial changes to the system and will present these changes to the developers for certification at the end of the pair programming session. This concept can extend to several different types of V&V activity, such as code inspections or bug fixes.

6. Conclusions and Future Work

We have created and are validating a method for managing and leveraging the effort put into V&V by a development team to provide an estimate of software defect density in-process. If corrective actions can be taken earlier in the software development life cycle to isolate and repair software defects, overall maintenance costs can decrease.

The DevCOP method that we are proposing will aid developers by providing defect information along with other V&V management information early in the software development process. First, after a set of certificates has been created, an overall estimate of defect density can be created based on the V&V weightings using a parametric model. We are gathering data from numerous industrial programs to calibrate our method to the general case.

DevCOP also allows developers to manage the effort that is put into V&V in a place where all developers can see what measures have been taken to ensure a piece of code is reliable and to treat it accordingly. The DevCOP method assists developers in identifying and analyzing sections of code that have not yet been certified, or to concentrate their efforts on a particularly critical section of code.

This information can also be used to provide a V&V history for particular code segments. Development teams can see what efforts were used to verify the code, even if a different team was working on the system or if poor documentation was available. If the code is found to be error-prone, the certificate information can provide guidance as to what techniques might need to be improved in the organization.

7. Acknowledgements

This work is supported by the National Science Foundation.

8. References

[1] Boehm, B. W., "Building Parametric Models," *International Advanced School of Empirical Software Engineering*, Rome, Italy, September 29, 2003.

[2] Boehm, B. W., Horowitz, E., Madachy, R., Reifer, D., Clark, B., Steece, B., Brown, A. W., Chulani, S., and Abts, C., *Software Cost Estimation with COCOMO II*. Upper Saddle River, NJ: Prentice Hall, 2000.

[3] Dolbec, J. and Shepard, T., "A Component Based Software Reliability Model," *Conference of the Centre for Advanced Studies on C*, 1995.

[4] International Society of Parametric Analysts, "Parametric Estimating Handbook." Available Online. Online Handbook. http://www.ispa-cost.org/PEIWeb/Third_edition/newbook.htm.

[5] Nagappan, N., "A Software Testing and Reliability Early Warning (STREW) Metric Suite," PhD Dissertation, North Carolina State University, 2005.

[6] Sherriff, M., Williams, L., "Tool Support For Estimating Software Reliability in Haskell Programs," *Student Paper, IEEE International Symposium on Software Reliability Engineering*, St. Malo, France, 2004, pp. 61-62.

[7] The Programatica Team, "Programatica Tools for Certifiable, Auditable Development of High-Assurance Systems in Haskell," *High Confidence Software and Systems*, Baltimore, MD, 2003.

[1] For more information, go to http://www.eclipse.org/.

[2] The plug-in is available at http://arches.csc.ncsu.edu/sherriff/devcop/.