A Method for Verification and Validation Certificate Management in Eclipse

Mark Sherriff
North Carolina State University
Raleigh, NC, USA 27695
+1-919-513-5082
mark.sherriff@ncsu.edu

Laurie Williams
North Carolina State University
Raleigh, NC, USA 27695
+1-919-513-4151
williams@csc.ncsu.edu

ABSTRACT
During the course of software development, developers will employ several different verification and validation (V&V) practices with their software, but these efforts might not be recorded or maintained in an effective manner. Our research objective is to build a method which allows developers to track and maintain a persistent record of the V&V practices used during development and testing. The persistent record of the V&V practices are recorded as certificates which are automatically stored and maintained with the code and creates traceability from the V&V practices to the code We have created a system that aids developers in the management of certificates in the Eclipse development environment. Also, we are researching a method to utilize a parametric model in conjunction with this V&V information to estimate the defect density of that program.

Categories and Subject Descriptors
D.2.8 **[Software Engineering]:** Metrics - *Performance measures*, *Process metrics*, *Product metrics.*

General Terms
Measurement, Design, Reliability

Keywords
Software Reliability Engineering, Reliability Estimation, Validation and Verification Management

1. INTRODUCTION
During software development, a development team will use several different methods to ensure that a system is of high-assurance [14]. However, the verification and validation (V&V) practices used to make a system reliable might not always be documented effectively or this documentation may not be maintained properly. This lack of proper 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.

One way to improve the documentation and management of V&V efforts is through the creation of certificates associated with the code base. *Certificates* are a record of a V&V practice employed by developers and can be used to support traceability between code and the evidence of the V&V technique used. These certificates can be automatically created, maintained, and verified by software tools, which allows developers to utilize them without excessive overhead.

Our research objective is to build a method which allows developers to track and maintain a certificate-based persistent record of the V&V practices used during development and testing. Further, we will build a parametric model which utilizes these certificates to provide an estimate the defect density of a program. To accomplish this objective, we are developing and automating a method called **D**efect **E**stimation with **V**&V **C**ertificates **o**n **P**rogramming (DevCOP). This method includes: (1) a mechanism for creating a persistent record of V&V practices as **certificates** stored with the code base; (2) **tool support** to make this method accessible for developers; and (3) a **parametric model** to provide an estimate of defect density.

We are currently developing a plugin for the Eclipse[1] integrated development environment (IDE) to support certificate management. The DevCOP Eclipse plugin allows developers to create, manage, and store certificates with the code base itself. We are also developing the DevCOP parametric model to provide an estimate of defect density 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 [2, 11]. Research has shown that parametric models [5] using software metrics, such as the Software Testing and Reliability Early Warning (STREW) [8, 12] suite, can be an effective means to predict product quality. Due to the increasing cost of correcting defects during the software development lifecycle, developers can benefit from early information regarding the defect density of their product.

In this paper, we describe our current work in developing and validating the DevCOP parametric model and the DevCOP Eclipse plugin to support the creation and maintenance of DevCOP certificates.

2. BACKGROUND
In this section, we will discuss the relevant background work and methodologies used during our research, including metric-based defect density estimation; V&V techniques; and parametric modeling in software engineering.

2.1 Parametric Modeling
Parametric models relate dependent variables to one or more independent variables based on statistical relationships to provide an estimate of the dependent variable with regards to previous data [5]. The general purpose of creating a parametric model in software engineering is to help provide an estimated answer to a software development question early in the process so that development efforts can be directed accordingly. The software development question could relate to what the costs are in creating a piece of software, how reliable a system will be, or any number of other topics.

Parametric modeling has been recognized by industry and government as an effective means to provide an estimate for project cost and software reliability. The US Department of

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

Defense, along with the International Society of Parametric Analysts, acknowledges the benefit of using parametric analysis, and encourages their use when creating proposals for the government [5]. The Department of Defense claims that parametric modeling has reduced government costs and also improved proposal evaluation time [5].

Boehm developed the Constructive Cost Model (COCOMO) [3] to estimate project cost, resources, and schedule. Further, the Constructive Quality Model (COQUALMO) added defect introduction and defect removal parameters to the COCOMO to help predict potential defect density in a system. Nagappan [8] created a parametric model with his Software Testing Reliability Early Warning (STREW) metric suite to create an estimate of failure density based on a set of software testing metrics. In our research, we will also build a parametric model to estimate defect density based upon V&V certificates recorded with the code.

2.2 Verification and Validation Techniques

During the creation of software, a development team can employ various V&V practices to improve the quality of the software [1]. For example, different forms of software testing could be used to validate and verify various parts of a system under development. Sections of code can be written such that they can be automatically proven correct via an external theorem prover [14]. A section of a program that can be logically or mathematically proven correct could be considered more reliable than a section that has "just" been tested for correctness.

Other V&V practices and techniques require more manual intervention and facilitation. For instance, formal code inspections [4] are often used by development teams to evaluate, review, and confirm that a section of code has been written properly and works correctly. Pair programmers [15] benefit from having another person review the code as it is written. Some code might also be based on technical documentation or algorithms that have been previously published, such as white papers, algorithms, or departmental technical reports. These manual practices, while they might not be as reliable as more automatic practices due to the higher likelihood of human error, still provide valuable input on the reliability of a system.

The extent of V&V practices used in a development effort can provide information about the estimated defect density of the software prior to product release. The Programatica team at the Oregon Graduate Institute at the Oregon Health and Science University (OGI/OHSU) is working on a method for high-assurance software development [14]. Programmers can create different types of *certificates* on sections of code based on the V&V technique used by the development on that section of the code. Certificates are used to track and maintain the relationship between code and the evidence of the V&V technique used. Currently, the three types of V&V techniques that Programatica can create certificates for include expert opinion, unit testing, and formal proof. These certificates are used as evidence that V&V techniques were used to make a high-assurance system [14]. We propose an extension of OGI/OHSU's certificates for defect density estimation whereby the estimate is based upon the effectiveness of the V&V practice for identifying defects (or lack thereof) used in code modules.

2.3 Metrics to Predict Defect Density

Operational profiles have been shown to be effective tools to guide testing and help ensure that a system is reliable [6]. An operational profile is "the set of operations [available in a system] and their probabilities of occurrence" as used by a customer in the normal use of the system [7]. However, operational profiles are perceived to add overhead to the software development process as the development team must define and maintain the set of operations and their probabilities of occurrence. Rivers and Vouk recognized that operational profile testing is not always performed when modern constraints on market and cost-driven constraints are introduced [9]. They performed research on evaluating non-operational testing and found that there is a positive correlation between field quality and testing efficiency. Testing efficiency describes the potential for a given test case to find faults at a given point during testing. Our research uses non-operational methods to avoid excessive overhead, while still providing valuable information.

Nagappan [8] performed research on estimating failure density without operational profiles by calibrating a parametric model which uses in-process, static unit test metrics. This estimation provides early feedback to developers so that they can increase the testing effort, if necessary, to provide added confidence in the software. The STREW metric suite consists of static measures of the automated unit test suite and of some structural aspects of the implementation code. Case studies [8] indicate that the STREW-J metrics can provide a means for estimating software reliability when testing reveals no failures. Another version of the STREW metric suite was developed specifically for the Haskell programming language, STREW-H [12, 13]. STREW-H was similarly built and verified using case studies from open-source and industry. These findings also showed that in-process metrics can be used as an early indicator of software defect density for Haskell programs. In our research, we use a similar approach to predict defect density, taking software metrics and using a parametric model to provide early defect feedback to developers

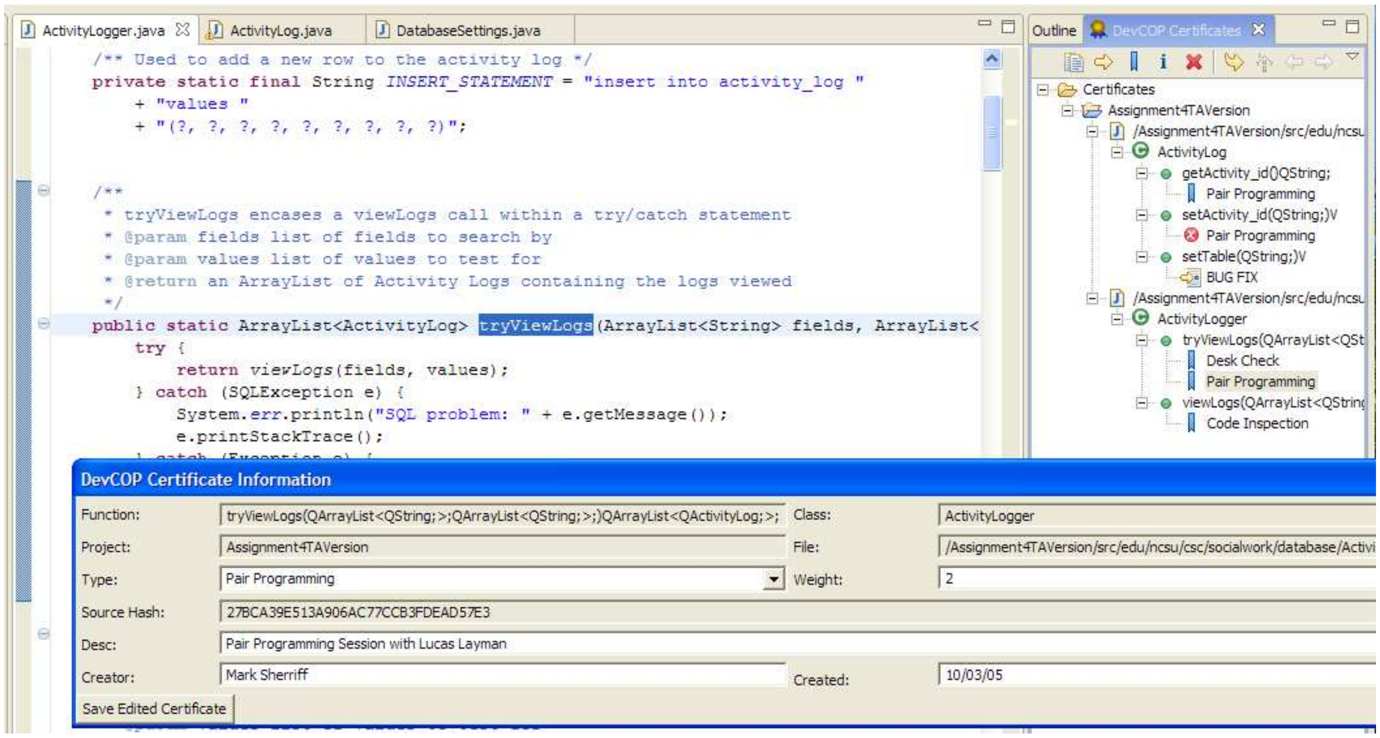
3. THE DevCOP ECLIPSE PLUGIN

We have created a DevCOP Eclipse plugin to handle the creation and management of V&V certificates during the development process[2] [10, 13]. The main purpose of the plugin is to automate the DevCOP method with little additional overhead for developers. The plugin allows developers to create and store certificates during the development process within the IDE so that this information can be utilized throughout the code's lifetime for defect density estimation purposes, for maintenance purposes, for analysis of the effectiveness of certain V&V practices, or for future reference in reused code. Figure 1 shows a screenshot of the Eclipse plugin for recording V&V certificates.

The current version of the plugin, Version 1.1.1, focuses on recording certificates that normally do not produce artifacts that are stored with the code. In other words, the plugin will aid developers by recording information about manually-performed V&V, not automatic or programmatic V&V, such as unit testing. Programmers can select one or more functions for certification through the Eclipse Package Explorer. They select the type of certificate (i.e. Code Inspection, Pair Programming, Bug Fix) and the relative importance of the certificate as 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 plugin reads and writes to these XML documents as certificates are created and edited.

The example certificate shows the information that is recorded about the V&V technique and the function on which the technique was used. Basic identifying information, such as the name of the

[2] The plugin is available at http://agile.csc.ncsu.edu/mssherri/devcop/.



```
    /** Used to add a new row to the activity log */
    private static final String INSERT_STATEMENT = "insert into activity_log "
        + "values "
        + "(?, ?, ?, ?, ?, ?, ?, ?, ?)";


    /**
     * tryViewLogs encases a viewLogs call within a try/catch statement
     * @param fields list of fields to search by
     * @param values list of values to test for
     * @return an ArrayList of Activity Logs containing the logs viewed
     */
    public static ArrayList<ActivityLog> tryViewLogs(ArrayList<String> fields, ArrayList<
        try {
            return viewLogs(fields, values);
        } catch (SQLException e) {
            System.err.println("SQL problem: " + e.getMessage());
            e.printStackTrace();
```

```
Certificates
  Assignment4TAVersion
    /Assignment4TAVersion/src/edu/ncsu
      ActivityLog
        getActivity_id()QString;
          Pair Programming
        setActivity_id(QString;)V
          Pair Programming
        setTable(QString;)V
          BUG FIX
    /Assignment4TAVersion/src/edu/ncsu
      ActivityLogger
        tryViewLogs(QArrayList<QSt
          Desk Check
          Pair Programming
        viewLogs(QArrayList<QString
          Code Inspection
```

DevCOP Certificate Information

Function:	tryViewLogs(QArrayList<QString;>;QArrayList<QString;>;)QArrayList<QActivityLog;>;	Class:	ActivityLogger
Project:	Assignment4TAVersion	File:	/Assignment4TAVersion/src/edu/ncsu/csc/socialwork/database/Activi
Type:	Pair Programming	Weight:	2
Source Hash:	27BCA39E513A906AC77CCB3FDEAD57E3		
Desc:	Pair Programming Session with Lucas Layman		
Creator:	Mark Sherriff	Created:	10/03/05

Save Edited Certificate

Figure 1. Screenshot of the DevCOP Eclipse plugin for recording V&V certificates.

function and its location along with creation information, are all included. At the time of creation, the certificate stores a hash of the source code with the certificate. The plugin determines whether a certificate is valid or not by comparing a stored hash of the source code at the time of certification with that of the current source code. If a change is made to a function's code, the source hash stored with the certificate no longer matches that of the function's source code, and the certificate is invalidated.

Our objective is to make 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 (i.e. changes to the abstract syntax tree of the code, not commenting or formatting changes) and will present the affected functions to the developers for certification at the end of the pair programming session. The concept of Active Certificates can extend to several different types of V&V activity, such as code inspections or bug fixes. Active Certificates allow developers to write or modify code normally, without increasing their work overhead.

Current improvements are being made to the plugin based on feedback from development teams. Suggestions such as integrating the DevCOP plugin with the refactoring mechanism in Eclipse or with the source control system itself are being considered. The automatic creation of other types of certificates is another feature under development. Certificates will be dynamically created from automated testing suites or other similar methods of V&V. We are also adding reporting functionality to the plugin to provide developers tools for evaluating their overall V&V effort, including identifying sections that might not have been covered by any V&V techniques as of yet.

4. THE DevCOP PARAMETRIC MODEL

A V&V certificate in DevCOP contains information on the V&V technique that was used to establish the certificate and is associated with a specific function in a piece of code. 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, less effective than a formal proof of the same code.

We are developing 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 [10]. We also wish to integrate our estimation directly into the development cycle so that developers may take corrective measures earlier in the development lifecycle.

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.

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

To build and verify our parametric model of our DevCOP method, we are utilizing the nine-step modeling methodology [11]:

1. Determine model needs;
2. Analyze existing literature;
3. Perform behavioral analysis;
4. Define relative significance;
5. Gather expert opinion;
6. Formulate a priori model;
7. Gather and analyze project data;
8. Calibrate a posteriori model; and
9. Gather more data; refine model.

The goal of the model is to provide an estimate of defect density based on V&V certificates and the coverage of each certificate type. 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 currently investigating how this technique can be applied to these two different languages.

5. 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 limitation is the granularity of certificates. Based on the Programatica Team's work [14] it was decided that methods would be the proper level of granularity for certificates. The determination of certificate weights used in the parametric model is still being researched through empirical studies with industry projects.

6. CONCLUSIONS AND FUTURE WORK

We have created and are currently validating a method for managing V&V certificate information. We are also developing a method for a development team to estimate software defect density in-process using this V&V information. Due to the high costs of fixing software defects once a product has reached the field, information that can be provided to developers in-process and can give an indication of software defect density is invaluable. 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 plugin allows developers to easily record their V&V activities within the development environment without increasing their overhead greatly due to the inclusion of Active Certificates. The plugin also provides developers with a mechanism 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. DevCOP certificate information can be used to provide a V&V history for particular code segments. Further, 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 will continue our work to improve the plugin based on developer suggestions and to gather data to validate the DevCOP parametric model.

7. ACKNOWLEDGEMNETS

We wish to give our sincerest thanks to the Programatica team for their input on the various parts of this work. This work was funded by the National Science Foundation.

8. REFERENCES

[1] Balci, O., "Verification, Validation, and Accreditation of Simulation Models," *Winter Simulation Conference*, 1997, pp. 125-141.

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

[3] 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.

[4] Fagan, M., "Design & Code Inspections to Reduce Errors in Program Development," *IBM Systems Journal*, vol. 15, no. 3, pp. 182-211, 1979.

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

[6] Musa, J., "Theory of Software Reliability and its Applications," *IEEE Transactions on Software Engineering*, pp. 312-327, 1975.

[7] Musa, J., *Software Reliability Engineering*: McGraw-Hill, 1998.

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

[9] Rivers, A. T., Vouk, M.A., "Resource-Constrained Non-Operational Testing of Software," *International Symposium on Software Reliability Engineering*, Paderborn, Germany, 1998, pp. 154-163.

[10] Sherriff, M., "Using Verification and Validation Certificates to Estimate Software Defect Density," *Doctoral Symposium, Foundations of Software Engineering*, Lisbon, Portugal, September 6, 2005, 2005.

[11] Sherriff, M., Boehm, B. W., Williams, L., and Nagappan, N., "An Empirical Process for Building and Validating Software Engineering Parametric Models," North Carolina State Univeristy CSC-TR-2005-45, October 19 2005.

[12] Sherriff, M., Nagappan, N., Williams, L., and Vouk, M. A., "Early Estimation of Defect Density Using an In-Process Haskell Metrics Model," *First International Workshop on Advances in Model-Based Software Testing*, St. Louis, MO, May 15-21, 2005.

[13] 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.

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

[15] Williams, L. and Kessler, R., *Pair Programming Illuminated*. Boston: Addison-Wesley, 2002.