SC207 Software Engineering

Review Report: Producing More Reliable Software

Guo Zaiyi (SA1)
Lecturer: Dr. Edmond C. Prakash
School of Computer Engineering
Nanyang Technological University

Abstract

This term paper is prepared for SC207 Software Engineering. The assigned paper is

Producing More Reliable Software: Mature Software Engineering Process vs. State-of-the-Art Technology?

By

James C. Widmaier/Department of Defense
410-854-6951, widmaier@ncsc.mil
Dr. Carol Smidts/University of Maryland Reliability Engineering Program
301-405-7314, csmidts@Glue.umd.edu
Xin Huang
301-405-1071, xhuang@Glue.umd.edu

This review report will discuss on following topics.

- Related topics covered in lectures
- Contribution of this paper
- Elaboration of techniques
- Relevant works
- Relation to lab project
- Possible improvements and extensions
- Comment on notations/diagrams
- Validation and expansion of ideas
- Comments on 5Ps
- Comments on result and relevance to future

Keywords: software reliability, Capability Maturity Model, formal methods, software engineering experiment, software process and product metrics

1 Introduction

A software project is assigned to two software developers and the main paper wishes to find out which approach produces more reliable software. One company is specialized in the state-of-the-practice waterfall method rated at a Capability Maturity Model Level 4. A second developer employed mathematically based formal method with automatic code generation.

2 Review of the paper

2.1 Locate the paper

The paper was searched using google search engine and was found under Kestrel Institute.

Click here to retrieve the soft copy of the paper.

2.2 Related topics covered in lectures

This paper tries to compare the reliability of two processes of developing a software. One of them, the **waterfall model** was discussed in "software process models" module, while the other, **mathematically based formal methods** is not covered during lectures. Formal methods involve creating system specification using mathematical notation (especially discrete mathematics), which is supposed to be more complete, consistent, and unambiguous than those produced using conventional methods.

There are other topics also mentioned in this paper, such as scheduling, project management, documentation, testing, etc.

2.3 Contribution of this paper

This paper explores what kind of approach can produce more reliable software. By examining the strengths and weaknesses of two methodologies, namely state-of-the-practice waterfall method and mathematically based formal methods with automatic code generation, it suggests **a hybrid software development methodology** in order to improve the reliability performance.

It suggests that during requirements analysis, it is better to use formal approach because mathematical specification reduces ambiguities and even spots contradictions in the user requirement. While for process management and documentation, it is better to use conventional approach which is strong in this area, as shown by the experiment.

The paper also discusses some issues during software development, such as inexperienced junior engineers, errors occurring during automatic code generation. Audiences are advised that these issues should be considered early in the project management to avoid defective software development progress and product.

2.4 Elaboration of techniques

The experiment was conducted as follows: it assigned the project to two companies specializing in each methodology. Both companies were given identical functional specs and agreed to a generous and equal cost, schedule, and explicit functional reliability objectives. After the completion of software, a third party reviewed the customer's requirements, refined the model after all faults were removed and develop a test model to test the software. Failure data were gathered for each developed software application and then used to estimate the operational functional reliability.

The way it defined the reliability is simplified comparing to the real situation. The reliability value is the probability of no failure for each gate transaction, i.e., the successful gate transactions divided by the total attempted gate transactions. It defined two levels of failures: level 1 failure was the one which brought the system to a critical state and reliability target was specified to be 0.99 per transaction. Level 2 failure was defined to be less severe but manifested themselves as the system not working properly and the target was 0.9 per transaction.

During software development process, each company used different techniques to aim the development. The company with Waterfall method used Objected Oriented approach, with the help of Rational Rose tool and Unified Modeling Language (UML). The one specialized in formal methods used Haskell language to write executable specifications, with automatic code generation by Specware.

The testing party constructed the test model by using TestMaster, a new test design tool based on the model reference test technology. The test cases were generated automatically using a script generator. The generator develops tests by finding a path through the specified system diagram from the starting to the exit state. Once a path has been defined, the test generator creates a test script for that path by concatenating all of the test action statements and data values required to move the system from its current state to the next state. Once this script is applied to the system under test, the system should follow the sequence defined by the path, if the system's implementation is correct.

To fully automate the entire testing process, Mercury's tool WinRunner was chosen. It generates test cases according to an operational profile, which is the set of operations which users will employ and their probability of occurrence. Thus the failure rate would approximate the real one if the system were put into use.

After collecting the statistical results, a comparison was made between two companies' performance. Several things were compared besides the final software reliability, such as people involved, development activities, etc. The underlying reasons for the result were also discussed, and a conclusion was drawn at the end.

2.5 Relevant works

Author(s)	John D. Musa
Year	1999
Book Title	Software Reliability Engineering: More Reliable Software, Faster Development and Testing
Some Description	It spotlights the practical steps that necessary to apply software reliability engineering to software development and testing. It introduces topics such as developing operational profiles, preparing and executing test, software reliability models, etc.
How does it relate to the main paper	It is a comprehensive publication on software reliability, containing detailed explanation on the techniques of determining the reliability of a software. It helps me to understand how "reliability" is defined and tested in the main paper.
Author(s)	Kamesh Pemmaraju Ed Lord Gary Mcgraw
Year	1999
Article	Software Risk Management: the Importance of Building Quality and Reliability into the Full Development LifeCycle
Some Description	This article discusses the importance of quality and reliability of software. It describes the issues we should take into consideration at different stage of a software development lifecycle to ensure the reliability.
How does it relate to the paper	The main paper concentrates on how the methodology employed affect software reliability, while this article elaborates issues to ensure software reliability at all stages of a development life cycle. It is more general, and gives a good understanding on producing a reliable software.
Author(s)	Michael G.Hinchey, Jonathan P.
Year	1995
Book Title	Applications of Formal Methods
Some Description	It illustrates the application of formal methods to realistic problems, each with an industrial relevance, in various application domains, describing how they can be scaled to large-scale problems, and providing an evaluation of methods, tools, and validation and verification techniques.
How does it relate to the paper	It shows how formal methods are applied in software engineering, how it is currently developing and gives me a general picture of formal methods.
Author(s)	Jonathan P. , Michael G. Hinchey
Year	1995
Article	Seven More Myths of Formal Methods
Some Description	It lists and dispels seven myths about the nature and application of formal methods.
How does it relate to the paper	The "Myths" it discussed, such as formal methods delay the development process; formal methods lack tools; formal methods replace traditional engineering design methods; are all closely related to the main paper.

2.6 Relation to lab project

This paper gives us some idea on improving software reliability by using proper methodologies when developing software. The approach we take in the lab is the waterfall model, which starts from user requirement analysis, followed by design, implementation, integration and testing. According to the paper, this is a state-of-the-practice method, which is strong in process management, documentation, thus the whole process is reproducible. But the less informal approach (i.e. use English rather than mathematical notations to do specifications) may result in incompleteness or errors.

The other approach, namely formal methods, is more advanced and unfamiliar to us, thus utilizing this technique is not practical for this particular lab project. However, it does impress me that analysis is a significant stage of the software development life cycle. A good analysis may be time consuming, but it makes great difference: it eliminates ambiguity, spots incompleteness and contradictions in clients' statements, thus less trouble will occur during subsequent stages. For the future projects, we may consider using formal approach to ensure a thorough analysis. Of course, this means we should spend some effort to master this technique before we can make use of it.

Through this paper, I also learnt that besides UML and Rational Rose, many other design and implementation tools aiming to ease the development process are available. For example, the functional programming language Haskell, which differentiates itself from "imperative" languages such as C++ and Java, is superb for writing executable specifications and suitable for programs which need to be highly modifiable and maintainable. Others such as Specware provides automatic code generation. Although I won't have chance to try these tools for this project, they may be useful in the future assignment.

This paper also raises some issues that we should take care of. One is that junior engineer failed to solve a seemingly easy problem with new Object Oriented tools. This led to unforeseen delay and other troubles. This is exactly what is happening in our lab project. It is our first time to use Rational Rose and a lot of diagrams need to get familiar with, and though we have already written many programs using Java, we are unfamiliar with many useful packages, such as JavaMail and Java3D API. Learning these new things takes a lot of effort, so we are a bit behind the schedule. However, we are more well-equipped after this learning experience, and ready to take more challenges. The most important thing is to be able to learn fast, as rapid development is very normal in computer engineering.

The other is automatic code generation is not always reliable, as shown in the paper. The unanticipated bugs cost the develops' precious time. Automatic generated code also suffers from a code optimization problem. Thus we have to be careful when deciding whether to use such kind of tools.

2.7 Possible improvements and extensions

I feel the following areas in this paper can be improved or extended.

Besides tabulating the experiment results, the detailed comparison of two developments can also be tabulated for the ease of reading and referring. Elements to be put into the table can be possible reasons for not detecting errors in requirements, the tasks scheduled but not completed (especially testing), unexpected issues raised, quality of documentation, weaknesses, etc.

As mentioned in this paper, one of the weakness of the team specialized in waterfall method is that "requirements analysis was not sufficient nor did it involve all 'next step' parties to detect inconsistencies or incompleteness". Consequently "the developed product reflected only what the coder interpreted to be the requirements". While the formal methods team has "continuity throughout the development from one stage to another stage" because the two scientists "did it all". So it is questionable that if formal methods is used by a larger team for analysis, whether it can still keep this continuity, that is, what is done by analyzers is fully reflected during design and implementation stages. Further research can be done to find what is the result of a combination of conventional methods and formal methods; whether it is always good, and what factors will influence the performance (such as management skills, etc).

2.8 Comments on notations/diagrams

There are 3 diagrams in this paper.

Figure 1 (pg 2) is an illustration of the experimental process. It provides an overview of the sequence of the different processes (S-O-A process, S-O-P process, and testing process). It is concise and very helpful to the audience to catch the whole picture of the experiment.

Figure 2 (pg 2) is essentially a flowchart of the Personal Access Control System (the software to be developed by both teams), with some variations. Besides action box and decision box which are common in flowcharts, it also includes two database to show the interaction of the system and external database. It helps the audience to understand the discussion on system reliability in the following paragraphs.

Figure 3 (pg 4) is a model used by TestMaster to test the software. It can be easily seen that the chart is derived from previous flowchart, discarding all the details irrelevant to testing. It consists of the states that a tester interested: if the software functions properly, it should follow the state transitions as desired. This figure is essential such that audience understand the testing process.

2.9 Validation and expansion of ideas

As this experiment is conducted only once, and there are many contingencies (such as people, team dynamics, nature of the software to be developed, software development tools used) which influence the result of the experiment. It is highly possible that one repeats this experiment and gets a different statistics result. This doesn't mean the conclusion of this paper is not trustable. The comprehensive discussion in

this paper illustrated the effect of contingencies before coming up with the conclusion.

Though the result may vary from case to case, the conclusion this paper suggests is worth trying. As indicated in section 2.7, within the conclusion itself, there are many question marks and further research may be carried out on these specific questions. For example, what kind of hybrid software development methodology is the optimum solution, and whether it varies with other factors, such as the maturity of project team, experience level of engineers, nature of software itself, and other constraints? To do this, one may design some software project various in nature (such as of different level of complexity, security requirement, etc), and assign them to a number of project teams, which all use hybrid software development methodology as suggested in this paper, but varying in some factors (such as people, design tools, implementation tools) and collect the statistics of reliability as well as other metrics, to see if can draw some conclusions.

2.10 Comments on 5Ps

The people, product, process, project and platform factors of two approaches are summarized as follows:

	SEI/CMM Level 4 (team 1)	Formal Methods (team 2)
People	1 entry level engineer 3 Senior S/W engineer 1 Process Engineer 1 Q.A. Engineer 1 Program Manager	1 Ph.D. Compute Scientist 1 Computer Scientist
Comments on people	Team 1 had more members, and it has a program manager to do management related activities such as planning, project tracking, peer reviews, etc. The entry level engineer who was unfamiliar with design tools caused some unexpected problem during development. Team 2 comprised only two members, experienced in techniques but relatively weak in project management.	
Process	Waterfall model	Formal methods model
Requirements	83 (hrs)	178 (hrs)
Sys & S/W	211	52
Design	47	283
Implement	36	36
Integration & test	385	8
Total	762	557
Comments on process	Team 1followed the normal waterfall model. Team 2 using formal methods devoted a lot in analysis and design, and very little time for integration and testing. This is possible because they used automatic code generation.	
Product	Reliability = 0.56	Reliability = 0.77
Comments on product	Product objectives and scope were made clear to both team, as stated in this experiment, "they were allowed unlimited access to the customer to understand /refine/ correct the specification".	
Project	This was not elaborated in this paper. However, from some descriptions, we find that team 1 is more well-scheduled and has a better organized management process for the project, while team 2, weak in project management, had to sacrifice their testing time when unexpected bugs occurred.	
Platform	This was not elaborated in this paper.	

2.11 Comments on result and relevance to future

As software systems are becoming more and more common, they will affect out businesses and daily lives deeply. Software failure would be unacceptable as software plays crucial roles in many areas. Hence, how to produce more reliable software is a topic worth thorough discussing.

This paper attempts to examine if state-of-the-art mathematical based formal methods can help in increasing software reliability. The formal methods are to provide a specification language which has a firm mathematical semantics and a development notion which has a clear concept of what needs to be proved for a design (ultimately implementation) to satisfy its specification. Thus it is expected produce better software. This paper got the result that neither of them (conventional waterfall model and formal methods with automatic code generation) could satisfy the reliability requirement. But it did get something worth thinking about.

According to the comparison, formal methods do have advantage in analysis and results show that the process which made use of formal methods had slightly better reliability, and the reliability could have been increased further if the test had been completely conducted. However, other factors, such as the bug in Specware, and poor management process degraded its performance. I would say it is a good point to use formal methods for analysis, but other aspects need improving. The way suggested by this paper is to combine the two methodologies to optimize the performance, and I strongly agree with that.

Formal methods are proven to be worth people's attention. However, making use of it doesn't mean conventional way has to be replaced; Formal methods is not an all-or-nothing approach. A proper combination of conventional methodology and formal methods is of great interest and seemingly that is the right direction to increase software reliability.

3 References

[1] Pressman, Roger S., Software engineering: a practitioner's approach, 5th edition, McGraw Hill, 2001.

[2] Musa J., Software reliability engineering: more reliable software, faster development and testing, McGraw Hill, New York, 1999.

[3] Haskell, The Haskell 98 Report, http://www.haskell.org/

[4] Pemmaraju K., Lord E., McGraw G., Software risk management: the importance of building quality and

reliability into the full development life cycle, http://www.cigital.com

[5] Michael G. Hinchey & Jonathan P.Bowen, Applications of formal methods, Prentice Hall, 1995.

[6] Formal Methods, http://www.afm.sbu.ac.uk/

[7]Langley formal methods, http://shemesh.larc.nasa.gov/fm/

[8] Specware, http://www.specware.org/

[9] Kestrel Institute, http://www.kestrel.edu/home.html