Code Reviews Enhance Software Quality

Richard A. Baker, Jr.
Schneider Automation, Inc.
One High Street
North Andover, MA 01845 USA
+1 508 975 9789
rbaker@modicon.com

ABSTRACT
This paper discusses a procedure for reviewing code changes that are made to a software product as it moves through its life cycle. In this procedure the technical manager reviews each change made to the code. The procedure succeeded in improving in product quality by 40% at a cost of 1-2% of the project effort.

Keywords
Code reviews, software engineering, software quality

INTRODUCTION
In 1989, our team of 6 software engineers and several quality engineers were enhancing a programming environment for Schneider Automation's Modicon line of factory automation equipment. The programming environment, called Modsoft, had certain quality issues that needed to be addressed. Faced with declining customer confidence in our product, the need for rapid improvements in quality was great.

As we searched the literature for a means to improve the development effort, we decided that reviews of the code would provide a cost effective means of improving the quality of the code. But rather than follow the formal code inspections that were written in the literature, we tailored the concept of reviews to our environment. Our methodology included the use of the revision control system to identify all changes made to the product and a line by line review of the changes by the project manager. The results show this was a very effective means of improving product quality.

PRODUCT DESCRIPTION
Our products are predominantly sold in the factory automation and process control markets in applications ranging from petrochemical plants to nuclear power controls to managing discrete logic on an automotive assembly line. This control is performed by our Programmable Logic Controllers (PLCs). The customer's application in the PLCs are developed by using a programming environment on a standard computer system. Modsoft is one such programming environment that runs under DOS.

One unique feature of a factory automation programming environment is that the customer expects to be able to program the PLC live, while the PLC is controlling the factory. This presents a particular issue to Modsoft, as the customer will not have the chance to check that the environment worked correctly before the operation of the factory has changed. As a result, the quality of this product is paramount.

Modsoft has over 1 Megabyte of executable code built from about 500,000 lines of C code. The entire product is maintained under the RCS revision control system on a Sun Sparc server. Most changes to the product are checked into the revision control system as soon as the code can be integrated into the main body of code without impacting other areas of the code. Each change is made with a note explaining why the file has been changed and identifying the problem report or specification that caused the change to be made. All issues found with this product have been carefully archived in a database.

The project consisted of 3-6 software engineers maintaining and enhancing the product from 1989 until the present. The project is lead by a technical manager who spends over 50% of his time developing software for this project, and who has been the one of the top software engineers on the project.

CODE REVIEW METHODOLOGY
The procedure used to review the code is described below. The manager of this project runs a UNIX shell script each week to identify each change that has been made in the

revision control system during the previous week (using "rlog"). With this list, the manager performs an "rcsdiff" on the files that have been changed. Often, the revision comment and the differences are enough to validate the change. Other times the files have to be viewed with an editor to investigate the context of the changes. Any issues are discussed one-on-one with the author of the change. Corrections were based on the consensus between the manager and the author. Any particularly good algorithms or elegant code are also discussed with the engineer.

The manager's code is reviewed by another engineer on the project.

The total cost of this review was in the order of 2-4 hours of reviewing each week, or 1-2% of the project effort.

One change that we tried to make to this methodology involved pairing the engineers and requiring that they check each other's code. This requirement was included on each engineer's annual performance review. However, even with the pressure of a poor performance mark on this section of the review, none of the 6 engineers in this experiment did the required reviews. So we returned to the manager as the reviewer.

DISCUSSION

There were several advantages to using this methodology. The most important was the efficiency with which the quality improvement was made. By using a single reviewer, the number of people used in a review is 1/4 of a formal review. Also, the single reviewer became very efficient in his reviews due to practice at reviewing code and to a thorough understanding of the product.

As a manager, this method provided an invaluable insight into the workings of the department and of the project. Each engineer's performance was thoroughly understood, allowing the manager to comprehend the productivity and the complexity of the assigned tasks. The progress of the project was well understood as each feature was added to the code base. Also, the scheduling skills of the manager were enhanced as the nature of the work and the skills of the engineers was better understood.

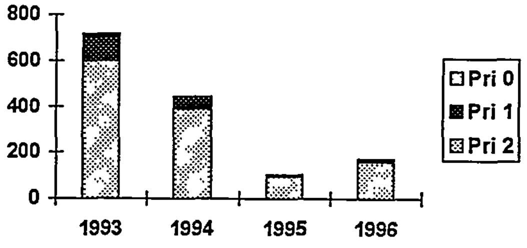
The employees liked this methodology for several reasons. First, they knew that the manager cared about the code that they produced and understood the effort that was required to produce it. Secondly, they understood that their performance reviews were based on their work and not on impressions and politics. Because the manager read the code, he understood their capabilities and could identify areas that needed improvement.

RESULTS

The number of serious problems found with the project have been compiled and show a 40% reduction in the number of bugs found in the testing phase when code reviews were regularly conducted.

Code reviews were regularly performed from 1990 through 1995, and the number of serious problems found in testing steadily declined as the product matured. For various reasons, code reviews were not performed in 1996, and a corresponding increase in the number of problems was seen. Given the increasing maturity of the project, a continuation of the steady decrease of problems from earlier years had been expected.

Serious Issues Found per Year



However, the number of problems reported against the project increased by 65% in 1996.

CONCLUSION

The use of code reviews by a technical manager as a means for monitoring the quality of the code was found to be a very efficient method for maintaining software quality. At a cost of 1-2% of the project, a 40% decrease in the number of issues was found. Potential problems are identified in the reviews and engineers are encouraged to use additional diligence knowing that their manager is reviewing their changes.

ACKNOWLEDGMENTS

This paper would not be possible without the contributions of Rich Cope who developed this methodology in the late 1989. I would also like to thank Joe Finnell, Sue Thorne, Leslie Mannion, Roy Villarreal, and other engineers who suffered through our Software Engineering experiments.

REFERENCES

1. Gilb, Ton and Dorothy Graham, Software Inspections, Addison Wesley, 1993.