

Software quality through distributed code inspection

B.S. Doherty,^a S. Sahibuddin^{b,a}

^a*Department of Computer Science & Applied Mathematics
Aston University, Birmingham B4 7ET, United Kingdom*

^b*Faculty of Computer Science & Information System
University Technology of Malaysia, K.B. 791, 80990 Johor
Bahru, Malaysia*

*E-Mail: bernard_doherty@csqm.aston.ac.uk,
sahibuds@aston.ac.uk*

Abstract

In the software development process, analysing and removing identified errors in the software area provide the basis for improving software quality. In the component implementation and debugging stage, the preceding specification is codified and component testing is done to test the correctness of each component. In this stage, it is desirable to use code inspection, which is a non-execution based testing. Fagan's code inspection is a manual process and must be conducted in a face-to-face situation. The distributed code inspection groupware model will seek to provide a flexible code inspection meeting structure in which the users need not be in the same room at the same time, hence, eliminating the need for a formal meeting. The model was developed with the aim of being flexible and adaptable to changes, making it easier to model the way the user works. The distributed code inspection prototype, which is based on the model, will also be flexible and adaptable to changes.

1.0 Introduction

The software development process is a group activity. Projects ranging from two to fifty people are fairly typical in software development. It is a complex process which requires collaboration of many different types of specialists the software development team may be located on the working together over substantial periods of time and over different geographical areas to create information-intensive several countries. Members of the work group share information, perform

independent task, and create an identifiable, communal product. The capability to co-ordinate a group, working on a single problem is critical.

In the software development process, analysing and removing identified errors in the software area provide the basis for improving software quality. In the component implementation and debugging stage, the preceding specification is codified and component testing is done to test the correctness of each component. In this stage, it is desirable to use code inspection. Code inspection, which is a non-execution based testing, is the review and analysis of source code modules by developers who are knowledgeable in the application domain and programming environment. Fagan's code inspection is a manual process and must be conducted in a formal, face-to-face situation. In the standard code inspection procedure, there are moderators to administer the meeting, readers to read the text, scribes to record any proposed comment, the author of the code and an inspector.

Current code inspection groupware developed by Brothers, *et al.*¹ is limited to face-to-face interaction. Not limiting the system to face-to-face interaction will open up new working possibilities. The distributed code inspection groupware model will seek to provide an informal code inspection meeting structure in which the users need not be in the same room at the same time, eliminating the need for a formal meeting. The model developed in this research is based on object-orientation with the aim of being flexible and adaptable to changes, making it easier to model the way the user works. The distributed code inspection groupware will allow an informal structure to operate. A prototype is being developed based on the model.

2.0 Software Development Process

Fairley² defined Software Engineering (SE) as the technological and managerial discipline concerned with systematic production and maintenance of software products that are developed and modified on time and within cost estimates. Among the primary goals of SE are to improve the quality of software products and to increase the productivity and job satisfaction of software engineers. SE consists of a set of structured methods, procedures, and tools which are used to engineer high quality cost-effective software.

Studies by DeMarco and Lister suggest that on large projects, developers spend 70% of their time working with others, while Jones reports that team activities account for 85% of the cost of large software systems.³ Complex systems require the collaboration of many specialists working together over a period of time to create information-intensive products. Members of the work group share information, perform independent task, and create an identifiable, communal product. The capability to co-ordinate a group working on a single problem is critical. Tools intended to support a work group in the system development process should address issues such as co-ordinating interdependent activities and project management.

3.0 Software Quality

Schulmeyer⁴ stated that analysing and removing identified errors in the software area provide the basis for improving software quality. Some organisations spend half their software quality effort on error tracking and analysis. A software product can be defective in one of four ways : a design defect, a

construction defect, an expressed warranty failure, or a failure to adequately warn the user. The quality of a software product depends on systematic approach. A measurable milestone must be included to verify that the work completed is consistent with the purpose of the program (Figure 1).

The objective of each processing operation in software development is to receive a defined input and to produce a definite output that satisfies a specific set of exit criteria. Unambiguous, explicit, and universally accepted exit criteria would be perfect as process control checkpoints.

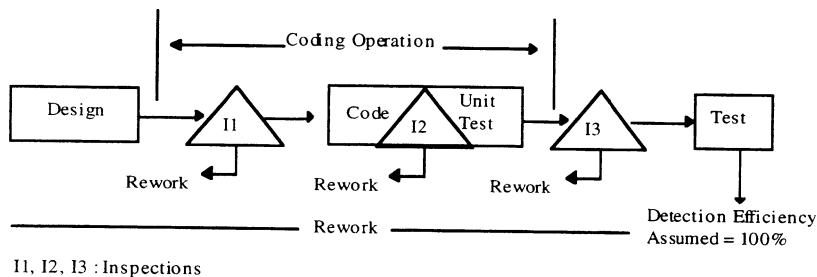


Figure 1 : Fagan's Inspection Model.⁵

There are a number of methods that can be used as an accepted exit criteria. Some of the technique that is available are Formal Technical Review (FTR), Software Inspections and Walkthrough.

3.1 Software Inspection

In Software Inspection, Gilb & Graham⁶ divided the process into two parts; product inspection and process improvement. Product inspections take place between development cycle stages. In this stage, the inspection leader will make the necessary plan and a kick-off meeting is held. The document will be checked individually for defects and a logging meeting is held to consolidate all the issues. The meeting itself is a method to discover more issues and also to identify and log ways of improvement. The log of issues and improvement suggestions will be given to the author. A follow up by the inspection leader is necessary to make sure all the logged issues have been acted upon. In the process improvement stage, the results of product inspections will be use to construct continuous process improvement of the software development process. The process improvement includes the logging of improvements within the previous stage, process change management teams and process brainstorming meeting.⁶

3.2 Formal Technical Reviews

Informal technical reviews take place all the time and are an essential part of real world programming work. But formal reviewing is not essential to programming and many small projects are accomplished without formal reviewing as stated by Freedman & Weinberg.⁷ However, large projects are more complex and requires reliable information in order to manage it, and FTR provides the information needed. FTR is not conducted by people who are part

of that production unit. FTR can be used to check functional specifications, design, documentation, test plan, and of course obviously, the code. FTR consists of one leader, one recorder and a number of other people. The number in the team depends on the material to be reviewed.

3.3 Walkthrough

The term walkthrough has been used in a variety of ways. Nevertheless, walkthroughs are much less formal techniques than Software Inspection. It is typically peer group discussion. Normally, a step-by-step simulation of procedure is done in a form of walking through the code, line by line, with a set of inputs. Walkthrough has been extended to the review of other material such as data descriptions or specifications. Prior preparation is needed for the presenter but not for other participants.

4.0 Code Inspection Process

Code inspection is a phase in the software development cycle, which is the intermediate between the implementation and testing stage. Code inspection is the review and analysis of source code modules by developers who are knowledgeable in the application domain and programming environment. In other words, it is a manual method of error detection. Experiments at IBM (International Business Machine) have shown that code inspections save time and money in the detection of certain types of coding errors before testing. ICL experiments also point to the fact that detailed design inspections improved the quality of code as shown by Kitchenham, Kitchenham & Fellows.⁸ Substantial net improvement in programming quality and productivity have been obtained through the use of formal inspections of design and code.⁵ Code inspection is a manual process. The most time consuming phases are: comment preparation, where code inspectors analyse the module to be inspected, and inspection meeting, where inspectors and author of the code meet to discuss and record the comments prepared (Figure 2). In the standard code inspection procedure, there are moderators to administer the meeting, readers to read the text, scribes to record any proposed comment, the author of the code and an inspector.

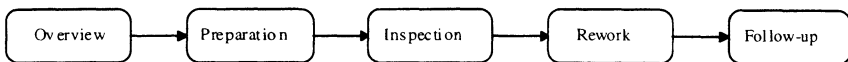


Figure 2: Code Inspection Process.⁵

Brothers *et al.*¹ proposed a system called Intelligent Code Inspection Environment in a C Language Environment (ICICLE) which includes software tools to detect routine sorts of errors, offers various forms of knowledge about the code being inspected and allows code inspectors to traverse source code in a windowed environment. It is designed to render the code inspection meeting paperless. An ICICLE code inspection meeting is intended to occur in one room with all the inspectors facing one another, close enough together for easy conversation. The Collaborative Software Review System (CSRS), proposed by Johnson & Tjahjono⁹, is another tool that address the effectiveness of code inspection. CSRS with an adapted review process model, provides

computational support that decreases the labour intensive nature of review. CSRS also collects a wide range of metrics on the process and products of the review.

5.0 Distributed Code Inspection

On large projects, developers spend 70% of their time working with others, and team activities account for 85% of the cost of large software systems. In a complex system, many specialists collaborate over a period of time and possibly over different geographical areas to create a product. Tools intended to support a work group in the system development process should address issues such as co-ordinating interdependent activities and project management.

In the real world, it is not difficult to find an example of a software project spawning different geographical areas. More and more multinational software companies have a big project broken up into smaller pieces, each in a different country. Novell, in the process of producing the network operating system, Netware, have programmers working in India and in USA, at the same time. An effort to develop FrIJDE (Free Integrated Java Development Environment) on the Web is another example. Programmers working in different geographical areas, work on different aspect of the project.

If there is a possibility that members of the group are working in different geographical areas, then the formal code inspection process needs to be redefined to suit the new criteria. A new model that take this into consideration is needed. In a distributed code inspection process, a tool is needed, as it is likely that manual distributed code inspection process is difficult if not impossible. The tool is a groupware or Computer Supported Cooperative Work (CSCW) system.

5.1 Enabling Technology

5.1.1 Computer Supported Cooperative Work

Wilson¹⁰ defines CSCW as a generic term that combines the understanding of the way people work in groups with the enabling technologies of computer networking and associated hardware, software, services and techniques. Ellis, Gibbs and Rein¹¹ presents two taxonomies for viewing groupware. The first taxonomy is based upon notions of time and space; the second on application-level functionality. These time and space considerations suggest the four categories of groupware.

- Face to face interaction at the same place and at the same time, e.g., meeting room technology.
- The asynchronous interaction at the same place but at different times, e.g., physical bulletin board.
- Synchronous distributed interaction at different places but at the same time, e.g., real-time document editor.
- Asynchronous distributed interaction at different places and at different times, e.g., electronic mail system.

The message system is the most familiar example of a CSCW system. Included in this group are electronic mail, computer conferencing and bulletin board systems. In this kind of system, the asynchronous exchange of textual messages between users occur. Multi-user editors are used to jointly compose

and edit a document. The goal of group decision support systems and electronic meeting rooms is to improve the decision making process by speeding up the process or by improving the quality of the result. Computer conferencing allows the user to do the following; real-time computer conferencing; computer teleconferencing; and desktop conferencing. Intelligent agents are generated automatically by the system. They are responsible for a set of tasks and their actions resemble those of other users. In a coordination system, users will be allowed to view their actions and those of others, within the context of the overall goal. Coordination systems are normally based on these four models; form-oriented model, procedure-oriented model, conversation-oriented model, and communication structure-oriented model.

5.1.2 The World Wide Web

For a CSCW system to support the distributed code inspection process, it must be implemented on some form of network. There are a number of problems if the groupware system is to be implemented in a local area network or in a wide area network from scratch. One is that an enabling infrastructure in terms of development tools have to be set up. The problem of integration with existing environments and migration is also an issue.

The World Wide Web (WWW) provides a great deal of potential. It can be viewed as a platform of integration with existing environment and as an enabling infrastructure for the groupware system. The WWW offers simple and unified access for users to retrieve information or to make it available for others (Figure 4). This approach did not provide support for collaborative information sharing as argued by Bentley *et. al.*¹² With Java, the issue of migration can be tackled, and provides the tools for extending the WWW and integrating it with existing end-user environment.

5.2 Distributed Code Inspection Process Model

In this model, the aim is to provide a technique that is more flexible than Software Inspection but more systematic than walkthrough. The model should be flexible enough such that the code inspection process can be used synchronously *ala* Fagan, or it can be used asynchronously.

5.2.1 Observations

As part of the process of developing the distributed code inspection model, an observation was made on two programming groups working on a small project. The first group is at Aston University, UK and the second group is at the University Technology of Malaysia, Malaysia. A questionnaire was given to get their feedback with regards to code inspection process and distributed code inspection process.

A discussion with an IBM representative at Portsmouth, UK was also held. Concurrently, a questionnaire was also sent to get a better response. The information gathered was then used to develop the model.

5.2.2 The Model

The process will start with the planning on the method to be used. The moderator have to decide between a synchronous and asynchronous technique. In the case of asynchronous method, the process will start with the briefing document being distributed via e-mail. This document will brief the inspectors

about the source code and also the length of the process. After familiarising themselves with the materials, the inspectors can proceed with individual inspections. They can log any recommendations or error findings into the log book. At this stage, any clarification can be sought from the moderator or the author using the facilities provided. The process then continues with group inspection where every member of the team will login at the same time and discuss the issues using the facilities provided. Further recommendations and error findings can be logged at this stage. During the last stage, the moderator will consolidate the log book and make sure that the objective set in the plan is achieved. The recommendation is forwarded to the author or the editor and the process exits.

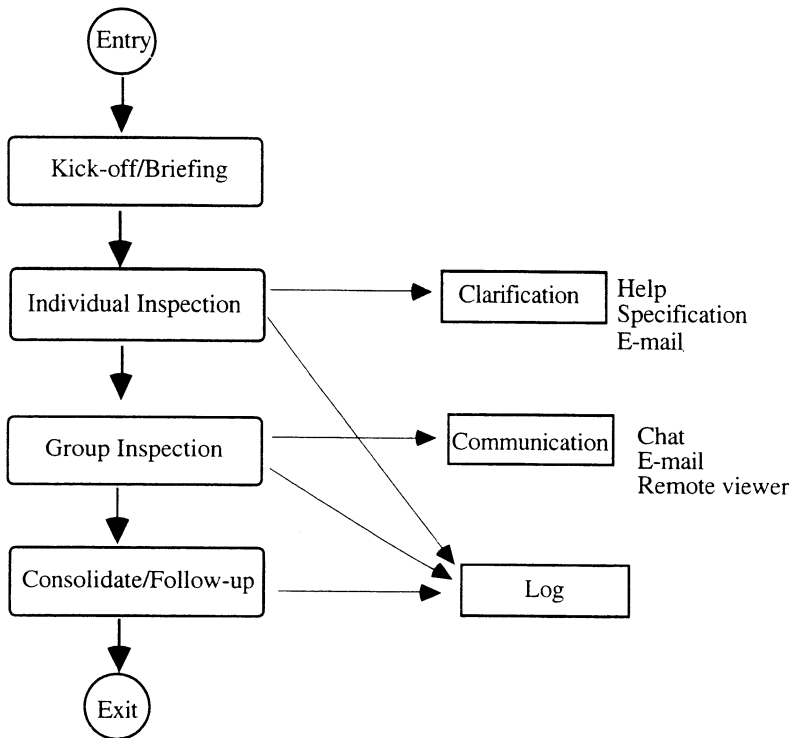


Figure 3 : Distributed Code Inspection Process Model

In a synchronous method, the process will start with a kick-off meeting. This meeting will brief the inspectors about the source code and other objectives of the process. The inspectors can proceed with individual inspections. Again, they can log any recommendations or error findings into the log book. At this stage, any clarification can be sought from the moderator or the author using the facilities provided. The process then continues with group inspection where every member of the team will login at the same time and discuss the issues using the facilities provided. As in the asynchronous method above, further recommendations and error findings can be logged at this stage. The moderator will then consolidate the log book and make sure that the objective set in the plan



is achieved. The recommendation is forwarded to the author or the editor and the process exits.

In this process, the role of the reader and scribe cease to exist because of automation. The role of moderator is also different to the role in a formal meeting set up. The fact that it is unnecessary for the moderator, the inspector or the author to be on-line at the same time, point to the necessity of integrating e-mail into the system so that a communication line can be established. In this model, the length of the inspection is longer than the formal meeting. But the fact that it is flexible and distributed, more than make up that short coming. The consolidation is done by the moderator at the end of the time limit specified by collecting all the inputs from the inspector.

5.3 Distributed Code Inspection Prototype

The CSCW system for distributed code inspection process which runs on top of the WWW is less complicated. By extending the function of WWW using Java, the muddle of thinking about middleware when developing a distributed system is skipped. In this model, the enabling technology and the integration platform is handled by the WWW. On top of this layer is the existing end-user environment, which is integrated by Java (Figure 4).¹³

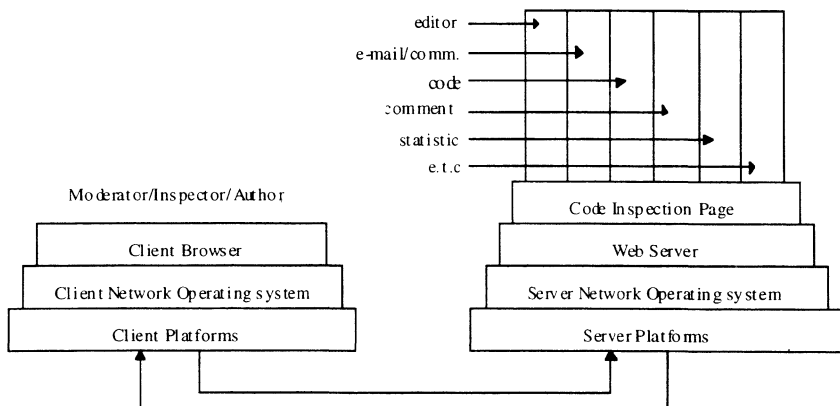


Figure 4: Distributed Code Inspection Architecture

In this prototype, the members of the team will use the code viewer in order to inspect the code. Any recommendation and error finding is entered via the log entry facility. Clarification can be sought by using the e-mail, on-line help and on-line specification. During group inspection, chat and e-mail facilities will provide the necessary mechanism for the discussion. The view remote facility is used when further clarification is needed from the author or the moderator or from other inspectors. This facility will allow the members of the team to view other members' code viewer passively. The log viewer will let members of the team to check the contents of the log book and make any corrections. The log compilation facility is to consolidate the recommendation and error finding.

Two similar type of testing will be conducted onto the prototype by two small programming groups from Aston University, UK and University Technology of Malaysia, Malaysia. The testing will consist of two phases. The first phase will

be a test that involve the execution of a step by step instruction. As for the second phase, it will be regarded as a free mode inspection whereby the groups involved in the testing will be allowed to browse freely, the prototype. After the testing has been carried out onto the two programming group, they will be given a set of questionnaires to respond, in relation to the prototype that they had tested earlier.

6.0 Summary

As a summary, this model of distributed code inspection process provides an informal structure to the code inspection process. The flexibility of this model means that it can be used either in a synchronous or asynchronous way. Both the reader and scribe are no longer required, and the role of moderator, inspector and author have changed slightly. This is to compensate the fact that members of the group is distributed and may not be on-line at the same time. With the flexibility and extendibility provided by this model, the code inspection process should be able to play its role in enhancing the software quality.

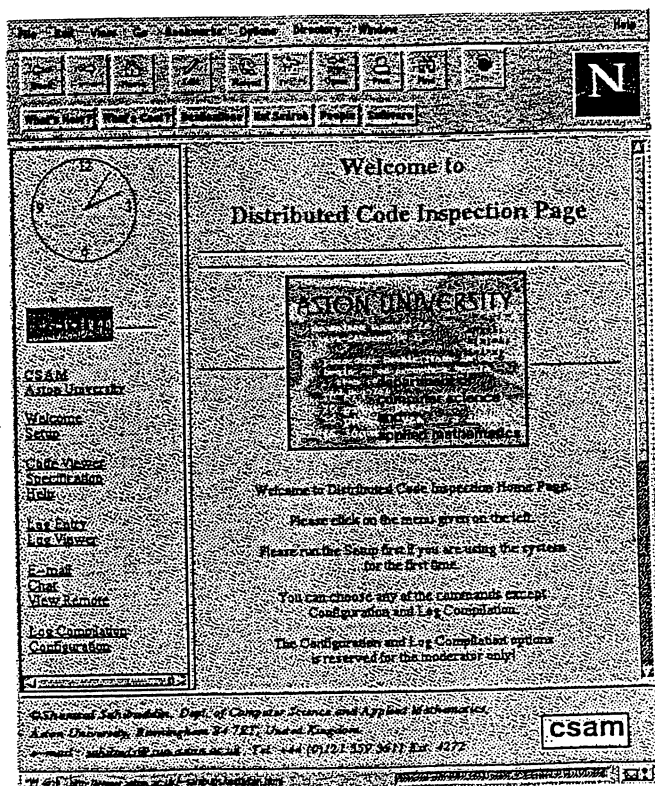


Figure 5: Distributed Code Inspection Page

Key Words

Software Quality, Code Inspection, Computer Supported Cooperative Work, World Wide Web.

References

1. Brothers, L., Sembugamoorthy, V. & Muller, M. "ICICLE: Groupware for Code Inspection", *CSCW 90 Proceedings*, October, ACM, pp. 169-181, 1990.
2. Fairley, R. *Software Engineering Concepts*, New York: McGraw-Hill, 1985.
3. Vessey, I. & Sravanapudi, A.P. "CASE Tools as Collaborative Support Technologies", *Communications of the ACM*, January, 38(1), pp. 83-95, 1995.
4. Schulmeyer, G.G. *Zero Defect Software*, New York: McGraw-Hill, 1990.
5. Fagan, M.E. "Design and Code Inspections to Reduce Errors in Program Development", *IBM System Journal*, IBM, 15(1), pp. 182-211, 1976.
6. Gilb, T. & Graham, D. *Software Inspection*, New York: The Free Press, Macmillan Inc., 1993.
7. Freedman, D.P. & Weinberg, G.M. *Walkthroughs, Inspections, and Technical Reviews*, New York: Dorset House Publishing, 1990.
8. Kitchenham, B.A., Kitchenham, A.P. & Fellows, J.P. *The Effects of Inspections on Software Quality and Productivity*, ICL Technical Journal, May, Staffordshire: ICL, pp. 112-122, 1986.
9. Johnson, P.M. & Tjahjono, D. "Improving Software Quality through Computer Supported Collaborative Review", in Michelis, G.D., Simone, C. & Schmidt, K. (eds.), *ECSCW '93 Proceedings*, Dordrecht: Kluwer Academic Publisher, pp. 61-76, 1993.
10. Wilson, P. *Computer Supported Cooperative Work: An Introduction*, Oxford: Intellect Books, 1991.
11. Ellis, C.A., Gibbs, S.J. & Rein, G.L. "Groupware: Some Issues and Experiences", *Communications of the ACM*, January, 34(1), pp. 39-58, 1991.
12. Bentley, R., et. al. *Supporting Collaborative Information Sharing with the World Wide Web: The BSCW Shared Workspace System*, Schlob Birlinghoven: German National Research Centre for Information Technology, 1995.
13. Doherty, B.S. & Sahibuddin, S. "Modelling Distributed Code Inspection Process", *REDECS '96 Proceeding*, June, Universiti Pertanian Malaysia, Selangor, Malaysia, pp. 69-73, 1996.