Association for Information Systems
AIS Electronic Library (AISeL)

ICIS 2001 Proceedings

International Conference on Information Systems (ICIS)

December 2001

Determinants of Inspection Effectiveness in Software Development: An Empirical Analysis

Sunil Mithas
University of Michigan

Ramanath Subramanyam
University of Michigan

Mayuram Krishnan
University of Michigan

DETERMINANTS OF INSPECTION EFFECTIVENESS IN SOFTWARE DEVELOPMENT: AN EMPIRICAL ANALYSIS

Sunil Mithas
University of Michigan
Business School
smithas@bus.umich.edu

Ramanath Subramanyam
University of Michigan
Business School

M. S. Krishnan
University of Michigan
Business School

Abstract

Software inspections are formal evaluations of the intermediate work products (artifacts) of the development process. These artifacts are examined to ensure that a high quality work-product is delivered to the testers and ultimately to the end-users of the software product. The crucial role of inspections in determining quality of the software makes it important to assess the effectiveness of inspections.

While prior research has identified several factors that influence effectiveness of software inspections, our understanding of the influence of team composition (personnel mix and team size) and the type of the inspected artifact (project plan, requirements specification, design document, code) on effectiveness of inspections is minimal. We develop hypotheses for the factors affecting inspection effectiveness and attempt to validate these hypotheses in a field setting. Our preliminary results show that, during early stages of software development, an increase in the proportion of experienced reviewers (with greater domain experience) is associated with both an increase in the total number of defects discovered in the inspection process as well as an increase in the likelihood of detecting high severity defects. However, during later stages, we find that greater programming experience is associated with both an increase in the total number of defects discovered in the inspection process as well as an increase in the likelihood of detecting high severity defects. These results have important implications for both practice and research.

Keywords: Software inspections, management of IT personnel, software process improvement, project development, systems development methods and tools.

INTRODUCTION

Software is notorious for its defect proneness. Consistent and disciplined processes reduce defects and improve both overall and in-process quality. Software inspections are an important component of a disciplined approach to software development to improve the *in-process* quality. This viewpoint is emphasized in most process maturity frameworks including SEI's Capability Maturity Model, which mandates the use of inspections. Although a successful inspection is argued to improve overall product quality, software managers are often concerned about the cost effectiveness of an inspection process since the inspection process by itself may entail commitment of significant resources (Jones 1997). Hence, there is a need to better understand the determinants of software inspection effectiveness to aid software managers in their resource allocation decisions.

Prior studies on software inspection have addressed *process*, *personnel*, and *product* aspects (Porter et al. 1995; Laitenberger and DeBaud 2000). A major part of this literature tends to focus on *process* aspects such as methodologies and reading techniques to improve the software inspection process and to improve the effectiveness of inspections (Basli 1997; Parnas 1987; Sauer et al. 2000). Complementing the *process* line of literature, some studies focus on *personnel* aspects such as the role of the team members (Russell 1991) and desirable sizes of inspection teams (Madachy et al. 1993; Weller 1993). Research addressing the *product* aspect of inspections (Briand et al. 1998; Parnas 1987) generally tends to focus either on one type of artifact (requirements, design, or code) at a time or links the product and process dimensions. Conspicuously missing in the literature is the validation of the inter-relationship between the process, personnel, and product dimensions. Also there are no studies that account for multiple artifacts simultaneously in the same study.

In this study, we focus on the inter-relationship between effectiveness of inspections (a process dimension), the composition of inspection teams (a personnel dimension) and the type of the artifact (a product dimension).

RESEARCH MODEL AND THEORY

We define effectiveness of inspections in terms of defects elicited in the inspection process of various stages in the project life cycle. The inspection process is performed in a team setting with the goal of the members being identification of as many defects as possible to ensure an end product of high quality. Defects are generally defined as flaws in specification, design, or implementation of software that cause the system to fail to perform its intended function. These defects may creep into software projects in different stages of the development life cycle (Grady 1992).

Prior studies on inspection have used the total number of defects discovered in the inspection process as a measure of inspection effectiveness (Bourgeois 1996; Madachy et al. 1993). However, defect counts alone may not provide a complete picture of the quality of software processes and systems (Harter 2000). This is because defects may vary both in terms of effort involved in fixing them and in their disruptive impact on the user's operational environment (Jones 1997). Further, the earlier the defects are identified in the project life cycle, the greater the overall cost savings (Briand et al. 1998). These cost savings are likely to be greater if the inspection process can effectively identify high severity defects as opposed to low severity defects early in the project life cycle. Thus, in this research, we measure inspection effectiveness in two ways: (1) high severity defects as a proportion of total number of defects elicited in the inspection process and (2) total number of defects.

The conceptual model of our research is presented in Figure 1. We next discuss the theory underlying our conceptual model and hypotheses.

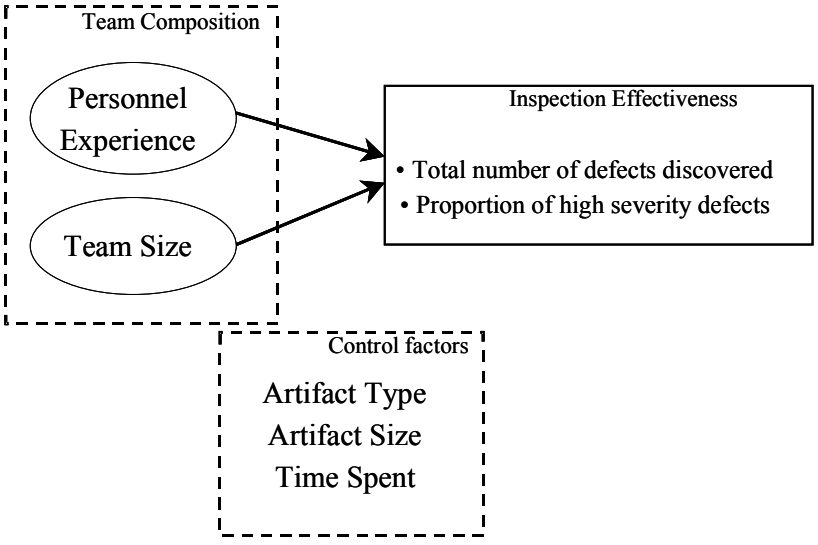

Figure 1. Conceptual Model

Type of the Artifact

Due to differences in the degree of ambiguity of different life-cycle artifacts, the nature of the inspection task may vary. Further, due to these differences in the nature of the inspection task, different personnel skills may be needed for inspecting different artifacts. Accordingly, the effectiveness of the inspection process may vary based on the type of the artifact. More often than not, the type of the artifact corresponds to the life cycle stage in software development. Although there may be more than one type of artifact for each life cycle stage, in this study, we focus only on one characteristic artifact for each life cycle stage. Table 1 describes different life cycle stages in software development and the corresponding types of artifacts (Chaar 1993).

Table 3. Life Cycle Stages and Artifact Type

Life Cycle Stage	Type of the Artifact	Description
Requirements Definition	Requirement specification	The requirements of the project are gathered, analyzed, and described in the *requirement specification*. This document is validated in the inspection process and is later used as a reference to create the software product.
High Level Design	Design specification	The functionality of the product necessary to satisfy the requirements is identified during this stage and the necessary components of the design are described in the *design specification* document, which would be subject to inspection.
Low Level Design	Design structure	The design is broken down into detailed procedures and data structures and documented in a *design structure* document. This is validated against the requirements and design documents during the inspection process.
Implementation	Code	The design structure is transformed into a particular language and the inspection team examines the resulting *code* for defects prior to subjecting it to testing.
Testing	Test plan	Prior to testing of code, a *test plan* is created that describes how the code would be tested during subsequent stages such as unit testing and system testin

Team Composition

Although the benefits of inspections are well recognized by software managers, determining the right team composition (personnel mix and team size) remains a challenge. This is because software managers are often faced with resource and schedule constraints and there are no clear guidelines on how to deploy personnel in the inspection process most effectively during various stages of the project life cycle.

Personnel Mix

Prior studies have suggested that experience and knowledge of the reviewers in an inspection team are important for effectiveness of inspections (Fagan 1986; Strauss and Ebenau 1993). The reviewer experience level may be categorized along two dimensions: *domain* experience and *programming* experience. As software professionals gain more experience, they tend to have greater domain experience compared to programming experience. For instance, after working for a few years in the same domain, such as communication software, a professional may gain proficiency in a particular suite of protocols. This expertise in a particular domain may be useful in inspections, particularly during project planning and requirements analysis.

We believe that an effective inspection requires different levels of domain and programming experience based on the type of the artifact being inspected and the corresponding life cycle stage. During the early stages of development, such as requirements-definition and high level design, the artifacts are at higher levels of abstraction and we expect that the domain knowledge may play an important role at that stage. Hence, an increase in the overall domain knowledge of the team may improve inspection

effectiveness. Conversely, in the later stages of the development, since the artifacts are at lower levels of abstraction and there is a greater focus on implementation aspects, programming experience may determine the success of the inspection process.

H1a: During the early stages of software development (e.g., project planning and requirements specification), an increase in the proportion of experienced personnel (domain experience) would be associated with an increase in the *likelihood of identifying defects of high severity*.

H1b: During the early stages of software development (e.g., project planning and requirements specification), an increase in the proportion of experienced personnel (domain experience) would be associated with an increase in the *total number of defects* identified in the inspection process.

Further, prior research has recognized that inclusion of reviewers external to the development team may improve inspection effectiveness (NASA 1993). We believe that external reviewers (customers and software personnel external to the project team) contribute to the domain knowledge of the inspection team and this knowledge may improve inspection effectiveness. Further, we argue that inclusion of external reviewers may be more beneficial during early stages of software development.

H2a: An increase in the proportion of external reviewers during early stages (e.g., project planning and requirements specification) would be associated with an increase in the likelihood of identifying *defects of high severity*.

H2b: An increase in the proportion of external reviewers during early stages (e.g., project planning and requirements specification) would be associated with an increase in the *total number of defects* discovered.

Team Size

Findings from earlier literature on inspections show mixed results for the influence of team size. While Bisant and Lyle (1989) recommend a small team composed of two people (an author and a reviewer), other researchers (Madachy et al. 1993; Weller 1993) find that a team size ranging from three to five people is relatively more effective in inspections. This underscores the need to better understand the effect of team size on inspection effectiveness.

The mixed results in previous studies could be due to an interaction between personnel experience and team size. For example, during early stages of software development, a large team comprising professionals with more domain experience may be better suited to cope with inspections of relatively ambiguous artifacts. However, in the later stages, a large team comprising professionals with more programming experience may be more effective in inspecting relatively unambiguous artifacts such as code.

H3a: During *early stages* of software development, a larger team with more experienced personnel (who have greater domain experience) would be associated with an increase in the likelihood of identifying *defects of high severity*.

H3b: During *early stages* of software development, a larger team with more experienced personnel (who have greater domain experience) would be associated with an increase in the *total number of defects* discovered.

H4a: During *later stages* of software development, a larger team composed of people with greater programming experience would be associated with an increase in the likelihood of identifying *defects of high severity*.

H4b: During *later stages* of software development, a larger team composed of people with greater programming experience would be associated with an increase in the *total number of defects* discovered.

Measures of Control

In our model, we account for the cumulative time spent by the team in the inspection process. We treat this factor as exogenous since it could be influenced by the software development organization. Further, we explicitly include the size of the artifact as

a control factor in our model since the size of the artifact can significantly influence inspection effectiveness. The unit of measurement for size depends on the type of the artifact. For instance, for source code written in the same language, such as COBOL, the commonly used measure is a standard source line of code (SLOC), whereas for documents describing requirements, design, and test specification, page counts have been traditionally used to represent size (Gilb and Graham 1993). We use appropriate measures of size as used in prior literature.

RESEARCH SITE AND DATA COLLECTION

We use a combination of field study and archival research in collecting data for this research paper. The research site is a subsidiary of a communications software firm, which provides communications-related products and software services to more than 70 OEMs and service providers around the world. The subsidiary is an ISO 9001 certified company and has been assessed at level 4 of SEI's CMM. In order to test our hypotheses, so far, we have collected preliminary data from 93 inspections from software projects over a period of two years. We plan to collect data on additional projects at our research site in order to validate all our hypotheses.

PRELIMINARY ANALYSIS AND IMPLICATIONS

We use linear regressions for analyzing total number of defects and logistic regressions for analyzing the high severity defects as a proportion of the total number of defects (Greene 1999). Our preliminary results show that, during early stages of software development, an increase in the proportion of experienced reviewers (with greater domain experience) is associated with both an increase in the total number of defects discovered in the inspection process as well as an increase in the likelihood of detecting high severity defects. However, during later stages, we find that greater programming experience is associated with both an increase in the total number of defects discovered in the inspection process as well as an increase in the likelihood of detecting high severity defects. There are important implications of this result for managers of software projects. For instance, in the context of personnel deployment, it is known that services of experienced reviewers are not only more expensive but are also scarce. Our preliminary result suggests that it may be more effective, from cost and quality perspectives, to use them in early stages of development rather than in later stages.

We intend to present our research model and significant findings at the conference.

References

Basili, V. "Evolving and Packaging Reading Technologies," *Journal of Systems Software* (38:1), 1997.

Bisant, D. B., and Lyle, J. R. "A Two-Person Inspection Method to Improve Programming Productivity," *IEEE Transactions on Software Engineering* (15:10), 1989, pp. 1294-1304.

Bourgeois, K.V. "Process Insights from a Large-Scale Software Inspections Data Analysis," *Cross Talk*, *Journal of Defense Software Engineering*, 1996, pp. 17-23.

Briand, L., El-Emam, K., Fussbroich, T., and Laitenberger, O. "Using Simulation to Build Inspection Efficiency Benchmarks for Development Projects," in *Proceedings of the Ninth International Conference on Software Engineering*, IEEE Computer Society Press, Los Alamitos, CA, 1998, pp. 340-349.

Chaar, J. K. "In-Process Evaluation for Software Inspection and Test," *IEEE Transactions on Software Engineering* (19:11), 1993, pp. 1055-1070.

Fagan, M. E. "Advances in Software Inspections," *IEEE Transactions on Software Engineering* (12:7), 1986, pp. 744-751.

Gilb, T., and Graham, D. *Software Inspection*, Addison Wesley, Reading, MA, 1993.

Grady, R. B. *Practical Software Metrics for Project Management and Process Improvement*, Prentice Hall, Englewood Cliffs, NJ, 1992.

Greene, W. H. *Econometric Analysis*, 4th ed., Prentice Hall, Englewood Cliffs, NJ, 1999.

Harter, D. E. *The Life Cycle Effects of Software Maturity*, Unpublished Doctoral Dissertation, Carnegie Mellon University, 2000.

Jones, C. *Software Quality: Analysis and Guidelines for Success*, Thomson Computer Press, London, 1997.

Laitenberger, O., and DeBaud, J. M. "An Encompassing Life Cycle Centric Survey of Software Inspection," *Journal of Systems and Software* (50), 2000, pp. 5-31.

Madachy, R., Little, L., and Fan, S. "Analysis of a Successful Inspection Program," in *Proceedings of the 18th Annual NASA Software Engineering Laboratory Workshop*, Goddard Space Flight Center, Greenbelt, MD, 1993, pp. 176-198 (available at http://sel.gsfc.nasa.gov/website/documents/sec02.htm#2.3.18).

NASA. *Software Formal Inspection Guidebook*, Technical Report NASA-GB-A302, National Aeronautics and Space Administration, Office of Safety and Mission Assurance, NASA Headquarters, Washington, DC, 1993.

Parnas, D. L. "Active Design Rreviews: Principles and Practice," *Journal of Systems Software* (7), 1987, pp. 259-265.

Porter, A. A., Siy, H., and Votta, L. G. "A Review of Software Inspections," *Technical Report* CS-TR-3552, University of Maryland, College Park, MD, 1995.

Russell, G. W. "Experience with Inspection in Ultralarge-Scale Developments," *IEEE Software* 8(1), 1991, pp. 25-31.

Sauer, C., Jeffery, D. R, Land., L., and Yetton, P. "The Effectiveness of Software Development Technical Reviews: A Behaviorally Motivated Program of Research," *IEEE Transactions on Software Engineering* (18:1), 2000.

Straus, S. H., and Ebenau, R. G. *Software Inspection Process*, McGraw-Hill, New York, 1993.

Weller, E. F. "Lessons from Three Years of Inspection Data," *IEEE Software* (10:5), 1993, pp. 38-45.