Evaluating Workflow and Process Automation in Wide-Area Software Development

D. E. Perry
Software Production Research Dept
Bell Laboratories
Murray Hill NJ 07974
dep@bell-labs.com

A. Porter*
Computer Science Dept
University of Maryland
College Park, Maryland 20742
aporter@cs.umd.edu

L. G. Votta
Software Production Research Dept
Bell Laboratories
Naperville, Illinois 60566
votta@bell-labs.com

M. W. Wade
Quality Management Group
Lucent Technologies Inc
Naperville, Illinois 60566
michaelwwade@lucent.com

Most software engineering research has focused on improving the quality or reducing the cost of software, but has ignored the need to reduce cycle time (the calendar time needed to develop and distribute a product.) Because a short time-to-market can be a significant advantage in rapidly changing and highly competitive markets, many companies are demanding tools and practices that build quality software, faster.

To help understand the importance of reducing cycle time, consider the software inspection process. Although this is an expensive process, its cost is often justified on the grounds that, since the longer a defect remains in the system the more expensive it is to repair, the cost of finding defects today must be less than the cost of repairing them in the future. However, this argument is incomplete because inspections affect cycle time far more negatively than has been realized.

For example, we have observed that a typical release of the 5ESS switch ($\approx$ .5M lines of added code) can require roughly 1500 inspections, each with four or more participants. For this size feature, our research indicates that inspections alone increase cycle time 10 weeks – from 60 to 70.)

Many people believe that workflow and process automation tools can significantly reduce cycle-time. We share this belief, but have some reservations since our previous research suggests that building quality software products rapidly will require much more than just new technology; it will also require careful analysis of the software processes in which the technology is used.

To study this issue we've developed a workflow tool that allows distributed groups to execute a wide variety of software inspection processes. More importantly, we are using this technology, in a live software development project,

* This work is supported in part by a National Science Foundation Faculty Early Career Development Award, CCR-9501354.

to support controlled experiments exploring how process structure affects cycle time.

In the next Sections we summarize some of our previous research, describe HyperCode, a workflow tool to support software inspection across physically-distributed development groups, present an controlled experiment we are conducting on a live software development project to understand how this technology affects cycle time, and discuss some preliminary observations drawn from our work.

1 Preliminary Research

Several inspection methods have recently been proposed. Although, each method claims to improve effectiveness, their costs haven't been adequately considered. We hypothesized that many of these methods increase effectiveness, but only by significantly increasing inspection cycle time. Therefore, we evaluated this hypothesis with a controlled experiment in a live development project [2].

For each inspection performed during the project, we randomly assigned 3 independent variables: (1) the number of reviewers on each inspection team (1, 2 or 4), (2) the number of teams inspecting the code unit (1 or 2), and (3) the requirement that defects be repaired between the first and second team's inspections. The reviewers for each inspection were randomly selected without replacement from a pool of 11 experienced software developers.

The dependent variables for each inspection included inspection cycle time (elapsed time to completion), total effort, and the observed defect detection rate.

With respect to workflow and process automation, we made two observations.

- workflow technology may reduce several inefficiencies in the inspection process, but
- relatively straightforward changes to the inspection process can have dramatic, negative effects on cycle time.

Since workflow and process automation tools often introduce process changes – sometimes subtly, sometimes overtly – it is essential to understand how different process structures affect cycle time. That is, without rigorous analysis we can't simply assume that these tools will provide the hoped for benefits.

2 Workflow and Process Automation in Wide-Area Software Development

To eliminate defects, many organizations use an iterative, three-step inspection procedure: Preparation, Collection, Repair[1]. First, a team of reviewers reads the artifact, detecting as many defects as possible. Next, these newly discovered defects are collected, usually at a team meeting. They are then sent to the artifact's author for repair. Under some conditions the entire process may be repeated one or more times.

Although conceptually simple, this description hides a number of painful details. To conduct a code inspection at Lucent Technologies, the unit's author must first extract the current version and its design documentation from one of several configuration management systems (even at a single Lucent Technologies site several systems are often used.) Next, reviewers are invited to participate. Once the review team is assembled, individual roles are assigned. For Preparation these roles indicate the technical areas each reviewer should concentrate on. For example, requirements conformance, hardware/software interface consistency, or testability. For the Collection meeting these roles often indicate meeting responsibilities. For instance, the moderator leads the meeting and ensures that all repairs are completed, the reader directs the review of the code unit, and the scribe records the inspection's results.

Next the author distributes inspection materials to all participants and schedules the Collection meeting. Once the reviewers have the inspection materials they can do Preparation. During this phase the reviewers analyze the artifact keeping a record of all issues they uncover. After Preparation the team assembles for the Collection meeting. If the reviewers and/or the author are from different locations, then some participants will have to travel or a conference call will take place (this depends on the complexity and criticality of the code being inspected).

During the Collection meeting all issues discovered by the reviewers during Preparation are discussed and new issues are raised. Each issue that must be fixed is recorded and after the Collection meeting, this list and all repair records are archived to meet ISO-9000 quality reporting requirements.

2.1 HyperCode

Workflow and process automation tools can improve inspections in several ways.

- Encouraging concurrent design and development. When multiple activities or participants must be synchronized or work sequentially, long delays can be inserted into the process.
- Reducing process and paper overhead. When information is stored or manipulated off-line there is often a significant overhead due to document maintenance, retrieval, and dissemination. For example, we observed that a substantial portion of inspection cycle time is spent copying the artifact for each reviewer, preparing lists of the defects found, collating these lists, and entering the results of the inspection into online configuration management system.
- Supporting physically distributed development teams. More than ever before, software systems are developed using third-party vendors and/or geographically separated development teams. Two problems that can occur are that meetings must be scheduled within small, mutually convenient time windows, and that inconsistencies can go unnoticed for long periods because communication is limited.

Consequently, we developed the HyperCode collaborative inspection tool. This system is essentially a set of cgi scripts used in conjunction with the Netscape World Wide Web browser plus an API to two local configuration management systems. The system supports: (1) security, (2) document preparation, (3) process administration, (4) document annotation, and (5) archiving.

Security. Users are provided with web ID's and passwords so that the system can control access to documents.

Document Preparation. When an inspection is necessary the code unit's author uses a browser to select the current version of the code and its documentation from the configuration management system. HyperCode then converts the text to HTML.

Process Administration. The code unit's author uses a point-and-click interface to select reviewers from a pool of available reviewers. He or she can also assign various roles to the reviewers. These roles are later used to guide the system's behavior. For example, the team moderator is the only reviewer who can delete issues. Also, in the repair phase, only the author can "close" an issue. Once participants have been chosen, the system sends email to the participants. E-mail is then used to negotiate attendance and scheduling.

Document Annotation. During Preparation reviewers can check out the inspection package and make annotations to any portion of the artifact. When annotations are made, they are linked into the artifact and are available to the author and other reviewers at that time. Any number of reviewers may access the artifact, from any location within the Lucent Technologies firewall. Reviewers can also comment on the annotations made by other reviewers. Virtual Collection meetings are conducted by asking each reviewer to vote on each issue.

Archiving. Because this Lucent Technologies site adheres to ISO-9000 standards many of the inspection materials must be archived for quality reporting and auditing purposes. Many of these records must be signed by management personnel and archived in paper format. The HyperCode system automatically creates paper copies of the records it maintains on-line. These can be printed out, signed by the appropriate parties and archived.

Although HyperCode duplicates some of the the functionality of several existing systems, it nevertheless provides several unique benefits. (1) Minimal resource consumption. HyperCode runs over Netscape. Consequently, it can be used on a wide variety of machines, requires no special hardware and can be used free of charge. (2) ISO-9000 aware. HyperCode generates all necessary ISO-9000 records. (3) Measurement hooks. Allows us to automatically record inspection activities and recreates much of the data needed in out process experiments. (4) Consistent with the look and feel of the paper process. Users who prefer paper can print all documents.

3 The Experiment

We are currently using HyperCode to support software inspections. Like other workflow tools, HyperCode skirts many of the constraints inherent in the traditional, off-line process.

For example, once an issue is raised all reviewers can see it. With the traditional process each reviewer's issues remain unseen by others until a team meeting is held. Consequently, a large portion of inspection meetings is spent divulging, explaining and discussing each issue. HyperCode allows developers to discuss issues without face-to-face meetings.

3.1 Hypotheses

This is just one example of how workflow and process automation tools can change a software development process, and possibly upset the cost-benefit tradeoffs that justified the process' original structure. With respect to software inspection we hypothesize that HyperCode inspections will differ from traditional inspections in several ways.

- HyperCode inspections will have (1) shorter cycle-time, (2) require no more human effort, (3) incur fewer indirect costs (e.g., travel, conference calls, photocopying), and (4) be no less effective than traditional inspections.
- Inspections with meetings will have (1) longer cycle-time, (2) require more human effort, (3) incur more indirect costs, and (4) be no more effective than meeting-less inspections.
- If the reviewers and the author of a code unit are geographically distributed then, (1) cycle-time, (2) effort, and (3) indirect costs will be greater, and (4) effectiveness will be lower than they will when all inspection participants are co-located.

To evaluate these hypotheses we designed and are conducting a controlled experiment. Our purpose is to compare the tradeoffs between on-line and off-line inspections when different inspection processes are used.

3.2 Experimental Setting

We are currently conducting this experiment at Lucent Technologies on a project that is developing cellular phone services. The development team consists of over 20 persons spread across two sites, one in Illinois (IH) and another in New Jersey (WH). This project, which involves adding several dozen features (new services) to an existing telephony system, began coding in June 1995 and will perform an estimated 200 inspections by Dec 1996.

3.3 Experimental Design

Variables The experiment manipulates three independent variables: (1) the method used for reviewer preparation (HyperCode or traditional), (2) the method used to collect issues (team meeting or deposition), (3) the location of author and reviewers (reviewers can be at IH, WH, or both locations; the author can be at IH or WH).

For each inspection we measure several dependent variables. (1) cycle time, (2) inspection effort (person-hours), (3) support costs (travel, facilities, conference calls, etc.), (4) observed defect detection rate. We also capture repair statistics for each defect.

Design This experiment uses a $2^2 \times 6$ randomized block design. When a new feature is ready to be inspected, the inspection method is chosen on a random basis. Choosing a treatment involves randomly selecting the reviewer preparation and issue collection methods and then randomly selecting reviewers without replacement from the reviewer pool.

4 Summary

The effectiveness of any tool, whether workflow or not depends on the how well it co-exists with the processes it supports. If it is congruent with those processes, then there's a chance that it enhance them. If not then it may be a severe hindrance.

HyperCode automates several aspects of the current manual process and so is *prima facae* consistent with the process it supports. In fact, the initial response from users has so been overwhelmingly positive that it's difficult to constrain the tool's use to fit our experimental design. We attribute this response to two things: first, the elimination of the paper as the medium, and second, the elimination of synchronous interactions.

We also notice, however, scheduling difficulties, physical distance between reviewers, and physical distance between developers who must negotiate the repair of defects appears to affect the benefits of the tool. The exact extent to which this tool works, therefore, must be assessed via our rigorous experiments.

References

1. "IEEE Standard for software reviews and audits". IEEE Std 1028-1988, Soft. Eng. Tech. Comm. of the IEEE Computer Society, 1989.
2. Adam A. Porter and Lawrence G. Votta and Harvey P. Siy and Carol A. Toman. "An Experiment to Assess the Cost-Benefits of Code Inspections in Large Scale Software Development". *The Third International Symposium on the Foundations of Software Engineering"* Washington, D.C., October 1995.

This article was processed using the LaTeX macro package with LLNCS style