Does Every Inspection Need a Meeting?

Lawrence G. Votta Jr.
AT&T Bell Laboratories
Naperville, Illinois 60566

ABSTRACT

At each step in large software development, reviewers carry out inspections to detect faults. These inspections are usually followed by a meeting to collect the faults that have been discovered. However, we have found that these inspection meetings are not as beneficial as managers and developers think they are. Even worse, they cost much more in terms of product development interval and developer's time than anyone realizes.

Analysis of the inspection and collection process leads us to make the following suggestions. First, at the least, the number of participants required at each inspection meeting should be minimized. Second, we propose two alternative fault collection methods, either of which would eliminate the inspection meetings altogether: (a) collect faults by deposition (small face-to-face meetings of two or three persons), or (b) collect faults using verbal or written media (telephone, electronic mail, or notes).

We believe that such a change in procedure would increase efficiency by reducing production times without sacrificing product quality.

1. Introduction

Inspections are a part of the engineering process. An engineer's work is repeatedly examined by other engineers to discover faults remaining in the design. For each component, there are usually several inspections, each on a different aspect, such as design, code, and test plans.

The problem we address is not in the inspections themselves, but in how we execute the inspection process. Many development organizations use a three step procedure: inspection, collection, and repair. It is common practice to complete the collection step by holding a meeting of all reviewers (see Figure 1—the collection meeting is held at the end of $T_{collection}$).

Meetings are now considered to be central to inspections, and we find that a whole folklore has evolved to justify the use of inspection meetings far beyond the original purpose of collecting faults. Further, it is now assumed that it is only within the context of collection meetings that inspections can effectively be done.

However, our research raises a question about the efficacy of these collection meetings and we suggest alternatives that should be tested.

1.1 Motivation

Lest anyone think that the question of whether inspections are best performed by holding collection meetings is not important, consider that our recent studies suggest 20% of the requirements and design development interval is spent just waiting for reviewers to meet [1]. Figure 1 is a time diagram of the inspection, collection, and repair intervals with typical times observed for design inspections performed during a large software development project.

To understand what had already been done in evaluating the efficacy of reviews and the means of measuring them, we searched the literature since 1970. Nowhere in the published software engineering literature has anyone considered the increase in product interval as a cost factor. (Humphrey [2] gives an excellent summary of inspection cost models, pp. 184-188.) Only recently has this cost been recognized as significant for most product development enterprises [3].

1.2 Applicability

We believe that the study of inspections presented here is generally applicable to almost any software development. We support this claim by observing that much of the literature describing how inspections should be done, etc., comes from the same development organization that generated our data. For instance, Humphrey's descriptions of data collection and his examples of the forms used to collect inspection data are from this development organization ([2], pp. 171*ff*).

1.3 Road Map

Sections 2 through 4.2 present the motivation for finding and describing alternatives to inspection meetings. After describing the common reasons why inspection meetings are held, we present the costs and benefits of inspection meetings, propose several alternatives, and recommend a move toward a different collection mechanism. Finally, we describe some other needs that the collection meetings satisfy, and which must be considered if an alternative method is to replace them successfully.

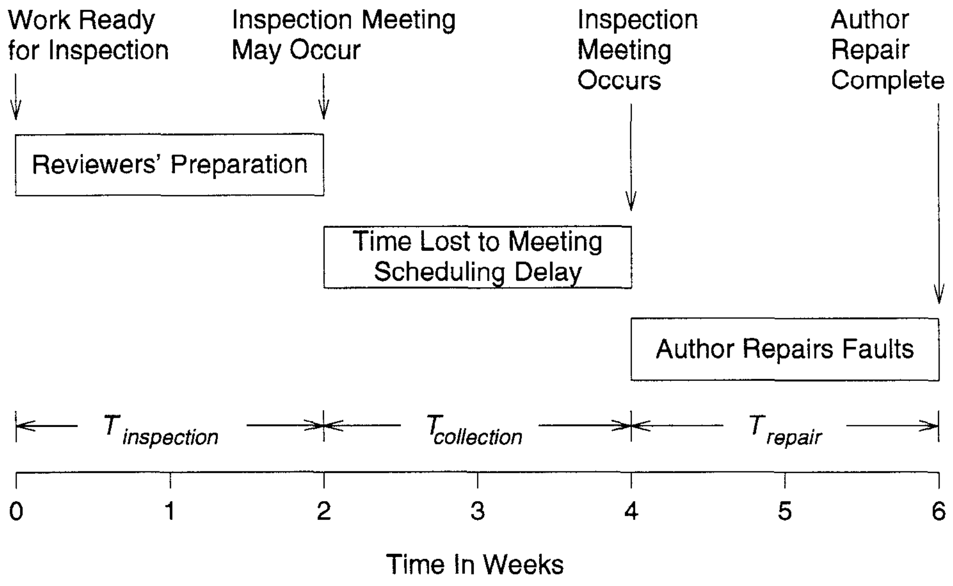

Figure 1. Typical Time Line for Inspection Process
This time line depicts the different steps in the execution of an inspection process in a large software development. The author finishes the work (requirements, design, code) and then must make it available for $T_{inspection}$ working days, giving reviewers time to inspect it. The moderator anticipates the completion of the inspection interval and schedules the meeting that finishes the collection step. $T_{collection}$ is the number of working days lost between the end of the inspection step and the collection meeting. The scheduling delay depends on the number of reviewers required at the meeting and the calendar of each reviewer. After the inspection meeting, the author needs T_{repair} days to repair the faults. Typical values in a large software development for a feature of 600 NCSL are 10, 10, and 10 working days for $T_{inspection}$, $T_{collection}$, and T_{repair} respectively.

2. Why Hold Inspection Meetings?

The current thinking concerning inspections originates from Weinberg's concept of egoless programming [4]. He observed that like anyone else, programmers have the tendency to overlook evidence that conflicts with their beliefs. He referred to this as *cognitive dissonance.* Inspections were suggested as a way to efficiently minimize this effect.

2.1 What Do Authors of Design Documents and Their Managers Believe?

From May through December of 1991, we observed several inspection meetings. After the repair step, we collected data on each fault from the author. As part of this interview, the author was asked to list all reasons for holding the inspection meeting. We then coded the responses as described below. Usually, after the inspection was completed, we would interview each author's direct supervisor as well. Their responses were similar to those of the authors.

The top five reasons for the meetings, given in descending order of frequency (see Table 1), were as follows:

1. Synergy — the interaction of reviewers in the group setting creates a team that finds many faults during the meeting that were not discovered by individual reviewers in their preparations.
2. Education — less experienced reviewers learn from experienced reviewers how the inspection process works, how the development process works, and how to be a good reviewer.
3. Schedule Deadline — the inspection creates a planned event for people to work towards.
4. Competition — the group setting provides an arena where people publicly contribute and earn the esteem of their peers, and hence strive to improve themselves.
5. Requirement — the document inspection process specifies that a meeting be held to collect reviewer comments.

This simple data set provides much food for thought. Managers and developers agreed on the three most frequently given reasons for holding a reviewers' meeting at the end of the collection stage.

The perception of both managers and developers is that significant increases in fault identification occur in the meetings because of synergy. But does this really happen? Surprisingly, we found it does not, but since synergy was the most frequently given response, we defer the discussion of it until the next section, after we have considered the other four reasons.

Demming argues that education by observation and participation is not effective ([5], pp. 52*ff*). Actually, he puts it much more bluntly when he teaches his class—an effective paraphrase is "the blind leading the blind." Furthermore, inspection courses already exist, both inside and outside of AT&T Bell Laboratories, that are known to be more cost effective than in-process training [2], [6].

As for competition, Demming argues that management invariably starts trying to quantify the successful competitors (most design faults found, etc.) and rewarding them. This management behavior destroys teamwork, for instance between designers and testers ([5], pp. 62*ff*).

Reason	Author (18)	Manager (11)	Combined (29)
Synergy	78% (14)	82% (9)	79% (23)
Education	61% (11)	64% (7)	62% (18)
Schedule Deadline	50% (9)	45% (5)	48% (14)
Competition	22% (4)	55% (6)	34% (10)
Requirement	17% (3)	45% (5)	28% (8)

Table 1 Survey of Reasons Given for Using Meetings as the Primary Means of Collecting the Results of Inspections

The table shows the percentages and the number (in parenthesis) of authors and managers who thought inspections were done for the reasons listed. Both authors' and managers' responses were coded from answers to the question, *"Why do you think inspection meetings are useful?"* All other reasons given occurred with a frequency of less than 20%. The bold numbers in parenthesis behind the column headings are the total numbers of people in the survey.

Inspection meetings do have some beneficial characteristics that we would want to preserve. They provide a schedule deadline and a well defined process. Both are known to be helpful for project and time management ([7], pp. 67*ff*).

2.2 Measured Cost and Benefits of Review Meeting Not Considered in the Literature

Fagan's work [8] in 1976 established the basic cost benefits of inspection for improving software quality. His research indicated that it is less expensive to detect and repair a fault at the step where it was inserted than to try detecting and removing the resulting failures during software testing. The costs of testing, isolating, repairing, and retesting the software later were far greater than the cost of employing any reasonable number of reviewers to inspect it immediately after it was written. He described inspection experiments using meetings to collect the reviewers' feedback. This was the genesis of the inspection meeting.

Several authors throughout the 1980's suggested various refinements in how inspections should conducted. Parnas and Weiss in 1985 introduced the notion of an active design review [10]. Their many proposals included reducing the number of people in collection meetings to a minimum. Their rationale was that in a meeting of n people, only 2 can be interacting, while $n-2$ can only listen, and so many of the $n-2$ are probably wasting their time.

This waste of reviewers' time is not the only cost associated with time. We have found another significant cost in the delay that results from preparing, scheduling, and performing the inspection feedback with a meeting (see Figure 1). Further, although synergy is firmly believed to be the major justification for holding inspection meetings, experimental evidence shows that little or no synergy actually occurs. However, we did find that fault collection meetings do seem to minimize the number of "false positives" (faults which the author later finds require no repair).

It seems that using meetings to inspect the software product involves a complex cost-benefit trade off with few guidelines that are true in all cases. The solution is in understanding the costs and benefits for each local development process.

3. Cost Benefit Analysis

A change cannot be considered an improvement unless the gains realized are sufficiently greater than any losses incurred. In the present context, the gains and losses are measured in terms of time, money, and product quality. Therefore, before proceeding, we must make certian that the benefits of the changes we propose would outweigh the costs and that quality would not be compromised.

3.1 Synergy

Simply put, *synergy* refers to the belief that the inspection team in a meeting will outperform any individual or the sum of individual team members working alone. The *Subarctic Survival Situation* exercise [12] dramatically shows this effect. (A group will outperform any individual unless the individual is an arctic survival expert.) In terms of software inspections, survival is the equivalent of never allowing a fault to reach the next step of the development process. The belief is that in many cases teamwork can substitute for expertise (our most highly sought after resource in a software development).

If this effect occurred in the software inspection process, we would see many inspection faults found only by holding a meeting. If we labeled all faults found in preparation (before the meeting), then the remaining faults discovered (during the meeting) must be attributed to *synergy*. We made this measurement as part of an earlier study which was aimed at understanding how well capture-recapture sampling techniques estimated remaining design faults after an inspection. Figure 2 displays our data (from experiments described in [13]).

Why is the effect so small? We believe *synergy* is not happening. To understand why it is not, we need to understand what synergy is, and what effects it can and cannot have.

Theoretically, a team uses three types of skills to make decisions and solve problems: interpersonal skills (how team members communicate with each other), rational skills (how the team reasons about situations), and task skills (how the team can best divide tasks) [13]. Synergy occurs as individual team members adjust their roles to exploit their stronger skills and compensate for their weaker skills. They accomplish this by using the team skills. However, teams do not always have to use all three team skills for excellent synergy to occur.

A good example of synergy when only two of the three team skills are employed is an exercise like the *Subarctic Survival Situation*. There the strong focus of a crisis, and each team member's relative ignorance of the problem domain (which prevents the team's task skill from being important), forces the team to adjust roles quickly to best use each team member's skills. Most teams will display some degree of synergy (i.e., the degree to which the team's choices are more correct for subarctic survival than are any individual team member's choices) because the team's communication skills allow the piecing together of individual team member's knowledge, while the rational skills help prioritize and assemble this knowledge.

Now consider our case with inspection collection meetings. They are extremely disciplined meetings with well defined roles. This inhibits the team skills from adapting to individual team member's strengths. The moderator controls the pace so the

Date	Author	Foundation	Contributions
1971	Weinberg [4]	Experiments	1. Why do reviews?
1976	Fagan [8]	Experiments	1. First cost benefit analysis. 2. How to conduct reviews; defined roles. 3. Two hour review rule.
1979	Yourdon [9]	Personal Experience	1. No managers at inspections.
1985	Parnas [10]	Personal Experience	1. Active design review. 2. Smaller inspection meetings.
1986	Fowler [6]	Experiments	1. Summary of current metrics and forms.
1988	IEEE [11]	Personal Experience	1. Standardize definitions of reviews.

Table 2 Milestones in the Literature on Inspections
A brief summary of the major contributions to the practice of inspection meetings for software development. Many others have also contributed to the practice and understanding of the review process.

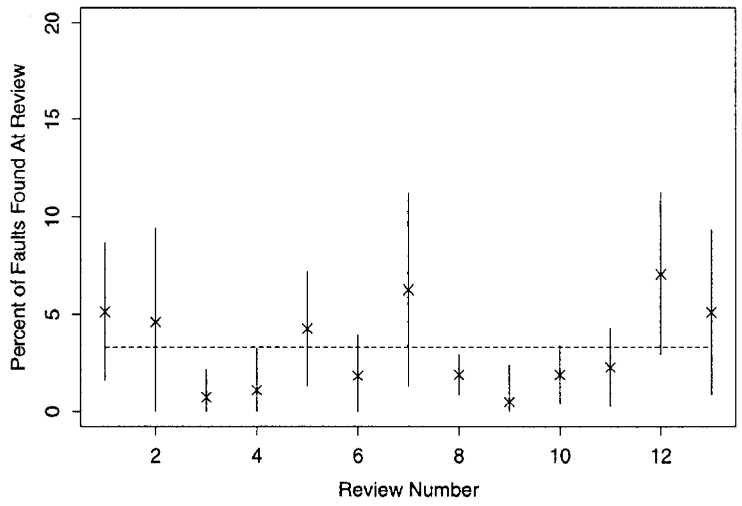

Figure 2 Measured Synergy for Design Inspections
Each point represents the synergy rate for a particular collection meeting, i.e., the number of faults which went undetected by reviewers in their preparation (before the meeting held to collect the inspection results), divided by the total number of faults recorded at the meeting. This rate is marked with an ×. The vertical line segment through each × marks one standard deviation in the estimate of the rate (assuming each fault was a Bernoulli trial). Thus the more faults used for the rate estimate, the shorter the line segment, and hence, the more precise the estimate of the rate. This provides information as to the significance of any one rate measurement. The average synergy rate is about 4% (the dashed line) for these 13 inspections.

group completes the work in the allotted time. However, many times the goal becomes just that—to finish in the allotted time—not to discover and record the most faults. Therefore, even though at first glance it appears that synergy is occurring because time pressures are present, the extreme discipline of the meeting, coupled with the question of goals, can significantly inhibit the proper adjustment between team skills and individual skills which is crucial for the synergy effect to occur.

Is the benefit of ≈4% increase in faults found at the collection meeting (for whatever reason) worth the cost of $T_{collection}$ and the reviewers' extra time? The answer is no. This can be illustrated by comparing the cost-benefits of a meeting based method versus a deposition based method of inspection.

The cost of the time interval, $T_{collection}$, is composed of (1) the hazard cost of being later to market, (2) the carrying cost of the development when the author is not working on the product, and (3) the rework cost resulting from the author continuing work on the product only to have the work invalidated by a fault found in the inspection.

Market window costs, although real and in many cases significant, are hard to estimate, and for our analysis we will assume that they are 0.

Detailed calculations for the development organization which we observed showed collection meetings involve 2 weeks more waiting time than do deposition meetings, assuming 1 day to collect depositions.[1] This extra interval generates a cost difference of $\$1.6K$ per inspection in favor of depositions. Collection cost factors (2) and (3) generate this advantage. Although, it is possible that collection meetings may find as

1. Is one day reasonable to perform all the depositions? Yes. Our initial experiments demonstrate that 5 reviewers can easily have their meetings with the author and moderator the day after the preparation interval $T_{inspection}$ ends.

Authors' Repair Time	Authors' Repair Action		
	No Change	Fixed Local	Fixed Global
>1 Hour	none	3%	3%
<1 Hour	1%	92%	1%

Table 3 Authors' Repair Times and Actions for Design Inspection Faults
This table gives the authors' repair times and actions for 616 suspected faults recorded during 13 inspection meetings. Note the few inspection faults that were in the *No Change* category. We suspect the meeting team makes good decisions and removes reported faults that are not really faults. The key question is how many real faults go unrecorded and are not found until later in the development?

many as 1.6 more faults than the depositions, this potential benefit of collection meetings does not offset their higher cost (*\$.9K* per inspection). Therefore, deposition meetings are more cost effective than collection meetings.

3.2 False Positives

For the 13 reviews in Figure 2, Table 3 shows the author repair metrics of repair time crossed with repair locality. The repair time is measured in units of one hour. For our arguments we need no better resolution; therefore, we have summarized the data into less than and greater than one hour. The locality measures how the repair was classed:

1. *No Change* - the author found no repair action was necessary.
2. *Fixed Local* - the author repaired the design without any other design component being involved.
3. *Fixed Global* - the author needed to coordinate the repair of the design with other designers and their designs.

A quick scan of Table 3 shows that only 1% of all faults recorded at 13 inspection meetings were *No Change*. Now, the important question is what will happen if we remove the inspection collection meeting: will the work load of the author go up too much because the *No Change* rate increases? The answer is no. Also important is whether the inspection meeting is screening out too many identified faults that are then not recorded because of an erroneous decision by the team. Our notes of these meetings show that 9 out of every 10 faults discussed were recorded. This observation limits the occurrence of failures to record a real fault to a maximum of 11% of the total number of faults recorded at the meeting. Reviewing the list of faults not recorded, we observe that 2/3 of the 11% are a restatement of faults already recorded, or are some other obvious *No Change*. This reduces the 11% to 4%. Thus the inspection meeting fails to record only 4% of the total number of faults found either in preparation or at the meeting.

The cost of letting a fault pass on to a later development step is known to be about 10 developer hours. Therefore, the maximum cost of missing some faults because of failure to record them is 10 developer hours times 4% of β (the number of faults recorded at the review), or $.4\beta$.

On the benefit side (reduction of false positives) we have at most 1 developer hour (see Table 3) saved for every false positive that the author does not have to check (i.e., *No Change*). Thus the maximum saving is 1 developer hour times $.11 - .04$ times β *No Change* faults recorded at the inspection collection meeting,

It would be cost effective to use the inspection meeting to screen faults only if the effort needed to repair a fault, times the number of faults let through, is less than the effort needed to handle a *No Change* times the number of *No Changes*. But, since $.4\times\beta$ developer hours is greater than $.07\times\beta$ developer hours, it is *not* cost effective to use the group meeting to reduce the number of false positives.

Further, the high cost of subsequently fixing an identified inspection fault that went unrecorded leads us to the firm conclusion that **no comments or suspected faults should ever be allowed to go unrecorded in a design inspection.**

3.3 Schedule Latency

Figure 1 depicts the time consumed by an inspection process using a fault collection meeting. We have found that the inspection meetings account for $\approx$10% of the development interval. The calculation is simple. These software products have a development interval of from 60 to 90 working weeks. They undergo 4 inspections that require collection meetings. With 2 weeks loss per meeting due to scheduling contention ($T_{collection}$), we can understand that the impact of the current inspection process on the development interval is to increase it unnecessarily by as much as 13%.

If there is any doubt that these times are significant, Stalk shows that elapsed time (development interval) is important for financial and marketing reasons [3]. The longer money is borrowed, the longer it takes to realize a profit. The longer the development interval, the slower the response to customer needs and changing market conditions—and possibly, the less satisfied the customer, and the less the product is perceived to be a *quality product.*

How do we make $T_{collection}$ shorter? We could reduce the number of meetings on each reviewer's calendar, and/or the number of reviewers required at the meeting, and/or the duration of the meeting. This is intuitively easy to see if we think of each reviewer's calendar as composed of 4 two hour periods each day. Let p denote the probability that a two hour slot is occupied by a meeting and is equal to the average number of meetings per day, $\bar{m}$, divided by 4 ($p = \bar{m}/4$). Let q be the probability that a two hour slot is not occupied, where $q = 1 - p$.

Now let's ask when a meeting could occur among the author, a moderator, a recorder, and 5 reviewers (a total of 8 people)? If all have the same p and are independent, then the probability of any two hour slot being available is simply the product of the probabilities of each of the 8 people being available, $\left[1 - \frac{\bar{m}}{4}\right]^8$. The average number of time periods that will elapse before the meeting occurs is $\frac{1}{4}\left\{\left[1 - \frac{\bar{m}}{4}\right]^{-8} - 1\right\}$.

Using the plot in Figure 3 the reader can deduce the remedies we are suggesting to shorten the waiting time ($T_{collection}$) for a meeting. For instance, reducing the number of meetings on everyone's calendar means reducing $\bar{m}$. This is seen by

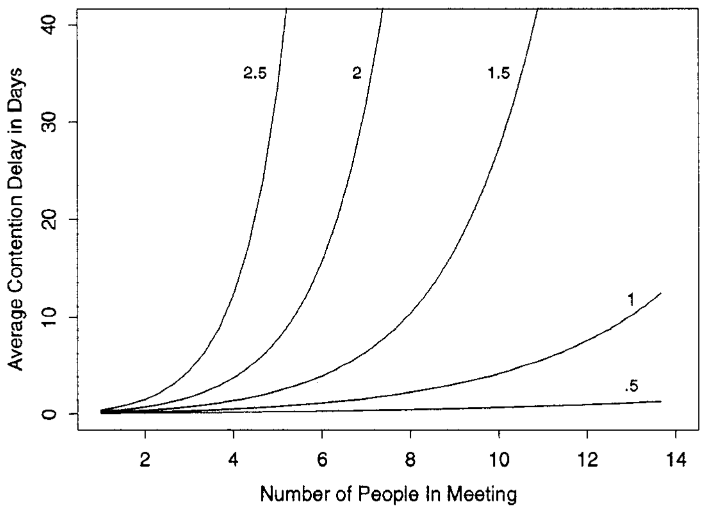

Figure 3 Waiting Time in Days for Meeting
The average waiting time in days is plotted as a function of the number of people at the meeting and the average number of meetings in a day ($\bar{m}$). The different curves correspond to different values of $\bar{m}$ indicated near the end of each curve. All people are assumed to have the same value of $\bar{m}$. The plot demonstrates why the suggestions we are making in section 3.3 will reduce the average waiting time for a meeting. For instance, if $\bar{m} = 1.5$, reducing the number of people from 8 to 6 reduces the average contention delay from 11 to 4 days.

observing that when $\bar{m} = 1.5$, roughly 11 days are needed before the meeting can be held. Changing $\bar{m}$ to 1 meeting per day, we find only 2.2 days of average contention delay.

3.4 People's Time

In a meeting of n developers (typically 1 author, 1 moderator, 1 recorder, and $n-3$ reviewers), usually only 2 of the developers interact at any time. In our observations of inspection meetings, we tried to keep track of what the $n-2$ other developers were doing when 2 developers were interacting. We learned two things:

1. 30% to 80% of the $n-2$ other developers appeared to be listening at any given time.[2]
2. The percentage of $n-2$ other developers listening usually decreased during the meeting, being highest at the beginning and lowest toward the end.[3]

We can safely say the amount of unproductive time is in the range of $n-4$ (the moderator and recorder usually pay attention to the two who are interacting), times the average inspection meeting length of 2 hours, times an estimated attention span correction of .5 to account for time when everyone is listening, such as at the beginning and the end of the meetings and during discussions. Therefore, the average inspection meeting *wastes* $n-4$ developer hours per meeting. This number is particularly important when contrasted with the much smaller time an author and moderator would spend in taking depositions.

3. We measured *attention* at inspection collection meetings of the reviewers by observing individual reviewers. We interpreted two actions as inattentiveness: (1) the reviewer was obviously not paying attention (dozing, side conversations, etc.) or (2) the reviewer was reading different material not related to the task at hand. The percentage is arrived at by taking the ratio of time spent not paying attention to the total elapsed time of the meeting.

3. The percentage of developers listening increased during arguments, disagreements, and other unusual events. Listening would also increase during the final 10 minutes.

4. Alternatives

Inspection meetings are held to collect reviewers' findings and comments. Further, meetings constitute milestones for reviewers to plan towards. However, there costs can exceed their benefits.

So what alternatives to inspection collection meetings exist? We visualize a range of options, from managed meetings to depositions to non-meeting reporting methods.

4.1 Managed Meetings

Managed Meetings have well defined agendas, specific goals and completion criteria, and a moderator who directs the group's attention and controls the pace of business (for example, see [14]). These techniques are described in several excellent references, and software development inspection meetings already borrow heavily from them.

Observations of many inspection meetings has convinced us that current practice is already structured and generally well run. Seldom did we see an inspection meeting wander for any more than 30 to 60 seconds before the moderator would refocus the discussion on the inspection.

The combination of the moderator and recorder proved efficient. Almost every fault was successfully identified, stated, understood, and recorded in about 90 seconds.

Our suggestions for improvement focus on timeliness, culture, and the number of required reviewers. Most reviewers

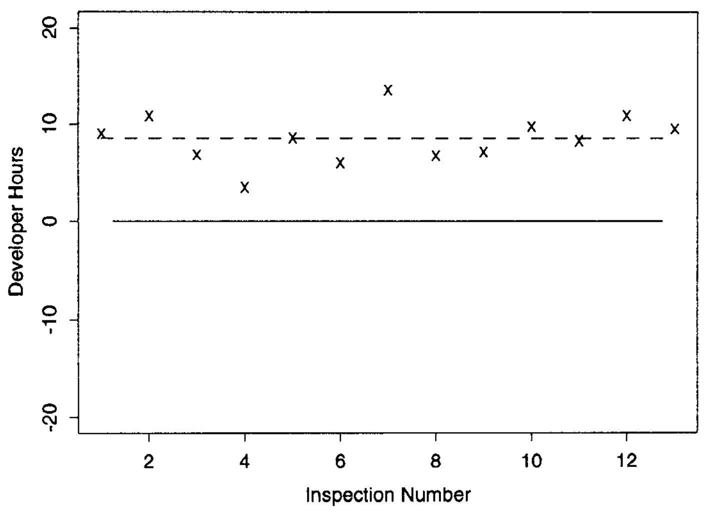

Figure 4 Developer Hour Cost Differences for Design Inspections
For 13 design inspections, we have plotted the difference in developer hours between an inspection meeting and a deposition method of collecting faults. The solid line indicates the break even point between meetings and deposition methods. Above this line, depositions are more economical; below this line, meetings are more economical. All inspections would have required fewer developer hours if depositions had been used. The average savings would have been about 8.5 developer hours (marked by the broken line).

would be at the inspection meeting on time. However, there were always one or two reviewers who were several minutes late. The meeting could not start until all reviewers were present, and therefore, about 10 minutes was wasted at the beginning of each meeting. Methods to encourage timeliness may be appropriate. Further, a culture change is needed to discourage the acceptance of "late comers" and the resulting lax attitude towards timeliness. Finally, the number of required reviewers at inspection meeting should be minimized.

4.2 Depositions

A *deposition* is a three person meeting where the author, the moderator, and a reviewer collect the reviewers' findings and comments. The moderator schedules and runs the meeting, and performs the role of the recorder (an additional saving since for an inspection meeting these roles are performed by two separate people). Depositions shift the workload from the reviewers to the author and moderator. The author and moderator need to attend more meetings, but the total number of developer hours used for depositions would be less than for an inspection meeting.

An easy way to see that this is true is to use a simple model and some typical inspection data. To do this we make four assumptions:

1. reviewers' preparation times are the same for both a deposition and an inspection meeting;
2. the scheduling effort for depositions is the same as would be necessary to schedule a single inspection meeting (observing the scheduling method of the moderators indicates this is true);
3. the duration of a deposition is equal to the sum of a fixed overhead time plus the time needed to report all the faults observed by each reviewer (which we assume to be the fraction of faults found multiplied by the duration of the inspection meeting);
4. the extra time required in all depositions to record duplicate faults is about the same amount of time as required in the collection meeting to discuss faults found by multiple reviewers (i.e., depositions suffer no penalty).

The developer hours spent in an inspection meeting equals 2 hours multiplied by the total number of reviewers, k, plus 3 (the author, moderator, and recorder).

$$E_{\textit{inspection meetings}} = 2(k+3) \ .$$

If we let k be the number of reviewers, a_d the overhead time of starting and stopping a deposition ($a_d = 10$ minutes for our calculations), p_i the fraction of faults found by the i^{th} reviewer, and if we assume 2 hours for the length of the inspection meeting, we can calculate the developer hours spent for a deposition as

$$E_{\textit{depositions}} = 3ka_d + 2 \cdot 3 \sum_{i=1}^{k} p_i \ .$$

In Figure 4, we have plotted the difference, $E_{\textit{inspection meetings}} - E_{\textit{depositions}}$ for the 13 inspections. When $E_{\textit{inspection meetings}} - E_{\textit{depositions}} > 0$, then inspection meetings are taking more effort than depositions and depositions are more efficient. When $E_{\textit{inspection meetings}} - E_{\textit{depositions}} < 0$ then depositions are taking more effort than inspection meetings and inspection meetings are more efficient.

The data shows that depositions are almost always more efficient than inspection meetings.[4]

4. Do depositions always win? No. If the number of reviewers is greater than 20 this model breaks down because the cost of getting the inspection results collected in a timely manner ceases to be negligible.

4.3 Correspondence

Correspondence includes forms of communication where the reviewer and author never actually meet. Communication can be verbal, handwritten, electronic, etc. Although this form of inspection fault collection would create the least overhead, many important forms of feedback would not be available to the authors, reviewers, or moderator.

5. Conclusions

Our analysis leads us to make the following recommendations.

1. Almost all inspection meetings requiring all reviewers to be present should be replaced with *Depositions*—3 person meetings (author, moderator, and one reviewer). Scheduling difficulty increases nonlinearly with the number of participants. Dropping the requirement for a moderator, creating 2 person meetings, would be an even greater improvement.
2. Training courses teaching inspection techniques should be readily available and conducted frequently. Inspection meetings provide a dubious education at best, and they should not be relied on to perform this function.

The benefits would be much lower overhead and greatly reduced inspection cycle time, with little or no sacrifice of quality. The inspection training course would replace any loss of the education now potentially gained in the inspection meetings. If there is concern that a training course would be of less value than the "hands-on" experience, we could include apprentice software developers in the deposition meetings as observers. (Because apprentices are new to the development organization, their schedules are not demanding, so including them in the deposition meetings would add little to the scheduling contention).

Should inspection feedback be collected using correspondence? Not without further study. Well monitored, controlled experiments must be done to demonstrate whether the benefit to the authors of having no meetings to attend is greater than the potential cost from lack of fault recording clarity, etc. Also we have not yet discussed the problems of process, of how project management would audit the existence of the inspection with this method, etc.

Finally, it is worthwhile to continue to explore inspection techniques with experiments generating real data that aid in answering these questions:

1. What is the optimal number of reviewers?
2. Are inspection checklists effective?
3. Are reviewers with specific roles more effective than general reviewers?
4. Can we implement a *just in time* inspection policy using depositions?

In this work, we have melded empirical data with our analysis to demonstrate how these two interact to provide insight and direction for future work. The use of empirical data in process analysis and improvement is essential. Without data it is impossible to understand which problems are important to study and solve and which are not—it is impossible to do engineering.

Acknowledgements:

I would like to thank the developers and managers in the 5ESS development organizations who took part in the different capture-recapture review experiments. My colleagues Dave Long, Dewayne Perry, Adam Porter, Nancy Staudenmayer, and Eric Sumner provided numerous valuable comments and suggestions. Arthur Caso of Associated Consultants provided invaluable aid in editing several versions of this paper.

References:

[1] M. G. Bradac, D. E. Perry, L. G. Votta, "Prototyping a Process Experiment," *Fifteenth International Conference on Software Engineering*, Baltimore, May 1993.

[2] W.S. Humphery, *Managing the Software Process,* Reading, Massachusetts: Addison-Wesley Publishing Co., 1989.

[3] G. Stalk, Jr. and T. M. Hout, *Competing Against Time: How Time-Based Competition Is Reshaping Global Markets,* Free Press, New York, 1990.

[4] G. Weinberg, *The Psychology of Computer Programming,* Van Nostrand, 1971.

[5] W. E. Demming, *Out of the Crisis,*Massachusetts Institute of Technology, Center for advanced Engineering Study, Cambridge, Mass., 1982.

[6] P. J. Fowler, "In-Process Inspections of Workproducts at AT&T," *AT&T Technical Journal,* Volume 65, Issue 2, March/April 1986.

[7] J.F. Keane, M. Keane, and M. Teagan, *Productivity Management in the Development of Computer Applications*, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1984.

[8] M.E. Fagan, "Design and Code Inspections to Reduce Errors in Program Development," *IBM Syst. J.*, vol. 15, no. 3, 1976, pp. 182-211.

[9] E. Yourdon, *Structured Walkthroughs,* Second Edition. Englewood Cliffs, NJ: Prentice-Hall, 1979.

[10] D. L. Parnas and D. M. Weiss, "Active Design Reviews: Principles and Practices," *Eighth International Conference on Software Engineering,* London, August 1985.

[11] *IEEE Standard for Software Reviews and Audits, IEEE Std 1028-1988,* Soft. Eng. Tech. Comm. of the IEEE Computer Society, June 1989.

[12] C. Lafferty, *The Subarctic Survival Situation,* Human Synergistics, Plymouth, Michigan, June 1975.

[13] S. G. Eick, C. R. Loader, M. D. Long, L. G. Votta, and S. Vander Weil, "Estimating Software Design Faults," *Fourteenth International Conference on Software Engineering*, Melbourne, May 1992.

[14] B.Y. Auger, *How to Run Better Business Meetings,* Amacom Visual Products, 1972.