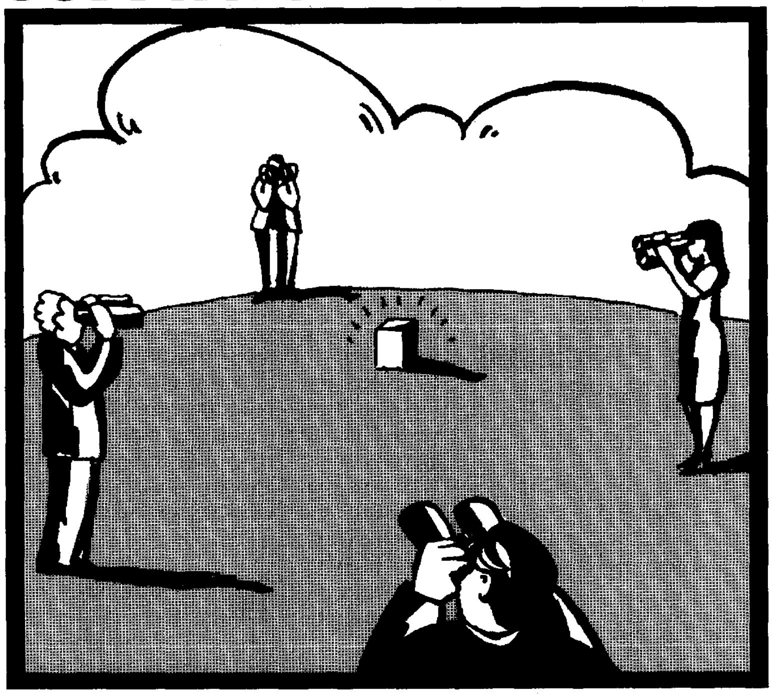
FEATURE

DISTRIBUTED, COLLABORATIVE SOFTWARE INSPECTION

An inspection system that supports a structured meeting model lets participants work from separate locations and provides easy-to-maintain on-line inspection material.

VAHID MASHAYEKHI
JANET M. DRAKE
WEI-TEK TSAI
JOHN RIEDL
University of Minnesota at Minneapolis



Inspection — a detailed review of a small amount of material by technically competent peers — is an effective method for detecting faults in software documents and code. Barry Boehm included inspection in his list of the 10 most important approaches for improving software quality because, according to his research, it catches 60 percent of the faults.[1] Not only do peers working together find more faults, they also find *more serious* faults than the software's producer alone can find.[2]

Inspection normally involves four to six people, so it is costly. Besides setting aside time for the meeting, participants must travel to the meeting site. Each inspection covers only a small portion of the product, so it takes many meetings to completely inspect a software product. These logistics can make inspection a bottleneck in the software-engineering process.

Despite its problems and expense, inspection has proved cost-effective, because participants uncover faults before they propagate to the next phase of the life cycle. The cost-effectiveness of inspection would be improved even further by a distributed, collaborative meeting environment that eliminates the need for face-to-face meetings.

In this article, we present an inspection environment that lets geographically distributed inspection participants "meet" with people in other cities through workstations at their desks. The current version of all material is accessible on-line. Inspection products are created on-line, so secondary data entry to permanent records is not necessary. The inspection information is also available for review and metrics collection.

Our environment's automated support adds structure and consistency to the inspection process, helping participants achieve the consistent and uniform review that the Software Engineering Institute and Watts Humphrey[3] consider crucial for developing a high-quality software process. Enforcing structure results in a repeatable process and gives measurable results.

But how will inspection participants react to working in such an environment? We designed a case study to compare inspections in our distributive collaborative environment with face-to-face meetings. The results show that meetings using our environment are as effective as face-to-face meetings and that electronic support helps in fault correlation, especially by eliminating paper shuffling. On-line inspection material is easier to maintain than hard copy and is adequate for the inspection meetings, but not quite as usable as hard copy for individual code inspection before meetings, according to our survey of study participants.

Our Collaborative Software Inspection project differs from general meeting tools because it is specifically structured for software inspection and facilitates effective use of team resources for fault finding and analysis. CSI differs from the Icicle code-inspection tool under development at Bellcore,[4] which focuses on artificial-intelligence support for fault detection. Our tool provides a distributed, structured environment for performing inspections on all software-development products, including specifications, designs, code, and test cases.

INSPECTION PROCESS

Both Edward Yourdon[5] and Humphrey[3] developed widely used techniques for inspection. In both approaches, team members have specific roles, prepare individually for the inspection, attend the inspection meeting, and find faults that result in action items.

The preparation stage differs in the two techniques. In Yourdon's technique, the reviewers read the target material and informally note faults and concerns. Reviewers are also encouraged to note positive aspects of the target material. In Humphrey's technique, each reviewer creates a fault list and gives the list to the producer of the material before the meeting. The producer correlates the faults and prepares to address the faults in the inspection meeting. Humphrey's model also adds an optional introductory meeting during which participants review background material and inspection criteria.

CSI lets the inspection team use either Humphrey's or Yourdon's inspection technique, but in our case study we follow Humphrey's because it is more structured and provides intermediate results through individual and correlated fault lists.

Inspection meeting model. In Humphrey's model, the inspection team is a group of peers with the technical knowledge required for detailed inspection. The quantity of target material addressed in one inspection is small because of the detailed level of review.

The participants have specific roles: producer, moderator, recorder, and reviewer. The producer is the author of the target material. The moderator organizes the meeting and keeps it on track. The recorder makes the action-item list and status report. The reviewers raise questions and identify faults in the target material.

The inspection is divided into four phases: initialization (planning), preparation, meeting, and postinspection. Figure 1 shows the phases of inspection and the activities that occur in each phase.

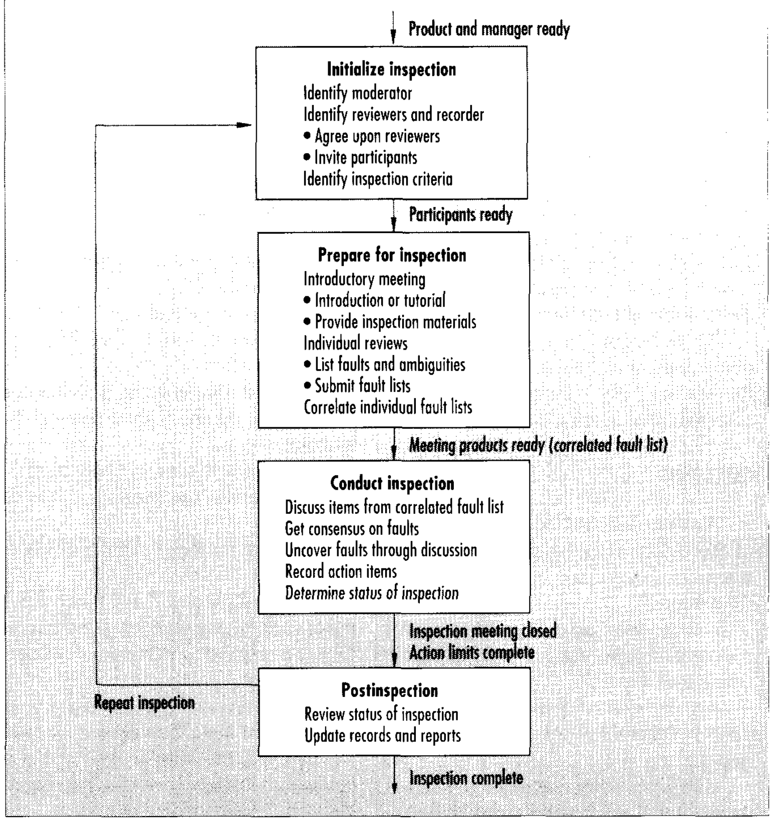

Product and manager ready
Initialize inspection
Identify moderator
Identify reviewers and recorder
• Agree upon reviewers
• Invite participants
Identify inspection criteria
Participants ready
Prepare for inspection
Introductory meeting
• Introduction or tutorial
• Provide inspection materials
Individual reviews
• List faults and ambiguities
• Submit fault lists
Correlate individual fault lists
Meeting products ready (correlated fault list)
Conduct inspection
Discuss items from correlated fault list
Get consensus on faults
Uncover faults through discussion
Record action items
Determine status of inspection
Inspection meeting closed
Action limits complete
Repeat inspection
Postinspection
Review status of inspection
Update records and reports
Inspection complete

Figure 1. Inspection process.

	Material						Participants				Type of interaction	
Process	Target material	Inspection criteria	Inspection status	Fault lists	Correlated faults	Action items	Moderator	Producer	Reviewers	Recorder	Synchronous	Asynchronous
Initialize inspection	S	S	C				X	X				X
Introductory meeting	R	R	U				X	X	X	X	X	
Individual review	R	R	U	C					X			X
Correlate faults	R	A	U	R	C			X				X
Inspection meeting	R	A	U	A	R	C	X	X	X	X	X	
Postinspection	R	A	U	A	A	A	X	X			X	X

Key: Process use of material:
A=Available C=Create S=Set up R=Read U=Update

Figure 2. Inspection activities and material.

Inspection material. Material used in inspection includes the target material, the inspection-criteria list, individual fault lists, the merged fault list, the action-item list, and the status report. Faults are recorded on the action-item list for subsequent correction by the producer. The target material may be a product of analysis, design, or coding. For analysis and design, the target material may be a dataflow diagram, an entity-relationship model, a state-transition diagram, an object model, or a text specification.

The goal of inspection is to find faults, not correct them. Therefore, participants only read the target material; they never update it. They refer to the inspection-criteria list to help them identify faults. During the inspection, the participants create fault lists, an action-item list, and a status report. The reviewers use the report forms for their initial fault lists.

Figure 2 shows, for each phase, the activities that occur, material required, use of the material, participants in the activity, and type of interaction during the activity.

CSI DESIGN

We designed the CSI tool on the basis of four kinds of collaborative inspection meetings, categorized by space and time dimensions:[6]

♦ *Same time, same place.* Participants meet in a single meeting room and interact simultaneously.

♦ *Same time, different place.* Participants in different rooms interact simultaneously.

♦ *Different time, same place.* Participants go to a single meeting room at different times of their own choosing.

♦ *Different time, different place.* Participants in different rooms perform inspection tasks at times that each chooses.

The first two meeting types correspond to synchronous inspection activities, such as the discussion and categorization of faults and the generation of action items. The second two correspond to asynchronous activities such as the participants' individual reviews of the target material to discover and categorize faults and the producer's correlation of the lists into a single list.

CSI tasks. We categorized CSI tasks according to the phases of software inspection in which they occur:

1. Distribute target material (initialization).
2. Individual reviews of the target material (preparation).
3. Correlate the faults revealed during individual reviews into a single list (preparation).
4. Discuss the faults in an integrated fault list during the formal meeting (inspection).
5. Record action items (inspection).
6. Determine the meeting status (inspection).
7. Record meeting results (postinspection).

Tasks 1, 2, 3, and 7 can be asynchronous; tasks 4, 5, and 6 are synchronous. The premeeting, asynchronous tasks provide products for the meeting. Most important are the fault lists produced by the individual reviews. Participants use some materials in both synchronous and asynchronous activities — for example, the target material and inspection criteria.

CSI supports both asynchronous and synchronous inspection phases. It reduces the overall workload in the initialization phase because participants perform tasks only once. For example, the producer loads the target material once, the recorder enters the action-item list once, and the reviewers type faults once.

Asynchronous activities. The two main asynchronous activities of software inspection are the individual reviews and the producer's correlation of faults. The process starts with the producer making the target material available to the reviewers. Once the target document is available, the reviewers browse through it, annotating it appropriately.

CSI supports annotations by creating hyperlinks between lines of the document and the reviewers' annotations. Later, the producer reviews and categorizes the annotations and makes them into one list. During the meeting, participants bring the integrated fault list on-line and discuss the faults. Participants can add more comments synchronously during the meeting.

Synchronous activities. The synchronous activities of inspection include discussion of correlated faults, reaching a consensus on the faults, recording the action items, and determining the inspection's status. CSI supports discussion among the participants with a teleconferencing tool named Teleconf.[7] CSI brings the target material on-line in a window on all participants' screens.

Figure 3 shows schematically what the user sees during the synchronous part of software inspection. The producer leads the group through the correlated fault list. The producer orders the list as he desires (for example, by severity of fault or by sequential order in the target material). Participants can also examine the annotations at this stage to better understand the faults.

The recorder lists the results of these discussions as action items. Each action item describes the fault and gives its location and a severity rating. This process continues until participants address all the faults on the fault list.

On the basis of fault quantity and severity, the participants arrive at a consensus on the meeting's status. The recorder enters the status in the inspection summary, which also records the team members, their roles, the attendance, and the date and time of the meeting.

CSI IMPLEMENTATION

To build CSI, we used the Suite software-development environment.[8] Suite supports persistent data by check-pointing to a disk. An object whose data structures are on a disk is passive. It becomes active when an instance of it is instantiated and the persistent data structures are loaded from the disk.

Suite's user interface treats all objects as editable data. Dialogue managers display a presentation of an object. The user can edit the presentation, and the dialogue manager communicates changes to the object. The object, in turn, ensures that the dialogue managers at other sites display consistent data. A dialogue-manager compiler compiles a Suite program, generating object editors that permit interaction between the user and the object.

Figure 4 shows a distributed system for software inspection. The persistent data is on the disks (Browser and Note Pad). Each object has persistent data at only one site, while the dialogue managers are at each site where the object is active.

Components. Eight objects make up CSI: Browser, Annotation, Note Pad, Action List, Inspection Summary, Criteria, Fault List, and History Log. Each object contains a specific piece of information used in inspection and provides the access the inspection team requires:

♦ The Browser displays the target material and supports hyperlinks from the target material to the Fault List, Note Pad, Inspection Summary, and Action List, as Figure 3 shows. A line number precedes each line of text in the target material. When the user selects a line number, an annotation window pops up that lets the user record comments about that line.

♦ The Annotation lets users record faults during their individual reviews. It offers guidelines to the reviewers in categorizing and sorting faults. The sorting capability indicates to the producer the reviewer's opinion about the faults' importance.

♦ The Fault List lets the producer group the faults found by the reviewers during the individual reviews into a single list. It automatically adds summaries of the individual faults to the integrated fault list. The Fault List helps the producer categorize, accept or reject, and sort

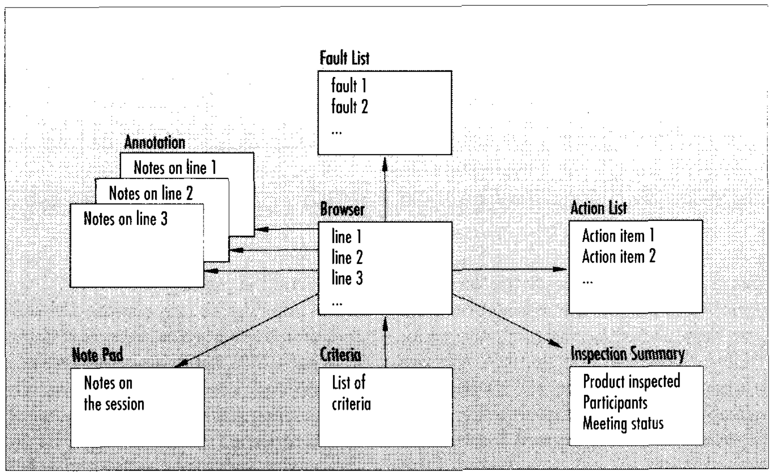


Figure 3. Synchronous activities of software inspection.

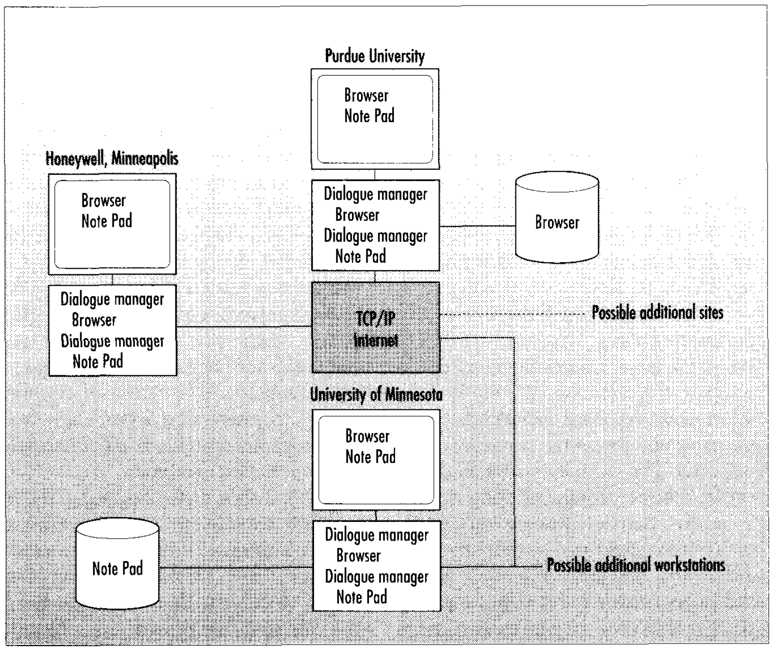


Figure 4. The function of dialogue managers in distributed collaborative inspection.

TABLE 1
CSI OBJECTS AND THEIR USE IN INSPECTION

Object	Use in inspection	Type of access
Browser	Target material	Read only
Annotation	Extended fault description	Read and write
Fault List	Correlated fault list	Correlation
Note Pad	Individual-review fault list	Read and write
Action List	Final fault list	Read and write
Inspection Summary	Status of inspection	Read and write
Criteria	Inspection criteria	Read only
History Log	Tracking of CSI activities	Background monitor

faults for presentation at the inspection meeting. The producer can sort on multiple keys, including category, disposition, time of creation, line number, and producer-specific groupings. CSI places the accepted faults at the top of the list and the rejected faults at the bottom. However, rejected faults can still be discussed during the inspection meeting. Reviewers can negotiate with the producer to upgrade a fault's status to an accepted state.

♦ The Note Pad lets participants record comments that do not pertain to a single line of target material. For example, some comments may span several lines of the target material, cover the entire target material, or deal with missing material.

♦ The Action List represents the result of the inspection meeting — a detailed list of faults sorted by severity, disposition, and line number — to reduce the work required in postinspection cleanup.

♦ The Inspection Summary records the meeting status. The decision may be to accept the target material as is, reject it, or accept it with certain modifications. Other information recorded includes the module name and the names and roles of participants. The primary writer is the recorder, who records the meeting status.

♦ The Criteria presents on-line the evaluation criteria used by the participants during the individual reviews to identify and categorize the faults. Criteria is a supporting object for the convenience of system users. In the pilot study, we used the criteria developed by Daniel Freedman and Gerald Weinberg.[2]

♦ The History Log records CSI activities and, for our study, compiled data to evaluate CSI's performance versus that of a traditional meeting.The History Log time-stamps actions performed by all objects and writes them to a log. The data collected includes the number and severity of faults, the time taken to find a fault, and the length of the software-inspection meeting.

Degree of collaboration. CSI uses two Suite coupling modes, one for asynchronous work and the other for synchronous work.[8] In asynchronous work, updates are propagated only after a user commits a change to an object. Thus, two users entering notes into the Note Pad do not see each other's changes until the notes are completed. In synchronous work, CSI propagates updates immediately after each character is typed. Thus, if a participant is typing an action item, other users can watch his progress, perhaps suggesting changes. CSI's match of coupling modes to interaction styles helps users work together and eliminates unnecessary communication overhead.

Table 1 shows the objects and the inspection material and type of access each object supports. Figure 5 shows the Browser, which contains a portion of the design document that was inspected in the case study we present next. This design-document segment contains 254 lines of text.

CASE STUDY

To assess the effectiveness of inspection in our distributive collaborative environment and compare it with face-to-face meetings, we used a case-study approach with replication logic.[9] (Such a case study is repeated by different groups, and we look for the same phenomena in multiple cases to confirm our observations.) Although the software-inspection process is well-defined and the CSI tool provides a well-defined environment, we have no control over how inspection teams behave during the inspection process. Participants use their knowledge of the problem and their intuition to find faults. Unlike in experimental research, we used random rather than statistical sampling.

Environment. We used a network of Sun Sparc 1 workstations in our graduate laboratory. A typical machine in this lab has 16 Mbytes of main memory. There are no software limits to the number of participants that CSI can support. However, inspection typically involves six to eight participants. The number of windows supported on a screen in the X Windows environment is not limited, but our users found that having more than four windows open simultaneously was confusing.

Participants. The participants were student volunteers from a second-quarter software-engineering class. Half were seniors and half graduate students in computer science. In the first quarter, they analyzed and designed a hospital-management system that tracked Medicare patient admission and the quality of care and social services the patients received.

We chose one design for implementation as a team project in the second quarter, which began with an inspection of the selected design. The reviewers were familiar with the problem, but the design chosen for implementation was not theirs. The reviewers were also familiar with individual inspection from the previous quarter, but they may not have participated in formal software inspection. The producers were the actual producers of the design. The authors and two other class members served as moderators and recorders, and gave instruction on CSI use.

The participants were divided into three teams. Each consisted of three reviewers, a producer, a moderator, and a recorder. Four of the nine reviewers had more than three years of industrial experience. The reviewers were highly motivated because the final class assignment was to implement the resulting design. The final products of the three teams resulted in 7,500, 5,500, and 9,300 lines of code.

All participants were introduced to software inspection in a one-hour class. The producers and recorders were given a two-hour introduction to CSI. One hour covered the asynchronous part; the second hour covered the synchronous meeting. Each reviewer was introduced to CSI at the start of the individual reviews.

Method and material. Each team inspected four pieces of design material. The numbers of lines of text were 238(M1), 158(M2), 222(M3), and 21(M4). Because CSI currently supports only text, we gave each reviewer paper copies of supporting diagrams (object models and state-transition diagrams).

Participants performed all 36 individual reviews using CSI. We used a separate directory for each inspection. The reviewers moved to the appropriate directory and opened the target material and inspection criteria. When they found faults in the target material, they opened the Annotation object and recorded the faults.

All 12 inspections used CSI for initial inspection and fault correlation. Five inspection meetings used CSI and seven were face-to-face meetings. Each of the two producers supported two pieces of target material; thus each producer had two inspection meetings with each team.

TO ASSESS CSI, WE HAD STUDENTS USE IT TO INSPECT A HOSPITAL-MANAGEMENT SYSTEM.

Measurements. CSI electronically recorded the results of individual inspections, the correlated fault list developed by the producer, and the final action-item lists. The face-to-face meetings produced handwritten action-item lists.

Each team inspected the same four pieces of design material — M1, M2, M3, and M4. The reviewers classified faults according to types:

♦ *Major.* The fault is likely to result in failure to meet requirements.

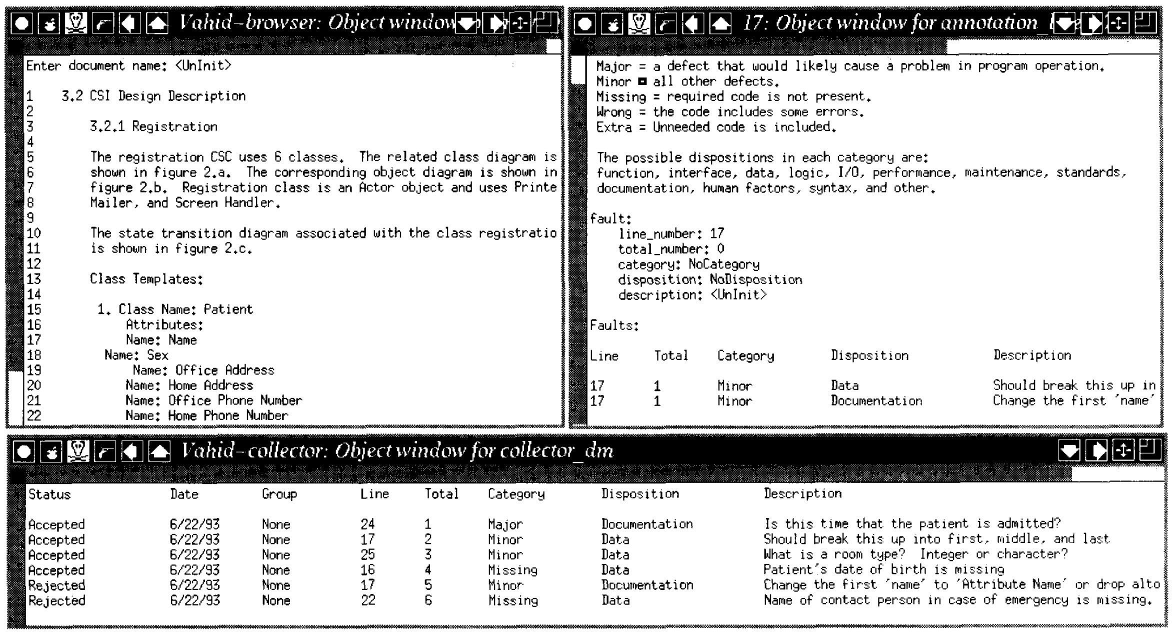
♦ *Minor.* The fault is likely to cause small problems.

♦ *Missing.* The design does not cover a requirement item.

♦ *Extra.* The design is beyond the requirements.

♦ *Wrong.* The design is incorrect.

♦ *Standard.* The design violates software-development standards.


Vahid-browser: Object window
17: Object window for annotation
Vahid-collector: Object window for collector_dm

```
Enter document name: <UnInit>

1     3.2 CSI Design Description
2
3         3.2.1 Registration
4
5         The registration CSC uses 6 classes.  The related class diagram is
6         shown in figure 2.a.  The corresponding object diagram is shown in
7         figure 2.b.  Registration class is an Actor object and uses Printe
8         Mailer, and Screen Handler.
9
10        The state transition diagram associated with the class registratio
11        is shown in figure 2.c.
12
13        Class Templates:
14
15         1. Class Name: Patient
16             Attributes:
17             Name: Name
18          Name: Sex
19              Name: Office Address
20             Name: Home Address
21             Name: Office Phone Number
22             Name: Home Phone Number
```

```
 Major = a defect that would likely cause a problem in program operation.
 Minor ■ all other defects.
 Missing = required code is not present.
 Wrong = the code includes some errors.
 Extra = Unneeded code is included.

 The possible dispositions in each category are:
 function, interface, data, logic, I/O, performance, maintenance, standards,
 documentation, human factors, syntax, and other.

fault:
    line_number: 17
    total_number: 0
    category: NoCategory
    disposition: NoDisposition
    description: <UnInit>

Faults:

Line     Total    Category      Disposition        Description

17       1        Minor         Data               Should break this up in
17       1        Minor         Documentation      Change the first 'name'
```

```
Status          Date        Group      Line    Total   Category     Disposition      Description

Accepted        6/22/93     None       24      1       Major        Documentation    Is this time that the patient is admitted?
Accepted        6/22/93     None       17      2       Minor        Data             Should break this up into first, middle, and last
Accepted        6/22/93     None       25      3       Minor        Data             What is a room type?  Integer or character?
Accepted        6/22/93     None       16      4       Missing      Data             Patient's date of birth is missing
Rejected        6/22/93     None       17      5       Minor        Documentation    Change the first 'name' to 'Attribute Name' or drop alto
Rejected        6/22/93     None       22      6       Missing      Data             Name of contact person in case of emergency is missing.
```

Figure 5. *CSI windows. When the reviewer finds a fault, he opens an Annotation by clicking on the line number. This brings up the Annotation window associated with that line. The Category and Disposition fields in the Annotation object provide a selection of values through pop-up windows.*

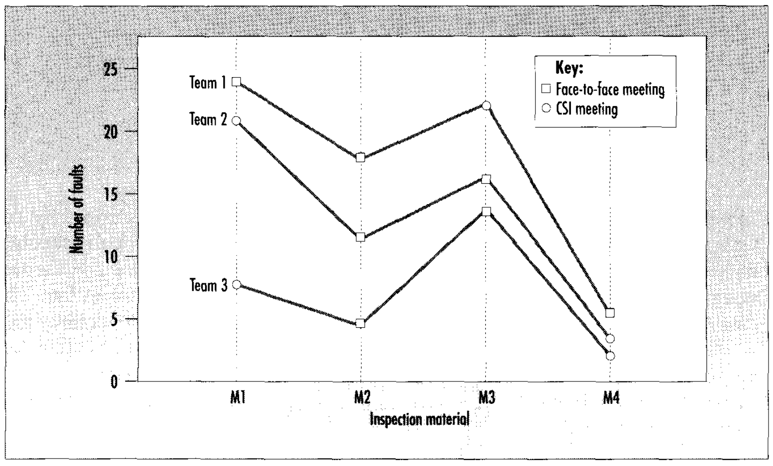


Figure 6. Faults found per inspection by team.

After the reviewers completed the individual reviews, we asked them to answer survey questions about using CSI for the reviews. The questions covered the availability of material and ease of use of CSI documents versus paper documents. After the inspection meetings, we gave the reviewers another survey. The questions covered the time the reviewers spent in both meetings, the feasibility of using computer teleconferencing to support verbal interaction, and the usability of CSI. The box below shows the questions and data collected for both surveys.

RESULTS

Here are the results of our case study, with our original research questions serving as a framework.

Are collaborative inspection meetings as successful as face-to-face meetings? Our results suggest that CSI meetings can be as effective as face-to-face meetings. Figure 6 shows the total faults found by each team in each of the four pieces of target material (*M1*, *M2*, *M3*, and *M4*). The squares indicate face-to-face meetings; the circles indicate CSI meetings. Lines connect the results for each team. The teams performed consistently, independent of the meeting form. Team 1 continually found more faults than team 2, and team 2 always found more faults than team 3.

We also compared the combined total number of faults found in individual reviews with the number of faults finally reported on the action-item list. Figure 7 graphs the faults for each team and again shows consistent performance within the teams, independent of the meeting form.

Answers to the survey questions indicate the participants found the use of CSI

SURVEYS AND RESULTS

Here we present the questions and results for our two surveys of case-study participants. After reviewers prepared individual fault lists, we asked them questions to determine how electronically assisted individual inspection compares with manual inspection: 5 is manual, toward 0 is worse, and toward 10 is better. The results are averages.

1. Was CSI as easy to use as the manual method? Result: 5.7

2. Did CSI provide access to all material needed? Result: 3.6

CSI did not provide as much access to material as the manual individual inspection. Reviewers like to have all of the design document, including tests and diagrams, rather than just the portion under review. Currently, CSI displays text only.

3. How did the time required to use CSI compare with the manual individual inspection? Result: 5.3

4. Can you find things in the text on the CSI screen as easily as on paper? Result: 3.8

5. Can you see (read) the screen as easily as paper? Result: 4.7

6. CSI is a new tool and it takes some time to become comfortable using a new tool. Were you able to concentrate on inspection rather than the tool? 1 means the tool always drew your attention away from inspection, and 10 means the tool never interfered with your concentration on the job. Result: 7.5

7. How long did it take you to become familiar with the CSI tool? Result: 34 minutes

We also administered a questionnaire after the inspection meeting. Again, the results are averages.

1. How long did you use the CSI screens? Result: 1 hour, 20 minutes

2. How long were you in the face-to-face meeting? Result: 2 hours, 20 minutes

3. How does the length of time required to access written material in the CSI-supported meetings compare with the time in face-to-face meetings? (Face-to-face: 5, longer toward 1, shorter toward 10) Result: 5.5

4. In the CSI-supported meetings, were you able to concentrate on inspection as well as in the face-to-face meetings? (Not able to concentrate on inspection at all: 1, total concentration on inspection: 10, face-to-face: 5) Result: 3.2

5. In the CSI-supported meetings, was there as much useful discussion as in the face-to-face meetings? (Less: 1, more: 10, face-to-face: 5) Result: 3.0

6. With the Teleconf sound tool, do you always know who is talking? (Never: 1, always: 10) Result: 5.0

7. With the Teleconf sound tool, can you hear everything? (Never: 1, always: 10) Result: 7.3

8. In CSI, how distracting is using the sound tool? (Tool is very distracting — no time for inspection: 1, tool is natural to use: 10) Result: 5.3

9. In CSI, how distracting is using the text objects? (Very: 1, no problem: 10) Result: 8.0

10. In the face-to-face meetings, how distracting are the paper copies of inspection material? (Very: 1, no problem: 10) Result: 7.0

Paper copies are not very distracting in the face-to-face meetings, but slightly more distracting than the text objects in CSI.

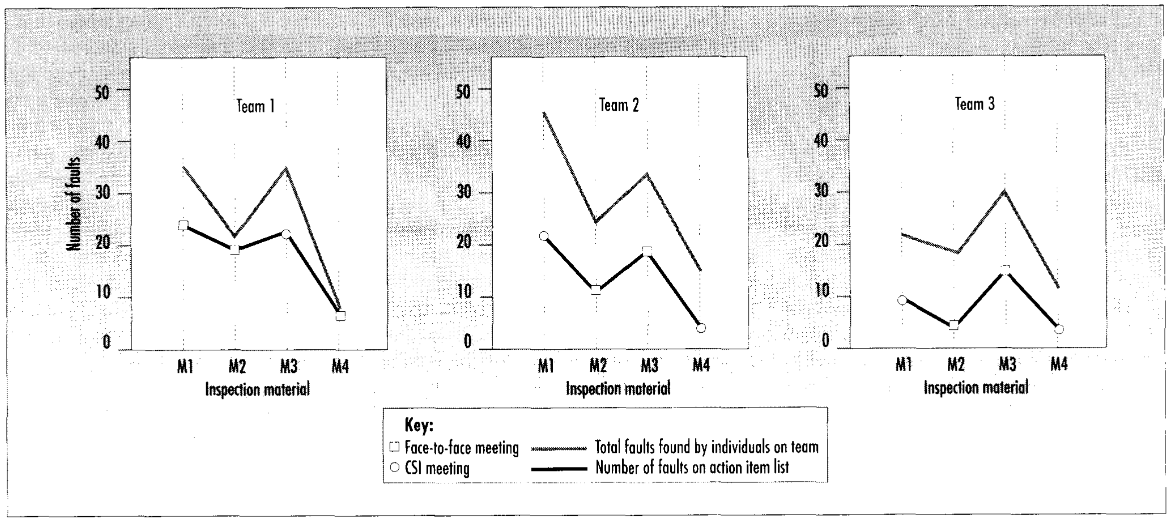


Figure 7. *Comparison of individual and final faults found per team.*

slightly more difficult than meeting face-to-face. They found it slightly more difficult to concentrate in the CSI meetings. Also, they felt there was less helpful discussion in distributed meetings using CSI and Teleconf than in face-to-face meetings. However, participants felt they had adequate access to inspection material in the CSI meetings.

Is on-line inspection material as usable as hard copy? Maintaining inspection material on-line reduces paperwork, makes the most current material available to participants, and reduces paper shuffling in the meetings. In this case study, all individual inspections and fault correlations were done using CSI. For the seven face-to-face inspection meetings, paper copies were necessary. The material copied includes

- ♦ correlated fault lists,
- ♦ target material,
- ♦ design-inspection criteria,
- ♦ forms for recording inspection summaries, and
- ♦ forms for action-item lists.

We observed several disadvantages of paper copies. Between the time the producer completes fault correlation and the beginning of the meeting, time must be allocated for making copies. Moreover, the paper and electronic copies of the target material must maintain the same layout and line numbering, because in the meeting reviewers cannot locate faults without consistent copies. Additional confusion can arise if, from the stacks of paper, reviewers grab the wrong piece of target material or a previous inspection's correlated fault list. Our pile of paper for the seven face-to-face meetings was small compared with that for a large project, which may involve hundreds of inspections.

In the five inspections that were completely electronically supported, none of these problems existed. The target and inspection materials are tied to the inspection at initialization. The forms are part of CSI, and fault notations are bound to the inspection when the reviewer creates them.

According to the survey results, participants felt access to materials using CSI was adequate in the inspection meetings but slightly inadequate in the individual reviews.

OUR STUDY FOUND THAT ELECTRONIC SUPPORT FOR ALL INSPECTION ACTIVITIES IS FEASIBLE.

Should the same tool support all inspection activities? Electronic support for all inspection activities using the same tool is feasible. Five of the 12 inspections used CSI for all inspection activities. For the seven that did not, moving to traditional meetings was more difficult than continuing with CSI because working with the paper copies was clumsier.

CSI was easy to learn, enabling team members to concentrate on the inspection rather than the tool. The mouse and key actions are consistent across the CSI objects, and users had little trouble learning the system. We observed that the average time spent reviewing all four pieces of design was four hours, and the reviewers concentrated on the review task quickly. We asked the reviewers how long it took to become comfortable with CSI. The median time was 34 minutes.

Most reviewers required more than one session to complete individual reviews. They were able to resume inspection without difficulty in using the tool.

The reviewers used CSI for the meetings without additional instruction, because the same objects support both individual reviews and meetings.

Is electronic support for fault correlation helpful? In our previous trials of CSI, reviewers performed the initial inspection on paper, and the producers manually correlated faults from paper lists. This was difficult and time consuming. Multiple paper lists are hard to manage. If the individual fault lists are not sequentially ordered, manual correlation of faults is almost impossible. Writing the correlated list from the individual lists is redundant work.

In the study reported here, on-line correlation of fault lists took a median of less than an hour. Producers could accept or reject faults, and reorder the list on the basis of line number or fault severity. No copying was done. Electronic support made correlation more effective because the producer could concentrate on understanding the faults, rather than on shuffling paper.

ACCESSIBLE INFORMATION IS A MANAGER'S DREAM, BUT INSPECTION IS PEER, NOT MANAGEMENT REVIEW.

What information was produced in automated inspection? The action-item list is the principal product of inspection. CSI creates it and makes it available electronically. In addition, the inspection summary, individual reviews, and the correlated fault list are on-line. The History Log time-stamps all system activities for later measurement.

We see a possibly negative side to the high accessibility to inspection results that the History Log provides. Easy accessibility to information is a manager's dream, but software inspection gains part of its effectiveness from being free of management intervention. Software inspection is peer review, not management review. The idea is to inspect the target material, not the producer or reviewers. Data from the History Log should be used to judge only the software process, not the inspection team.

What extra capabilities are needed to support automated inspection? CSI could benefit from

♦ a calendar program for scheduling,

♦ automatic start-up for initiating the meeting, and

♦ better recovery from node or net failures.

Does the inspection team have all the information it needs? In individual reviews, the reviewers said they wanted to see other parts of the documents. For example, they wanted to check a requirement from the requirements, specification or the introduction section of the design document. Currently, when designers want to reference the specification, they pull out a worn paper copy of the specification and ruffle through it. We envision a software-development environment in which all documents are on-line, current, and accessible. CSI could be improved by bringing even more software documents on-line, such as graphical specifications.

Locating items in text is a problem inherent in using documents. CSI and many other text-display systems locate items in the text by referencing the layout on the page. CSI uses line numbers. Changing font size or page width can affect what material falls on what line. An alternative is referring to the logical location within the text, such as paragraph numbers, chapter headings, and section numbers. Another alternative is hypertext links to words or phrases in the text.

In addition to faults in the target material, software inspections try to find missing items and conceptual faults. Reviewers' comments about these must be connected to the target material in some way. In this case study, we suggested that reviewers place such comments in the Note Pad, which is attached to the first line of the target material.

Does the manual process fit automated software inspection? In the version of CSI used for the case study, reviewers created a joint list from the individual inspections. For example, when the first reviewer found a fault, CSI recorded it. When another reviewer identified the same fault, he found the fault already listed on that line and did not record it. This brings up a question: Do we lose fault-finding capability because a fault is recorded from only one person's viewpoint? A second person may view the fault differently and detect additional faults as a consequence. This trade-off between efficiency and thoroughness in individual reviews deserves further study.

How should we order the correlated fault list? We observed that the reviewers preferred a correlated fault list ordered according to the sequential order of the target material. Six of the 12 correlated fault lists were ordered by line number and six by severity. In the inspection meetings, the reviewers and producers always checked the target material to make sure they understood the fault. Even though the faults had line numbers, the reviewers found it easier to locate a fault's source in the target material when they addressed the faults sequentially.

Is Teleconf's floor-control mechanism appropriate for inspection meetings? The Teleconf teleconferencing tool has objects for floor control and sound delivery. It supports two modes: first-in first-out and no floor control (free). In FIFO mode, participants queue to obtain permission to transmit sound, and only one station transmits sound at once. In free mode, each station can transmit sound to every other station.

Experiments showed that free mode imposes high CPU and network load. We reduced this load by using silence-deletion filters, which can reduce the traffic from quadratic to linear.[7] For example, in a four-person audio teleconference, silence deletion reduces the load from 96 to 24 Kbytes per second.

Each inspection team briefly used Teleconf. We believe that sound through the computer rather than through a separate channel is feasible. Tool use was straightforward and the sound was clear (the bit rate is the same as with a normal telephone). We used microphones and the speakers on the Sun Sparc workstations.

Because the microphones were handheld, participants found it difficult to use the mouse and keyboard at the same time. In more recent experiments with Teleconf, we used lapel microphones and headphones. Benefits are free hands, a quiet room for coworkers, and better silence deletion.[7]

The FIFO queuing protocol for floor control is adequate for small meetings such as software inspection. There is more silent time in inspection meetings than in other types of meetings. The producer and reviewers spend time reading the faults and target material and thinking. Participants set their own pace, and the queue gives each member adequate floor access. A moderated floor policy might also be appropriate for software inspection, as the participants play well-defined roles and there is a person of authority (the moderator).

Our case study is valid for inspection of design documents for programs between 7,000 and 10,000 lines of code and for designers experienced in individual inspection but new to collaborative software inspection. The results suggest that CSI meetings can be as effective as face-to-face meetings in finding faults. They also indicate that the ability of the inspection team has more influence on the number of faults found than the collaborative environment.

Software inspection is a well-structured group activity. CSI's success suggests that other well-structured group activities such as concurrent engineering and requirements analysis could benefit from a collaborative environment.[10]

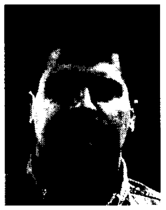
Interesting future work could include supporting a significant part of the in-meeting activities asynchronously, allowing for a finer granularity in fault recording (from sections to single words in a document); conducting performance studies on the audio component; applying new video technology; and studying recovery in collaborative systems. ♦

ACKNOWLEDGMENTS

We thank Shekhar Kirani, Horng-Jiung Lee, and all the participants in the software inspections.

REFERENCES

1. B. Boehm, "Industrial Software Metrics Top 10 List," *IEEE Software*, Sept. 1987, pp. 84-85.
2. D. Freedman and G. Weinberg, *Handbook of Walkthroughs, Inspections, and Technical Reviews*, Dorset House, New York, 1990.
3. W.S. Humphrey, *Managing the Software Process*, Addison-Wesley, Reading, Mass., 1989.
4. L. Brothers, V. Sembugamooprthy, and M. Muller, "Icicle: Groupware for Code Inspection," *Proc. Computer Supported Collaborative Work*, ACM Press, New York, 1990, pp. 169-181.
5. E. Yourdon, *Structured Walkthroughs*, Prentice Hall, Englewood Cliffs, N.J., 1989.
6. J. Nunamaker et al., "Electronic Meeting Systems to Support Group Work," *Comm. ACM*, July 1991, pp. 40-61.
7. J. Riedl et al., "SuiteSound: A System for Distributed Collaborative Multimedia," *IEEE Trans. Knowledge and Data Eng.* (to appear).
8. P. Dewan and R. Choudhary, "Flexible User Interface Coupling in a Collaborative System," *Proc. ACM SIGCHI Conf.*, ACM Press, New York, 1991, pp. 41-49.
9. R.K. Yin, *Case Study Research: Design and Methods*, Sage Publications, Newbury Park, Calif., 1989.
10. P. Dewan and J. Riedl, "Toward Computer-Supported Concurrent Software Engineering," *Computer*, Jan. 1993, pp. 17-27.



Vahid Mashayekhi is working on a PhD in computer and information sciences at the University of Minnesota. His research interests include collaborative systems, distributed systems, multimedia, and user recovery.

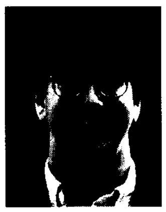
Mashayekhi received a BS and an MS in computer science from the University of Minnesota at Minneapolis.

W.T. Tsai is an associate professor of computer science at the University of Minnesota. His areas of interest are software engineering, artificial intelligence, computer security, and computer systems.

Tsai received a BS in computer science and engineering from Massachusetts Institute of Technology and an MS and a PhD in computer science from the University of California at Berkeley. He is a member of the IEEE and American Association for Artificial Intelligence.

Janet Drake is an assistant professor at the University of Northern Iowa. Her areas of interest include the analysis phase of software engineering, object-oriented analysis, interaction with customers and end users, and multimedia applications for software engineering.

Drake received a PhD in computer science from the University of Minnesota at Minneapolis.



John Riedl is an assistant professor of computer science at the University of Minnesota. His research interests include collaborative systems, distributed database systems, distributed operating systems, and multimedia.

Riedl received a BS in mathematics from the University of Notre Dame and an MS and a PhD in computer science from Purdue University.

Address questions about this article to Mashayekhi, Tsai, or Riedl, CS Dept., 4-192 EE/CSCI Building, University of Minnesota, 200 Union St. SE, Minneapolis, MN 55455; Internet {vmash, tsai, riedl}@cs.umn.edu; or to Drake, CS Dept., University of Northern Iowa, Cedar Falls, IA 50614; Internet drake@cs.uni.edu.