Software Reviews: The State of the Practice

Marcus Ciolkowski, *Kaiserslautern University of Technology*
Oliver Laitenberger, *Droege & Comp.*
Stefan Biffl, *Vienna University of Technology*

As software continues to permeate our daily lives, companies increasingly see the capability to develop and maintain reliable software products as a competitive advantage. Yet many companies continue to release defect-ridden software products. To push software development toward engineering-level precision, software development organizations must use quality-enhancing techniques, methods, and tools right from the beginning of a project.

One popular quality assurance method is the *software review*, which we define as a nonexecution-based approach for scrutinizing software products for defects, deviations from development standards, and other problems.[1] A *software product* is any work element resulting from such software development activities as requirements gathering, design, coding, or test plans. Some observers estimate that review methods and variants such as walkthroughs[2] and software inspections[1,3] permit the detection and correction of 50 to 90 percent of a software product's defects.[4,5] Even so, software reviews—"invented" at least 30 years ago[3]—still haven't become common practice in today's software development projects. At least, the extent to which review variants have penetrated the software industry remains unclear.

To investigate how industry carries out software reviews and in what forms, we conducted a two-part survey in 2002, the first part based on a national initiative in Germany and the second involving companies worldwide. Although the survey couldn't determine how many companies perform reviews, it does provide insight into how a sample of companies design and conduct reviews. This is important because reviews come in rather different flavors based on varying contextual characteristics and constraints.

A 2002 survey found that many companies use software reviews unsystematically, creating a mismatch between expected outcomes and review implementations. This suggests that many software practitioners understand basic review concepts but often fail to exploit their full potential.

0740-7459/03/$17.00 © 2003 IEEE

The survey yielded two main results. First, many of the responding companies integrated reviews into their software development projects, with goals ranging from early defect detection to better team communication. Second, review approaches vary widely, with a strong tendency toward nonsystematic methods and techniques.

What we mean by reviews

During a *software review*, an organization conducts an analysis or examination of software products that

- Primarily focuses on finding defects
- Is conducted by developers (not by an external group)
- Is conducted as part of the development process

This definition includes review variants such as walkthroughs and software inspections. In *walkthroughs*, one or more reviewers and the author meet to discuss a document.[2] *Inspections* involve a more formally defined process in which reviewers individually and systematically examine a document and record their findings.[1,3]

We can characterize reviews along several dimensions,[4] but we focus here on the review process and on reading techniques to support the reviewer in analyzing a document. In general, reviews typically follow a defined sequence of steps:

- *Review planning* involves selecting the review team, assigning roles and responsibilities, and scheduling the different review process steps.[6] Review entry and exit criteria can be established to make sure that reviews are planned and conducted appropriately.
- During *defect detection* (also called the preparation phase), reviewers individually read the document to get an overview and understand its content. Moreover, this is the phase to find and document defects.
- The review or team *meeting* is the traditional approach for collecting and discussing the defects detected by the individual reviewers. During the meeting, each individually reported defect is discussed and either accepted as a defect or refused as a false positive.
- Throughout the *defect correction* phase, the author of the reviewed document removes all reported defects and considers all stated comments.
- During *follow-up*, one of the review participants checks whether all reported defects have been dealt with. This step also includes giving feedback to the reviewers about the review results.

All review variants follow a similar pattern, although the parameters for each step may vary according to the review type chosen and its implementation in a given software development project and company. In some cases, a company may skip one or more steps altogether.

Reading techniques support the reviewer in detecting defects, and they vary widely. The least formal technique is ad hoc reading, where a reviewer just reads the document and notes any defects encountered. More formal reading techniques include checklists, perspective-based reading, and other advanced techniques.[7,8]

The survey

Despite the large volume of work on software reviews in the past three decades,[4,5] no large-scale study had examined review adoption in the software industry. So, in 2002, the International Software Engineering Research Network (www.iese.fhg.de/ISERN) and the Fraunhofer Institute for Experimental Software Engineering (IESE) initiated a large online survey on the state of software review practice. Because we defined the software review as a static examination conducted as part of the development process, we were only interested in regularly occurring review activities.

As noted, we conducted the survey in two parts, the first within the context of VISEK[9] (www.visek.de), a German project aimed at creating a virtual competence center on software engineering methods, and the second in cooperation with ISERN, an international network of researchers dedicated to empirical software engineering.[10]

We invited 865 people—contacts of Fraunhofer IESE and ISERN members—by email to participate in the survey. We also selected participants from the 2001 and 2002 International Conferences on Software Engineering and from the 2001 and 2002 International Symposia on Software Metrics, and placed calls in several newsgroups.

Despite the large volume of work on software reviews in the past three decades, no large-scale study had examined review adoption in the software industry.

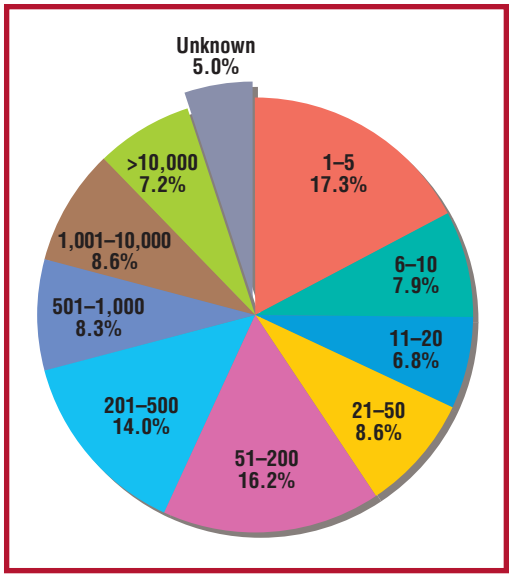


Figure 1. The size of the IT departments participating in the survey.

Altogether, 226 people from various domains and countries responded. Half of the respondents came from Germany and another 33 percent from the rest of Europe. Most respondents came from Germany because those responsible for both parts of the survey were located there and invested much time and effort to convince companies to participate. Similarly, the Norwegian project InterProfit funded some of the survey, resulting in a high percentage of companies from Norway (12 percent). Unfortunately, a comparable effort was not feasible for the North American market, so only 12 percent of the participants came from the US or Canada. Five percent came from elsewhere.

Companies of all sizes participated, from very small (one to five employees) to very large (more than 10,000 employees). Overall, about 47 percent of the respondents stated that they were from "small and medium-size enterprises" of fewer than 500 employees. Similarly, the respondents' IT departments ranged from very small to very large (Figure 1). Altogether, about 70 percent of the respondents stated that their IT departments had up to 500 employees.

Respondents also represented companies from various industry segments, including embedded systems (27 percent), telecommunications (10 percent), information systems (30 percent), Web or mobile applications (14 percent), and others (19 percent).

The most fundamental question about survey methodology is whether the respondents represent a valid sample of the entire population—that is, whether our sample is representative of "typical" software companies. As both parts of the survey depended on voluntary participation, the sample is likely biased toward companies that already use software reviews or are generally interested in quality assurance techniques. However, since we sought to examine how companies conduct the review process rather than what percentage of companies conduct reviews, this bias might be small enough to allow valuable conclusions. Another bias might result from quality and reliability being primary issues for companies that develop time- or safety-critical software. However, with no complete databases about software-producing companies worldwide, we couldn't do much to avoid these biases.

Survey results

We analyzed the results in terms of review process steps, goals, usage throughout different development phases, and economic aspects.

Execution of the review process

Table 1 summarizes the survey's main findings for each step in the review process.

Review goals. When asked about the main drivers for using review technology, 73 percent of respondents rated quality improvement as very important, 52 percent rated project status evaluation as important, and 54 percent viewed reviews as a means to enforce standards. Although other goals are important, too, the results indicate at first sight that early defect detection still is the major impetus for adopting review technologies. Also, the respondents perceived ad hoc (unsystematic) reviews as offering benefits other than defect reduction, such as improved communication and standards enforcement.

When asked about obstacles to using reviews, most respondents cited time pressure (75 percent), cost (56 percent), or lack of training in introducing reviews (50 percent). Looking at the introduction effort in more detail, only 12 percent of participants said they were trained in the review process, while 67 percent learned it by participation.

Using reviews in the software development process. Among all the respondents, 72 percent produce a requirements document, 68 percent produce a design, and 72 percent produce code during software development. The high number of companies that do not produce

Table 1

Summary of survey results

Review step	Primary survey results
Review goals	Main goals are quality improvement (73% of respondents) and enforcing standards (54%).
Development process	40% regularly review requirements and design documents, 30% regularly review code.
Review process	Only 20–40% conduct individual review steps regularly.
Planning, overview	20% usually perform planning, 30% do not use any formal entry or exit criteria for reviews.
Defect detection, preparation	40% usually conduct the preparation phase. Of those, 35% use ad hoc reading, 50% use checklists, and the rest use more advanced defect detection techniques or alternatives such as simulation.
Meeting	40% usually conduct a meeting.
Follow-up	35% have some kind of follow-up step. Independent of that, 40% verify rework, and 16% regularly give feedback.
Optimization	60% collect data, but 30% of these do no analysis, and fewer than 25% use the data to optimize reviews.

code seems surprising at first sight, but one possible explanation is that many companies outsource code production—a common practice in the automotive industry, for example.

However, even if they produce documents for a specific phase, fewer than half of the respondents regularly perform reviews. For example, of those who produce documentation, 42 percent review requirements, 40 percent review design, and 28 percent review code. That is, for all the development stages, most participants don't regularly conduct reviews.

Review planning. Among the respondents, 20 percent stated that they usually conduct an explicit planning and overview step. A third use no formal entry or exit criteria—for example, when asked how they identify whether a document should be reviewed, 32 percent said they use no criteria; 44 percent review a document because the project or quality plan requires it, and 40 percent do it because someone rates that document as critical. The rest use the document's size or complexity, for example. (This question had more than one possible answer, so the answers add up to more than 100 percent.) For other entry or exit criteria, we noted similar findings in that about a third of participants use no formal criteria.

Defect detection. About 40 percent of respondents regularly conduct a defect detection step—which means about 60 percent don't search for defects individually. Supporting this finding, half the respondents stated that reviewers only search for defects during the team meeting. Of those that conduct a defect detection step, 35 percent let their reviewers examine the documents in an ad hoc fashion, 50 percent use a checklist,[1] and 10 percent use more specific reading techniques such as scenarios or perspective-based reading;[6] the rest use simulation and other techniques.

Meeting. About 40 percent of participants regularly hold a review meeting. Such meetings can have different purposes, ranging from consolidating defects to detecting defects. Independent of whether they regularly conduct meetings, 50 percent of respondents consider defect detection to be a team activity, and 57 percent consider collecting defects as an important goal of a meeting, if they conduct one. This confirms that many participants see the value of team meetings in finding and collecting defects.

Defect correction. Clearly, reviews should involve some kind of correction phase, yet only about 40 percent of participants stated that they regularly conduct an explicit defect correction step. One possible explanation came up during a workshop where we discussed initial survey results. Some practitioners reported that, in their environment, defects are sometimes detected in a review but not corrected because the authors do not have enough time.

Follow-up. About 35 percent of participants stated that they regularly conduct a follow-up phase, during which someone verifies the rework and the reviewers receive feedback about the review results. Two additional observations support this finding. First, 40 percent of respondents stated that they have an explicit step to verify rework, independent of a follow-up phase. Second, only 16 percent of all companies provide regular feedback to the reviewers, 52 percent do so occasionally, and about one third never do.

Review economics

The participants reported, on average, three reviewers per review. Answers ranged from 1 to 20 reviewers.

To gauge review effectiveness, we asked participants to estimate the percentage of de-

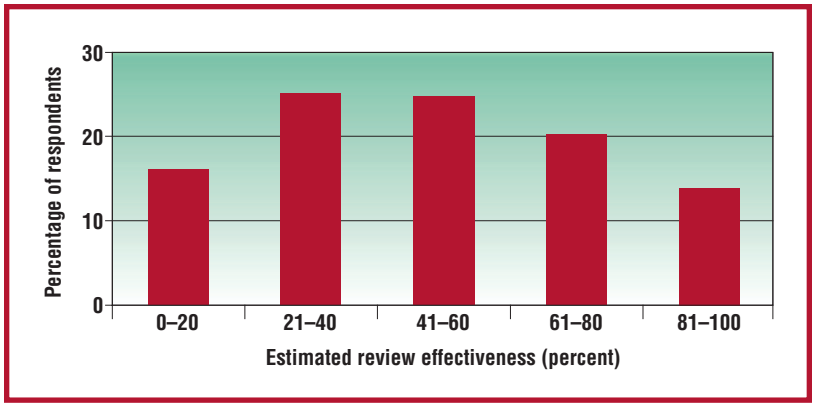


Figure 2. Survey respondents' ratings of review effectiveness.

fects typically caught with reviews. As Figure 2 shows, 16 percent of respondents stated they catch 0 to 20 percent of defects and 14 percent catch 81 to 100 percent, while 70 percent estimate their reviews' effectiveness in the mid range of 21 to 80 percent. Participants who collect defect data tend to rate review effectiveness higher than those who don't. However, since these data are only estimates, we hesitate to draw conclusions from them.

About 60 percent of companies represented in the survey collect some kind of data from their reviews. For example, 36 percent collect the number of defects found during a review, and about 20 percent collect effort data.

Of those who collect data, about 30 percent do not analyze them at all. Of the rest, 70 percent state that they do some form of optimization: 41 percent try to optimize the development process, 23 percent optimize the review process itself, and 6 percent do other optimizations.

Implications

Despite three decades of research on review technologies, no study has yet investigated their large-scale industrial adoption. This survey sample covers companies ranging widely in size and domain, but as participation was voluntary, we expect the sample to be biased toward companies that are interested in quality assurance technologies. In particular, we expect the penetration of reviews in practice to be lower than this survey suggests.

Our finding that respondents conduct reviews regularly but often unsystematically suggests that many companies don't exploit reviews' full potential for defect reduction and quality control. We base this statement on three observations.

First, reviews don't systematically cover the development phases. About 40 percent of respondents perform reviews on requirements or design documents, and about 30 percent on code. Empirical results show, however, that reviews of early documents can prove especially profitable,[4,5] and Michael Fagan's results show that code reviews also pay off even if the code is being tested later.[3]

Second, companies often don't systematically execute the defect detection step itself. About 60 percent of respondents said they do not regularly conduct a preparation phase for reviews, even though empirical studies have shown this step to be crucial for effective reviews.[11,12] Moreover, among those who conduct a preparation phase, 50 percent use a checklist to support that step, and fewer than 10 percent use a more advanced reading technique; fully 40 percent use ad hoc reading—that is, they don't offer systematic support for defect detection. Empirical results have shown that ad hoc reading generates highly variable results because it depends on the reviewer's expertise. Systematic reading techniques can significantly improve review effectiveness compared to ad hoc reading or checklists.[13]

Finally, companies seldom embed software reviews in a systematic process evaluation and improvement program. Overall, more than half the companies either do not collect data (40 percent) or collect data but perform no analysis (18 percent). Fewer than 25 percent of those who collect data try to optimize the review process, even though optimization has proved crucial to achieving highly effective reviews. There are no silver-bullet technologies—companies must adapt every technology to their specific contextual needs and characteristics to optimize its performance.

The Impact project is tracing industry success cases to research and tries to establish guidelines for successful technology transfer.[14] It found that for reviews, inspections, and walkthroughs, three factors usually proved critical to sustainable success in industry:

- *Process adherence*. Companies must integrate the review process into the development process and execute it regularly. This requires a defined, nonoptional review process.
- *Systematic reading technique*. Whereas results of ad hoc readings vary a lot and de-

pend on the reviewer's expertise, systematic reading techniques such as perspective-based reading[7] are more predictable, increasing review effectiveness.[13]

- *Optimization*. Experience shows that without optimization, reviews are not very effective.[15] Success stories that reported high effectiveness (90 percent or more) achieved this only by tailoring the review process to their needs and context.

Ad hoc reviews without clear process and method support can still be valuable—our survey data show that participants perceive benefits other than defect detection and quality control, such as enforced standards, project status evaluation, and improved communication. However, with 73 percent of participants citing increasing product quality as an important review goal, we see a mismatch between the expected purpose and the review implementation.

Companies aiming for sustainable, effective use of inspections, reviews, or walkthroughs should mimic successful companies' transfer guidelines. Web portals of experience centers such as CEBASE (www.cebase.org), VISEK (www.visek.de), and ESERNET (www.esernet.org) capture and present some of these guidelines.

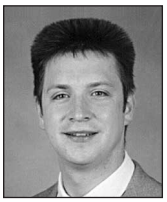
Acknowledgments

We thank the members of the International Software Engineering Research Network for their support in conducting the survey. In particular, our thanks go to Forrest Shull, Sira Vegas, and Dag Sjøberg. Parts of this work were funded and supported by the projects VISEK (funded by the German Federal Ministry of Education and Research, BMBF project no. 01ISA02), InterProfit (funded by the Research Council of Norway, project no. 24852), and ESERNET (funded by the European Union, project no. IST-2000-28754). We thank all survey participants for their contributions to this work and also thank workshop participants for their valuable feedback in the February 2002 ESERNET workshop on inspections and the April 2003 SQM workshop, "Inspection Methods in Practice."

References

1. T. Gilb and D. Graham, *Software Inspection*, Addison-Wesley, 1993.
2. E. Yourdon, *Structured Walkthroughs*, 4th ed., Prentice Hall, 1989.
3. M.E. Fagan, "Design and Code Inspections to Reduce Errors in Program Development," *IBM Systems J.*, vol. 15, no. 3, 1976, pp. 182–211.
4. O. Laitenberger and J. DeBaud, "An Encompassing Life-Cycle Centric Survey of Software Review," *J. Systems and Software*, vol. 50, no. 1, Jan. 2000, pp. 5–31.
5. K.E. Wiegers, *Peer Reviews in Software—A Practical Guide*, Addison-Wesley, 2002.
6. S. Biffl and M. Halling, "Investigating the Influence of Inspector Capability Factors with Four Inspection Techniques on Inspection Performance," *Proc. 8th IEEE Int'l Software Metrics Symp.*, IEEE CS Press, 2002, pp. 107–117.
7. V.R. Basili, "Evolving and Packaging Reading Technologies," *J. Systems and Software*, vol. 38, no. 1, July 1997, pp. 3–12.
8. F. Shull, I. Rus, and V. Basili, "How Perspective-Based Reading Can Improve Requirements Inspections," *Computer*, vol. 33, no. 7, July 2000, pp. 73–79.
9. O. Laitenberger, S. Vegas, and M. Ciolkowski, *The State of the Practice of Review and Review Technologies in Germany*, tech. report 011.02, Virtual Software Engineering Competence Center (VISEK), 2002.
10. M. Ciolkowski et al., "Practical Experiences in the Design and Conduct of Surveys in Empirical Software Engineering," *ESERNET Method and Experience*, R. Conradi, ed., LNCS 2765, Springer-Verlag, 2003, pp. 104–128.
11. S. Biffl, *Software Inspection Techniques to Support Project and Quality Management*, habilitation (post-doctoral) thesis, Shaker Verlag, 2001.
12. L.G. Votta, "Does Every Inspection Need a Meeting?" *ACM Software Eng. Notes*, vol. 18, no. 5, Dec. 1993, pp. 107–114.
13. B. Boehm and V.R. Basili, "Software Defect Reduction Top 10 List," *Computer*, vol. 34, no. 1, Jan. 2001, pp. 135–137.
14. L.J. Osterweil et al., "The Impact Project: Determining the Impact of Software Engineering Research upon Practice," *Software Eng. Notes*, vol. 25, no. 6, Nov. 2000, pp. 108–109.
15. L. Briand et al., "Quality Assurance Technologies for the EURO Conversion—Industrial Experience at Allianz Life Assurance," *Proc. 2nd Int'l Software Quality Week Europe*, Software Research Inc., 1998, pp. 1–23.

About the Authors



Marcus Ciolkowski is a research scientist at the Software Engineering Research Group at the University of Kaiserslautern, Germany, where he is pursuing his doctoral degree. He received an MS in computer science from the University of Kaiserslautern. His research interests include empirical methods for software engineering and quality management. Contact him at the Dept. of Computer Science, Univ. of Kaiserslautern, PO Box 3049, 67655 Kaiserslautern, Germany; Marcus.Ciolkowski@informatik.uni-kl.de; http://wwwagse.informatik.uni-kl.de/staff/ciolkows.

Oliver Laitenberger is a software consultant for Droege & Comp., an international consultancy. He received his PhD in computer science from the University of Kaiserslautern. His main research interests include industrial projects and process and quality management techniques, methods, and tools. Contact him at Oliver_Laitenberger@droege.de; www.droege.de.

Stefan Biffl is an associate professor of software engineering at the Vienna University of Technology. He received his PhD in computer science from the Vienna University of Technology. His research interests include project and quality management in software engineering. He is a member of the ACM and IEEE. Contact him at Stefan.Biffl@tuwien.ac.at; http://qse.ifs.tuwien.ac.at/~biffl.