Survey of Software Inspection Research

Sami Kollanus* and Jussi Koskinen*

Department of Computer Science and Information Systems, P.O. Box 35 (Agora), FI-40014 University of Jyväskylä, Finland

Abstract: There is a great need to assure and improve the reliability and quality of software. Software inspections were introduced over 30 years ago as an answer for this need and they have inspired a lot of research covering many different kinds of aspects. There is a need for an up-to-date survey revealing the current state and the overall evolution of the most prominent research on the area. This paper presents a comprehensive survey focusing on the most relevant 16 international high-impact scientific publication series. There are 153 articles included in the survey covering both technical and management aspects. The main results include a description of the research trends during 1980-2008 and a description of the main results of the included studies. The description is organized based on a taxonomy of the inspection research as having emerged based on the survey. At general level the surveyed research provides clear evidence that inspections generally benefit software development and quality assurance. There are several proposed theoretical variations for the inspection process but also many empirical studies. Although the conducted research is relatively scattered, proper science-based understanding about some of the most studied issues has been achieved. Our main conclusion is that conducting empirical research needs to be continued in order to validate the effects of the different kinds of proposed theoretical constructs in practice. Empirical studies are needed especially in order to better understand the proper implementation and the actual impacts of applying inspections in different kinds of industrial and organizational settings.

Keywords: Software inspections, software reviews, software quality assurance, literature survey.

1. INTRODUCTION

There is a great need to assure and improve the reliability and quality of software. It is claimed, for example, that 50-60% of the effort involved in producing large software systems is devoted to *quality assessment* activities [1]. The percentage may even be significantly higher for life-critical software systems. Quality assurance activities include testing but also other techniques may be used. Most importantly, software inspections enable early quality assurance prior to testing.

Software inspections were originally introduced already over 30 years ago by Fagan [2]. Inspection is a widely acknowledged technique within the software engineering research community. Several empirical studies have reported great savings or improved effectiveness when using inspections [3-5] to support quality assurance in the studied specific research settings. The progress in effectively implementing and using the different kinds of developed inspection techniques has, however, been relatively slow in practice.

Due to these reasons, it is important to review how the research field has evolved and what the main advances have been. There are also other significant related surveys on inspections [6-8], but they are partly out-dated or having limitations in their scope.

This paper presents the results of our survey of the software inspection research published during 1980-2008. The survey has comprehensively included 16 high-impact publication series on the general software engineering field, and includes a total of 153 relevant articles.

This paper is organized as follows. Section 2 briefly introduces the original inspection process as a background for the treatment of the research field. The applied research method is described in detail in section 3. Section 4 outlines the general main results and the identified trends concerning software inspection studies. That section also presents an emergent taxonomy of the surveyed research. Sections 5-7 investigate the individual classes of the taxonomy; namely technical view, management view, and other main topics. These sections give more detailed information about the most prominent branches of research and advancements related to each subclass. Section 8 presents some discussion regarding the future of research on the field and section 9 summarizes the main conclusions of the survey. Full bibliographic information of all of the articles included in the survey is provided in the reference list. Finally, Appendix 1 provides a complete classified citation list to the surveyed articles.

2. SOFTWARE INSPECTIONS

Software inspections are used to increase the quality of software, documents, and other artifacts produced during software development. The original Fagan's inspection process [9] is relatively straight-forward and consists of the fol-

*Address correspondence to these authors at the Department of Computer Science and Information Systems, P.O. Box 35 (Agora), FI-40014 University of Jyväskylä, Finland; E-mails: sami.kollanus@jyu.fi, koskinen@jyu.fi

1874-107X/09

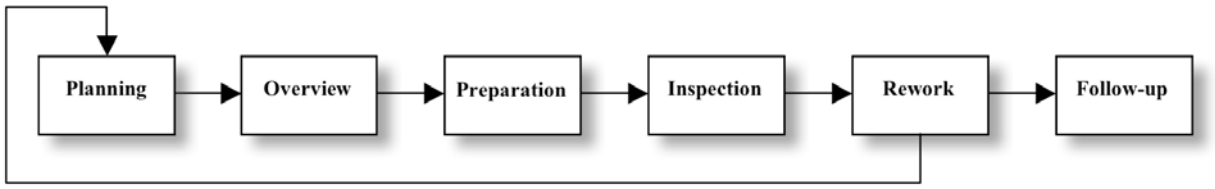

Fig. (1). Fagan's inspection process.

lowing phases as presented in Fig. (**1**): Planning, overview, preparation, actual inspection, rework, and follow-up.

Planning means checking defined entry criteria and taking care of the practical arrangements like resources and the location of the inspection meeting. *Overview* is a short meeting for training the participants. Tasks and roles are assigned in the meeting. The overview meeting is not a compulsory part of the original process. The next phase is *preparation* for the actual inspection meeting. Participants study and learn individually the materials being inspected based on their assigned roles. One of the key ideas in the inspection is to assign different roles to the participants based on their expertise. Fagan's inspection team includes four different *roles*: Moderator, designer, coder, and tester [2]. The moderator leads the team and takes care of the practical arrangements. The other roles represent the viewpoint of their expertise in the process.

According to Fagan, defects are primarily found during the actual *inspection meeting*, which focuses on going through the materials with checklists and registering (logging) the found defects. Many other researchers do not agree with Fagan in this defect finding issue. Instead, they consistently state that most of the defects are actually found already during the preparation phase. After the meeting the author(s) *rework* the found defects. If necessary, another round of inspection can be arranged after the rework. Finally, the moderator or the entire inspection team verifies the changes as part of the *follow-up* phase.

The inspection process described here is the basis for most of the inspection research discussed in this paper. Later variations of the process have been relatively small. Some authors, as stated, for example, in several handbooks including [10-12], prefer to use slightly different terms for phases or roles. However, the basics of the inspection process have remained consistent over time.

3. RESEARCH METHOD

This section gives a description of the applied research method and our research process. Brereton *et al.* [13] suggest conducting systematic literature reviews (including surveys) in the general context of software engineering. The systematic nature of the research process is generally important for transparency and for being able to be aware of the potential risks biasing the research setting. The risks are reduced, for example, by following the general normative guidelines, as suggested in [13] in form of a process model for reviews.

We feel that that model is one of the best for supporting the organization of literature reviews and surveys. The strong side of that model is that is provides a very detailed process model for conducting research. However, it also has some limitations; its use is rather elaborate and also reporting the surveys according to the model may take much space as journal articles. In case of writing a survey whose research questions are on a rather general level, all the details are neither relevant. The main intended focus area of that model has been evidence-based research, which is widely applied, for example, in medical research. The model consists of three main phases: *Plan review*, *conduct review*, and *document review*. These main phases in turn have multiple subphases.

Since we are not focusing on evidence-based research, but our goal is to create a broader description of the research field, we will apply that model here only as a general organizing framework for conducting our literature survey. We do not claim that this survey would be an exercise of strictly following all the details of the model. The guidelines of the model had to be somewhat adapted to our context. The adaptation has mainly meant giving relatively more emphasis and details to the research activities which are relevant in this context and giving less emphasis to those which are less relevant.

Plan Review

This main phase incorporates most importantly specification of the research questions, and developing and validating a review protocol.

Specify Research Questions

The first subtask was to specify the research questions. We basically wanted to gather information about *how much* and *what kind* of research in general has been conducted over an extended time period in order to reveal the *state and evolution* of the research field. Reporting of those issues consequently requires also developing or using some kind of a *taxonomy* of the surveyed research. By gathering the information on trends also a view of the *general body of knowledge* on the area. The elaboration and specification phase of our research questions has been rather straightforward. The final research questions in their general form were as follows. 1) What kinds of trends can be identified during the studied time period in the inspection research? 2) How the research conducted in this area can be classified? 3) What is the general scientific body of knowledge of the prominent studies in the area of software inspections? These questions were later specified a bit further as described in the oncoming subsections.

Develop Review Protocol

The review protocol guides conducting the research in a systematic fashion. There were two team members; the authors of this paper, participating to the survey. They each

had their own role and they cross-read each other's outcomes as an internal quality check. The main reviewer developed the initial protocol. We will here tell about our design choices. The made decisions were also followed unless stated otherwise in the following text. We first decided to focus on software inspections and software reviews. We decided to select only high impact publication series to the survey in order to focus on prominent research and to keep the amount of the analyzed materials under control. We decided that regarding the publication forums which will finally be selected all the relevant articles will be included. Full coverage in this sense supports formation of a comprehensive view of a large portion of the relevant research. Abstracts of all of the articles to be included were decided to be read fully through and the actual article contents at the level which was needed in order to answer the set research questions reliably. It was also decided that the solved research problem, applied research method, and main results of all the articles to be included in the survey, would be recorded.

Validate Review Protocol

The validation of the protocol is an internal task of the research team. The review protocol is instrumental in gradually increasing the systematic nature of the survey and aiming at quality control. In our case the significance of the systematic process has been acknowledged, it has been aimed at, and use of the protocol has afterwards proven to be meaningful for reaching the recommended objectives.

Conduct Review

This main phase includes many activities. The first and the most important one is identifying the relevant research. The other activities are selecting primary studies, assessing study quality, extracting required data, and synthesizing the data.

Identify Relevant Research

Due to the relatively great potential extent of the survey this was the most elaborate and in that sense central subphase. The initial core of the articles to be covered was based on our earlier expertise on the area [14-17]. Next we systematically studied the references of those articles and listed the journals in which the cited articles had been published. It soon became obvious that there was too much potential material to be included in the survey, if we did not somehow limit the number of the included sources or their intended scope.

There are currently over 360 *scientific journals* in the general area of computer science (i.e. computing) which have an *impact factor* and have been acknowledged by ISI Web of Knowledge [18]. There are over 80 journals which have been classified in ISI as software engineering journals. However, despite the great amount of these kinds of journals, there are not many journals which are dedicated to the core of software engineering as defined e.g. in SWEBOK [19] and which are relevant for software inspections.

On the other hand, there are hundreds of annual *scientific conferences* in the general area of computer science [20, 21]. DBLP [21] lists over 1100 conferences. There are also dozens of conferences dealing specifically with the various aspects of software engineering.

Due to these reasons, we decided to conduct; as suggested in [13], a systematic *pre-review* to get a general impression of the volume of the conducted research and published relevant articles. The processes of conducting the pre-review and the actual review were very similar. The pre-review gradually focused on the 15 main software engineering journals on the basis of having relatively high impact factors as determined by [18]. The conferences to be considered were selected by going systematically through the conference list of DBLP [21]. There were about 120 conferences (i.e. about 10% of the DBLP conferences) which had some identified software engineering connections. We further noted that about 95% of these are such that they are accessible via the most important *electronic article archives*: IEEE Xplore [22] or ACM Digital Library [23].

Next we determined the *article inclusion criteria* to be used in the pre-review (and potentially and also actually) also in the oncoming more detailed actual review. Most of the authors of the inspection-related articles use the term *inspection*, but some favor the more general term *review*, even if they actually discuss about the same kind of process. Therefore, it was decided to cover also all the papers, which are related to peer reviews. Since the term inspection is clearly dominant in the research area, we will consistently use it in this article.

The *covered time period* had to be long enough to reveal the existing research trends. After Fagan's [2] original publication on inspections, more continuous research on the area has not emerged before the 1980's. Therefore the year 1980 seemed to be a good starting point, covering practically all of the conducted research.

The pre-review for the conference articles was conducted by using the *search terms* "software inspection" and "software review" for the IEEE Xplore and the ACM Digital Library. The selection criteria were the appearance of either one of these terms in the title and the range of the years 1980-2008.

Potential journals were scrutinized in detail. Their *tables of contents* have been checked through completely for the selected time period. Additionally, the *search engines* of the respective publishers have been used as a double check. Therefore, the reliability of the representativeness of the survey regarding its intended scope has been at least as good as by using search engines alone. The way the relevance of the individual articles was ensured was that all those papers, which focused mainly on either software inspections or software reviews and had produced new scientific results, which are relevant in this context, were included.

The pre-reviews revealed 122 relevant journal articles and 107 relevant conference articles. Since the pre-review required going through about 120 conferences but only 15 journals, and yet there were more relevant journal articles, it became clear that these journals are more relevant sources in this sense. Journals typically have a significantly greater impact on the research community. It is also quite common that an extended or enhanced version of an important article published earlier in some conference proceedings will later

Table 1. The Number of the Inspection Related Papers in the Selected Publication Series During 1980-2008

Publication Series	Publisher	Studied Years	N
IEEE Transactions on Software Engineering (TSE)	IEEE	1980-2008	30
International Conference on Software Engineering (ICSE)	ACM	1981-2008	26
IEEE Software	IEEE	1984-2008	19
Journal of Systems and Software	Elsevier	1980-2008	19
Empirical Software Engineering	Springer	1996-2008	18
Information and Software Technology	Elsevier	1987-2008	11
Software Testing, Verification and Reliability	Wiley	1991-2008	9
Communications of the ACM	ACM	1980-2008	5
Software Quality Journal	Springer	1992-2008	4
IET Software	IET	1988-2008	3
Software Process: Improvement and Practice	Wiley	1995-2008	3
ACM Transactions on Software Engineering and Methodology (TOSEM)	ACM	1992-2008	2
Software: Practice and Experience	Wiley	1980-2008	2
Annals of Software Engineering	Springer	1995-2002	1
International Journal of Software Engineering and Knowledge Engineering	World Scientific	1991-2008	1
ACM Computing Surveys	ACM	1980-2008	0
Total			**153**

appear in a journal. For these cases it suffices to cover the later journal version in the survey to form a representative view of the covered research themes.

Due to the reasons presented above, we finally decided to focus on the identified main journals due to the otherwise too great amount of potentially included studies. Nevertheless we decided to include also ICSE (i.e. the annual *International Conference on Software Engineering*) since it is the most influential of the general software engineering conferences, accessible via both the IEEE Xplore and the ACM Digital Library, and containing a notable amount of inspection-related articles. Based on the described process, we finally had the list of publication series later to be presented in Table **1** for our actual survey. The process of the actual survey was basically the same as in the case of the pre-review, but it delved deeper into the details of the articles on the selected forums.

Select Primary Studies

We went through all the selected publication series during the selected time period. First we used the available *bibliographic information* of the articles, their titles and abstracts, to identify and list the relevant ones. Then full versions of the initially selected articles were purchased and read. Abstracts of all of the included articles were read through fully. Generally abstracts sufficed as a basis for selecting the primary studies, since our research questions were general in their nature. The article contents as such have been read through at varying levels of details depending on the specific needs. About half of the articles were such that they were read through practically completely.

The main essence of each reviewed article was first *summarized* into one sentence. Additionally a summary of some of the articles was written. Some extraneous information was also gathered which was not used further. For example, empirical studies were studied rather thoroughly. Those articles which were best validated were studied further. It was, for example, checked whether their validation was based on views or functions of experts or students. During this phase we also finally determined the perimeters of the survey and excluded some of the papers from the preliminary article list. The related decisions were made unanimously by the members of the research team.

The made selections are characterized and representative examples of the *excluded articles* are given as follows. Some of the articles clearly did not focus on inspections although they partly discussed them among other issues. For example, Shull *et al.* [24] discuss different reading techniques, which have been regularly used also in inspection research, but the focus is on program comprehension. Another similar example is the paper by Chillarege *et al.* [25] which studies defect classification. These kinds of studies clearly support inspection research to some degree. However, if we had included these papers in our survey, we should have included also all the literature representing techniques which could in principle be applied to inspections. That would have made the survey too elaborate, unfocused and large to be published as a journal article. Additionally, several articles (e.g. [26]) discuss quality improvement on a general level, but they do not cover software inspections on a detailed level.

Assess Study Quality

Assessing the quality of the primary studies can be used as an additional criterion for their exclusion. We focused exclusively on the articles published in high-impact publication series. Therefore, e.g. journal versions of the articles were favored instead of the conference article versions, if

there were both. Another example is different versions of the same articles in several journals or magazines. For example, *TSE* (*IEEE Transactions on Software Engineering*) and *IEEE Software* published special issues on software inspections in 2003. They included basically the same inspection articles but in different forms. The papers in *TSE* are written in a more scientific and detailed style whereas the papers in *IEEE Software* are directed to practising professionals and therefore typically contain less details and background information. We discarded these kinds of "duplicates" and included only the *TSE* articles in the survey.

Extract Required Data

The materials were gone through in an internally uniform way. The needed data was extracted during the process of reading through the articles. Their essential characteristics have been recorded according to the review protocol. The most important research results from each study were collected and written down. Specific *data extraction forms* could in principle have been used in this subphase. They are useful in decreasing the potential for bias due to variations in research process and the nature of the analyzed papers. As stated earlier, in our case the research questions were on a rather general level, which largely eliminated the potential of bias in this sense. However, we acknowledge that additional checks by multiple persons of the extracted data concerning the registered results of the individual studies could somewhat have further increased the reliability of the study in that regard.

Synthesize Data

After the previous phases, we created an *emergent taxonomy* classifying the articles based on their research question. That was necessary in order to manage the complexity and also to answer one of our three research questions. The classification did not follow any previously defined model, since we wanted it to reflect the actual status of the research and themes appearing in the articles. The next main level section will give more details about this issue. The synthetic activities included also identification of the trends in the inspection research over the studied time period, which will also be discussed in more detail in the next main level section.

Conduct Review

This last main phase simply consists of documenting the research process, along with writing and validating the actual research report based on the gathered data and the process followed in the earlier phases. We have published our initial review report in the form of a working paper. This paper is an enhanced version of it. This main section has described the central decisions made during the research process. The paper has aimed at covering the essential aspects of the research process and the results reached related to the set research questions. The following main sections will describe the actual findings. Finally, the validation of the final research report is in essence a task of the external reviewers on the intended publication forum.

4. GENERAL RESULTS OF THE LITERATURE ANALYSIS

This section introduces the most important research trends, which were identified in the surveyed inspection research during the covered time period. The first subsection describes the extent of the research in different publication series and during different time periods. The second subsection is focused on different topics in inspection research.

We describe how the research is broken down into different classes and how the trends have changed during the studied period. The third subsection presents some important general observations.

Research Volume

Table **1** presents the number of relevant papers found in each of the selected publication series. The studied time period for each publication series is also shown since some of the journals have not been published through the whole time period. The publication series are presented in descending order of the number of the identified relevant papers. The total number of the included articles was 153. The four most relevant publication series, *TSE*, *ICSE*, *IEEE Software*, and *Journal of Systems and Software*, accounted for over half of the articles.

It should be noted that the name of the studied *IEE*'s journal has been changed several times. It was initially *Software Engineering Journal* (during 1988-1996), then *IEE Proceedings – Software Engineering* (during 1997), *IEE Proceedings - Software* (during 1998-2006), and finally it has been renamed as *IET Software* (since the beginning of 2007). *Annals of Software Engineering* has not been published after 2002.

The number of inspection related publications has continuously grown till the recent years. There were very few papers published before the 1990's. Also in the beginning of the 1990's research activity was low. Research on the area started to increase as late as the mid 1990's. This growth can be seen in Figs. (**2**) and (**3**). Fig. (**2**) presents the total annual number of the published articles included in the survey. Fig. (**3**) presents the number of articles as divided into 5-year phases, besides the first bar, which presents also the early years of the studied time period. The figures show that after long continuous growth trend a remarkable decrease in the inspection research activity has occurred during the last four years.

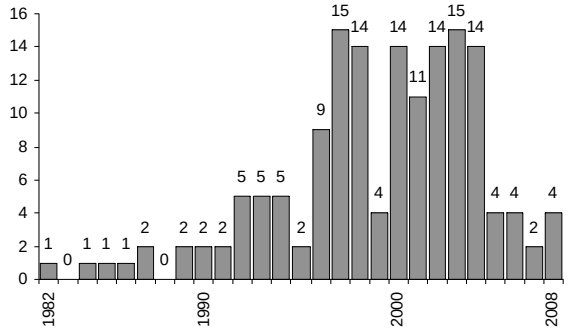

Fig. (2). The total number of annual inspection related papers in the reviewed publication series.

Many of the studied publication series have not appeared through the whole surveyed period (i.e. 1980-2008). How-

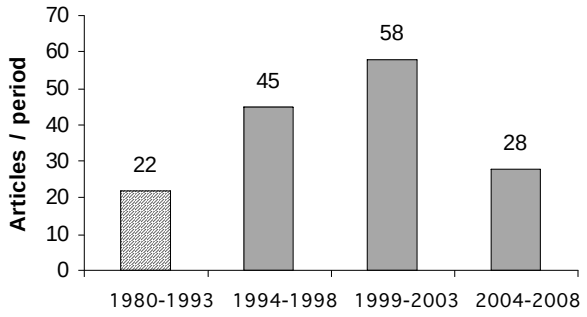

Fig. (3). The number of inspection related papers during 5-year periods in the reviewed publication series (as an exception the first bar gathers all the early years of the inspection research).

Table 2. The Used Taxonomy and the Amount of Articles in Each Class

Amounts Classes	Absolute Amount	Relative Amount
Technical view	*(76)*	*(49,7%)*
Reading techniques	25	16,3%
Effectiveness factors	23	15,0%
Processes	22	14,4%
Other technical topics	6	3,9%
Management view	*(27)*	*(17,6%)*
Inspection impact	14	9,2%
Other management topics	13	8,5%
Other main topics	*(50)*	*(32,7%)*
Defect estimation	23	15,0%
Inspection tools	16	10,4%
Comprehensive views	4	2,6%
Unclassified topics	7	4,6%
TOTAL	**153**	**100%**

ever, this natural phenomenon has only a relatively small effect to the general publication volume and to the identified trends. The most affecting source in this sense is *Empirical Software Engineering*, which has been published only since 1996. Most of the important topics in inspection research have nevertheless emerged around the middle of 1990's or later.

Classification of the Research

There are three earlier literature surveys on software inspections [6-8]. Porter *et al.* [8] conducted a relatively large survey, but it is already old and mainly represents only the early stage of the inspections research.

Laitenberger & DeBaud [7] is another extensive earlier survey. It is based on an analysis of roughly 400 studies and includes 99 references to inspection articles and reports. It aims at classifying the conducted inspection research. Several parts of that classification are still valid, but there are three concerns in our research context. Firstly, most of the materials covered in that survey have been collected before 1998, whereas 65% of the contents of our survey have been published since 1998. Secondly, the research field and the mainly studied subareas have changed. Thirdly, Laitenberger & DeBaud appear to have classified some articles into several classes. From our perspective, it gives more clear results to place each article simply into one class based on the research question. Due to these reasons, our survey updates and complements the views provided in that earlier survey.

The third survey of the area is Aurum *et al.* [6], which focuses on the variations of the Fagan's original method [2]. That survey is the most recent one including also four articles which have been published in 2000. However, it mainly deals only with the literature published before 2000. That survey includes 56 references. Our survey differs from it by a somewhat wider scope, inclusion of the more recent studies, and focusing on the high-impact publication series. About 55% of the contents of our survey have been published since 2000.

Due to the above mentioned reasons it was sensible to create a newer classification, reflecting better the status of the newer research. During our analysis we identified nine clear themes in the analyzed material. Only seven articles remained unclassified. Finally, we grouped the identified themes into three main classes: Technical view, management view and other main topics. The final classification and the number of the identified articles in each class are presented in Table **2**.

As already noted, Appendix 1 provides a complete classified list of the surveyed publications. Its Table **4** cites to the individual articles and also gives their publication years.

We used applicable parts of the earlier classification of Laitenberger & DeBaud [7]. It used technical-, managerial-, organizational-, assessment-, and tool dimensions. Otherwise, our taxonomy has emerged based on the identified nature of the studies included in this survey. The classification of the individual articles was based on their identified major research question. The idea of technical and management views as first-level classes is derived from [7].

Technical view is clearly the most studied main aspect. Table **2** shows that this class represents nearly half of the surveyed articles. There is a close match between the technical dimension of [7] and our technical view. Technical view includes variations of inspection processes, reading techniques applied in inspections, effectiveness factors affecting inspections and other substance related issues. Reading techniques and effectiveness factors were clearly the most studied issues. There exists especially many newer studies on the effectiveness factors. Therefore, we have also defined the effectiveness factors as a second-level class. On the other hand the volume of the research on processes and other technical topics was marginal.

The volume of the included new research related to the managerial and organizational dimensions based on the stated main research question is relatively low. Therefore, there was no need to split that part into separate main classes, as done in [7], and we use only the management view as a first-level class. The assessment dimension appeared to contain mainly only conference-level and partly more general-level and less related publications and was thus also useless to be separated. The volume of new research on

Table 3. Relative Amount of Publications in Each Class During the Different Phases

Phases and Years Classes	Phase I 1980-1993	Phase II 1994-1998	Phase III 1999-2003	Phase IV 2004-2008
Technical view	*(54,5%)*	*(53,4%)*	*(43,9%)*	*(51,6%)*
Reading techniques	0,0%	**17,8%**	**21,1%**	**17,2%**
Effectiveness factors	0,0%	**22,2%**	14,0%	**17,2%**
Processes	**54,5%**	6,7%	5,3%	13,8%
Other technical topics	0,0%	6,7%	3,5%	3,4%
Management view	*(27,2%)*	*(20,0%)*	*(7,0%)*	*(27,5%)*
Inspection impact	**22,7%**	4,4%	3,5%	**17,2%**
Other management topics	4,5%	15,6%	3,5%	10,3%
Other main topics	*(18,1%)*	*(26,7%)*	*(49,2%)*	*(20,6%)*
Defect estimation	9,1%	8,9%	**22,8%**	13,8%
Inspection tools	4,5%	15,6%	12,3%	3,4%
Comprehensive views	0,0%	2,2%	5,3%	0,0%
Unclassified topics	4,5%	0,0%	8,8%	3,4%
TOTAL	**100% (n=22)**	**100% (n=45)**	**100% (n=57)**	**100% (n=29)**

tools is modest, so it was only included as a second-level class. On the other hand, there is much new research on defect estimation. Therefore, it was included as a new second-level class.

It should be noted that this kind of classification has some limitations. For example, several studies that mainly represented the technical view also discussed issues that are relevant also from the management point of view. However, based on their major research question, we classified them under the technical view. Therefore, the surveyed articles include more research on the management view than can be seen from the sheer numbers in Table **2**. Consequently, only 18% of the articles were classified under the management view. Resolving this issue in a more sophisticated manner would require a much more complex classification method, which could, however, be less illustrative.

The final structure of the classification was not obvious in the beginning of our research process. We followed some principles to optimize the classification. The following list presents the most important principles:

- Some papers discussed issues from many classes. These kinds of papers were classified based on the major research question.
- The papers related to reading techniques could have been included under effectiveness factors. However, we wanted to explicate that set of techniques as a class of its own, due to the relatively large amount of the articles.
- Some papers about inspection tools were more focused on inspection process than on technical aspects of software. However, all papers that were clearly related to inspection tools were classified into the inspection tools class.

It is interesting to view the material also in terms of the time scale. Table **3** presents the proportion of papers included in each class during 5-year periods, but the first column makes an exception, representing all the early years (i.e. 1980-1993) of the conducted inspection research. The table shows the most studied viewpoints during each phase. The relative amounts of the articles in the most populated classes are written in bold-face in each column of the table. Evolution and changes of the research themes since 1980 represent a natural development and extension of scientific knowledge concerning an initially novel technique.

Phase I: The early phase (1980-1993) of inspection research focused on different modifications of inspection processes and the first experiences about the impact of inspections and implementing them in industry.

Phase II: During the next phase (1994-1998) inspection research expanded to cover new viewpoints. During this phase, the research on most of the classes started and inspection research mainly reached its current form. Reading techniques and other effectiveness factors became the most common questions in inspection research. Also different management issues were emphasized and the research on inspection tools was more active than during the later phases.

Phase III: The next phase (1999-2003) was the most active time in inspection research overall. The emphasis was especially on defect estimation and reading techniques. About half of the studies related to these topics were published during this phase.

Phase IV: It is characteristical to the last five years (2004-2008) that the conducted research represents the different classes of topics more evenly than before. The most active research topics have been different reading techniques, effectiveness factors and organizational inspection impact.

General Observations

The presented classification provides a view of the most studied research topics. However, due to the small number of articles in the smallest classes, the orientation of that re-

search may have been *sensitive* to the current interests of single active research groups or even individual researchers. For example, more than half of the papers related to defect estimation were published by two active research groups. So, the changing research trends can be partially explained by the evolution of individual interests. Nevertheless, the survey describes the issues that have actually been studied. When new researchers come to the field, they may bring new research questions with them.

The previous subsection presented and discussed the used classification. The classification could also have been formed by taking all the discussed topics in each paper into account. That would have created a more fine-grained view of the research, and could have revealed, for example, the actual role of the management aspects within the individual studies better. However, that kind of classification would also be relatively hard to create systematically and thereby its results could be more vulnerable to misinterpretations by other researchers.

The conducted studies can also be characterized based on the *applied research methods*. For example, 72% of the studies applied or reported some kind of empirical research. 56% of the empirical research was based on controlled experiments and the rest was based on field experiences in the software industry. 77% of the controlled experiments were student experiments, and the rest used professionals as subjects. Therefore, it can be stated that *empirical research* has a dominant position on the field. A typical example of inspection related research is a controlled student experiment.

Most (77%) of the surveyed articles discuss inspections. In addition, most of the other articles define the *peer review* process to be very similar to the inspection process. The authors of those papers appear to have some reason to avoid using the term inspection. We found only few papers on peer reviews that really significantly differ from inspections.

5. TECHNICAL VIEW

Technical view includes research that is focused on the question "How inspections should be implemented?" These studies are focused on different aspects in the inspection process. The papers discussing inspection tools are an exception to this. Some of them have this kind of research question, but in our classification all tool papers are kept in their own special subclass. Technical view has been the most popular aspect in inspection research. Half (50%) of the articles were classified under it. The following subsections will describe the most important results from the earlier studies related to the technical view, including: Different reading techniques, effectiveness factors, and inspection processes.

Reading Techniques

Reading techniques mean here different kinds of methods to find defects from documents; such as requirements documents, design documents, and source code, during the inspection process. This has been one of the most active research areas. About 16% of the papers were focused on it. All the presented techniques have been created to support *individual inspection work*. They obviously are based on the assumption that most defects are found already during the preparation phase before the inspection meeting. The preparation task is traditionally individual work and so far researchers have not been very interested in studying preparation as a group-based activity. However, many of the presented techniques include the idea of distributed work, but they do not consider preparation as a group-based process.

The original inspection method [2] included the idea of *using checklists in defect finding*. Checklists contain expertise concerning the most common defects and thereby support inspections. Ciolkowski *et al.* [27] reported that about half of the respondents in their survey used checklists in peer reviews, 35% did not use any kind of support material, and about 10% used more specific reading techniques presented in the literature.

Traditionally all inspectors use the same checklist in finding defects. Parnas & Weis [28] criticized this issue already in 1985 and assigned the defect finding task for different roles in their *active design reviews*. Another key idea in their method was to make the reviewer's role more active than in traditional checklist-based reading. Many other reading techniques in the later research are based on these basic ideas as highlighted by Parnas and Weis.

The research conducted related to the different reading techniques is mostly based on empirical settings, where some specific technique is compared to checklist-based reading or to *ad hoc* approaches. The starting point in creating a new technique has usually been some critique presented against the traditional checklist-based reading. Laitenberger & DeBaud [7] have summarized the critique into the following key points: 1) The nature of questions [in the checklists] is often general and they are not sufficiently tailored to take into account a particular development environment. 2) Concrete instructions on how to use a checklist are often missing. 3) Inspectors may not focus on defect types which have not been previously detected and, therefore, might even miss entire classes of defects.

Active research on different reading techniques started from an article published by Porter & Votta [29]. They presented a new *scenario-based reading technique*. Scenario-based reading is based on scenarios, which give inspectors more specific instructions than typical checklists. Additionally, inspectors are provided with different scenarios, focusing on different kinds of defects.

Porter & Votta [29] organized a student experiment, in which they compared scenario-based reading, checklist-based reading and *ad hoc* reading in requirements specification inspection. Their conclusion was that scenario-based reading was the most effective technique in finding defects, whereas there was no significant difference between the effectiveness of checklist-based and *ad hoc* reading.

The experiment of Porter & Votta [29] has been replicated several times. Porter *et al.* [30] presented new student experiment data, which supported the original results. Later, Porter & Votta also replicated the original experiment by using experienced professionals and reported again similar results [31]. Also the results reported by Miller *et al.* [32] regarding a student experiment have supported the original results. Some replications of the original study have produced also different kinds of results. Fusaro *et al.* [33] and Sandahl *et al.* [34] did not find significant differences be-

tween the applied techniques in their student experiment. However, at least some kind of support for the claim of the usefulness of scenarios has been produced by the previous research. The affecting factors behind the different results can only be guessed, based on these studies.

Basili *et al.* [35] define scenario-based reading as a general-level term, which they break down to more specific techniques. They call the original method of Porter & Votta [29] as *defect-based reading*. In their paper Basili *et al.* present a new scenario-based technique, which they call *perspective-based reading*. It is also a method for requirements inspection and the basic idea is based on different skills required and aspects covered in software development. Participants have an active role in inspecting requirements of their own area of expertise. For example, a testing expert first creates a test plan based on the requirements specification and then attempts to find defects from it.

Basili *et al.* [35] studied perspective-based reading in an experiment with experienced professionals as subjects. They compared that technique to the techniques usually applied by the participants of inspections to find defects. Their original goal for applying perspective-based reading was to improve the coverage of defects found in inspections. The results supported this goal, but also the individual participants found more defects by using perspective-based reading than by using their original techniques. Recently, Maldonado *et al.* [36] have replicated this study using university students. They also found the perspective-based reading more effective than use of checklists.

Later perspective-based reading has been applied both to code inspections [37] and design inspections [38]. These studies have provided promising results. In addition to the presented benefits, Laitenberger & DeBaud [37] reported that perspective-based reading appeared to balance the effect of experience on the number of the found defects. Therefore, this technique might partially compensate limited expertise. However, the results of Sabaliauskaite *et al.* [39] do not support the claim of the superiority of the perspective-based technique.

According to the definition of Basili *et al.* [35], most of the reading techniques presented in the surveyed articles are scenario-based. The basic idea is usually *cognitive activation* so that the inspector has to actively work with the inspected documents instead of mere straightforward reading.

Thelin *et al.* [40-42] introduce *usage-based reading*, which they apply in design inspections. Design documentation is inspected based on use cases, which are documented in requirements specification. This approach sets the focus on finding functional defects, which are relevant from the user point of view.

Dunsmore *et al.* [43-45] present an *abstraction-driven technique* for code inspections. In this technique, the inspector creates an abstraction level specification based on the code under inspection. The purpose of the task is to ensure that the inspector has really understood the code.

Kelly & Shepard [46] have created a code inspection method, which they call *task-driven inspection*. It is not only a reading technique, but it also includes some elements related to the inspection process. Their method includes the same kind of idea as the abstraction-driven technique by Dunsmore *et al.* [43], but Kelly & Shepard have defined the task more specifically. The inspector has to create a data dictionary, a complete description of the logic and a cross-reference between the code and the specifications.

Progress on scenario-based reading techniques can be summarized as having provided promising results as compared to checklist-based reading. However, the results are not completely consistent and most of the research settings have not been replicated by any independent research group. The conducted studies include mainly comparisons between checklists or *ad hoc* approach and some new techniques, whereas very few studies have compared the different scenario-based techniques. More research is needed in this area to understand how the different methods really perform under different conditions.

Effectiveness Factors

Effectiveness factors refer here to inspection process factors, which may affect the effectiveness of inspections. On the other hand, effects of inspections on software processes are discussed under the management view. Different kinds of effectiveness factors have been one of the big research issues. This class included 25 (15%) of the surveyed articles. The reading techniques presented in the previous subsection could also have been classified here, because the focus in that research is on inspection process factors.

The meaning of *effectiveness* is not self-evident, instead it may include several aspects. The most important terms in inspection research are *efficacy* and *efficiency*. Efficacy means the number of found defects and efficiency usually means the number of found defects per inspection hour. Some studies take only efficacy into account which may be reasonable in a context where extremely high quality is required. However, in most cases it is not a relevant metric, because inspection resources are limited in practice.

Therefore, most of the research is focused on finding an optimal cost-benefit ratio by determining efficiency. However, even effectiveness does not provide enough information to determine a practical cost-benefit ratio. The determination requires the user organization to measure the real benefits from finding a defect during inspections.

The starting point of the active research on effectiveness issues was probably Votta's [47] study, in which he questions the meaning of traditional meetings in the inspection process. Later, effectiveness issues have been studied from different perspectives. The following subsections discuss the most important effectiveness factors, which have appeared in the surveyed articles. They include individual performance, meetings, preparation, the amount of inspected materials, team size, training, and roles.

Individual Performance

Sauer *et al.* [48] studied inspection effectiveness issues theoretically by using behavioral theories. They argue that *individual expertise* is the most important factor in inspection effectiveness. This conclusion was recently supported

by Hatton [49], who found huge individual differences in defect finding task in his empirical experiment. This is actually the same conclusion that Barry Boehm [50] made almost 30 years ago related to the whole software engineering field. Knight & Myers [51] reported in their article that *experience in the used programming language* had a significant impact on the defect finding task. Individual performance is claimed to explain inspection effectiveness in some other studies as well; including [52].

Inspection Meetings

The usefulness of *inspection meetings* has been one of the most popular research topics related to inspection process since Votta's paper [47]. Votta compared the performance of a group (A) applying traditional inspection with another (B), which did not apply a meeting. Instead, individually found defects were collected by other means. Group B found even more defects than group A. Based on this finding, Votta argued that the meeting is meaningless and only a resource-consuming element in the inspection process. However, he also reported that the meeting succeeded in eliminating most of the false positives, i.e. assumed defects, which were not real defects.

Several studies [32, 53-55] have later supported Votta's findings. All these studies have reported that meetings did not improve defect finding as compared to various optional arrangements. These studies have considered meetings primarily as an extra cost and have recommended replacing meetings with other arrangements.

Despite all the critique, the critical view is not consistent among all researchers. Kitchenham *et al.* [56] use the inspection meeting discussion as a bad example about drawing conclusions. They state that the usefulness of inspection meetings can not be judged merely based on the amount of identified defects, because meetings have been reported to have also positive influences on the inspection process. D'Astous and Robillard [57] agree with this point and emphasize different kinds of perspectives in studying inspection meetings. Finally, there are also contrary results. Ebert *et al.* [58] found inspection teams with meetings to be more effective than teams without meetings.

There seems to be two different lines in the research related to inspection meetings. Some researchers are focused on inspection process effectiveness and usually criticize the usefulness of meetings as part of the inspection process. Other researchers emphasize different perspectives, which favor inspection meetings. An example of these perspectives is to consider meetings as a *place for learning and knowledge sharing* [59]. It should be noticed, that all researchers who criticize the effectiveness of inspection meetings, do not suggest replacing them completely. Johnson & Tjahjono [60] recommend meetings when employees are inexperienced with inspections. They also recommend discarding meetings after employees have gained more experience.

Preparation

Gilb & Graham [11] claim that *individual preparation* for inspections is the most important element contributing to the effectiveness of the inspection processes. It might be that this issue has been considered so self-evident that it has not been systematically studied. Some studies with other kinds of primary objectives have also reported results supporting this claim. Laitenberger *et al.* [61] noticed in their study at DaimlerChrysler that the more participants used time for preparation, the more defects they found. Also Christenson *et al.* [62] found a positive correlation between the preparation time and the amount of found defects.

Amount of Materials

Already Fagan [9] recognized that it is essential not to attempt to inspect too much material in any one inspection cycle and recommended inspecting about 125 lines of software code per hour. Also Gilb & Graham [11] emphasized the significance of the proper amount of material and gave strict recommendations concerning it. They claim that rigorous inspections lose their potential to identify really critical defects, if participants do not use enough time for preparation, which in turn is impossible if there is too much material to be inspected.

Dunsmore [63] gives some support for the presented recommendations. He calculated the optimal amount of materials based on empirical data from the industry. As a result he recommended inspecting 200 lines of object-oriented code per hour. On the other hand, Seaman & Basili [64] noted that in practice the speed of inspections was 60 pages of software code per hour even in their case in NASA. Bourgeois [65] has also reported similar results in his study. Thereby, the suggestions of the researchers and the practice in the field seem to differ a lot at least in these cases.

Team Size

Some researchers have tried to determine the most effective *team size* for inspections. Porter *et al.* [52] arranged an experiment to assess the costs and benefits of code inspections. They analyzed different factors including team size. They found out that four inspectors did not find significantly more defects than two inspectors. However, both four and two inspectors were clearly more effective than only one inspector. Porter *et al.* reported that team size had more effect on the *inspection interval* than the actual finding of defects. A big team carries the risk of delaying inspection arrangements, because it is often hard to find common time for meetings.

Also some other studies have found two inspectors to be clearly more effective than only one [49, 61]. So, two inspectors seems to be the usually suggested team size. However, these studies have tried only to identify the optimal time consumption per defect. If only the found defects are considered, increasing team size could be reasonable. However, without knowing the real saving gained by finding the defects, it is impossible to define the optimal team size that would be relevant also in practice.

Training

Proper *inspection training* has been proposed to be a factor affecting inspection effectiveness. Actually, this is a logical conclusion based on the revealed importance of individual skills in inspections. Rifkin & Deimel [66] reported that

training improved inspection effectiveness. They compared different kinds of inspection training programs. Based on their study, they recommended practical training focused on defect finding skills. That kind of training was found more effective than other training programs that focus on process level issues. Ebert *et al.* [58] later presented similar kinds of conclusions on the general improvement of inspection efficiency by training.

Roles

Some researchers have tried to understand the roles of those participating in inspection processes. Most of these studies are related to reading techniques. Most of the presented reading techniques include the idea of dividing the defect finding task into parts and thereby avoiding overlapping activities. In these studies, it is hard to distinguish the effect of the division of work from the effect of the applied reading technique. It can only be guessed that possibly both of these factors affect the results. Land *et al.* [67] have studied the effect of the procedural roles (e.g. moderator) on inspection process. They did not find these roles to have affected inspection effectiveness in their study.

Other Viewpoints

Porter *et al.* [68] studied several proposed effectiveness factors in an experiment. They varied team size, *number of inspection cycles* and defect correction between inspection cycles. These factors did not produce a significant effect on inspection effectiveness. Biffl *et al.* [69] gave some support to these results. They found that more than one inspection cycle causes a reduction in inspection effectiveness, when measured as identified defects per hour. However, they also calculated an estimate of the ensued savings. According to the calculations, a second inspection cycle may pay itself off.

Porter *et al.* [68] conclude that the variations in inspection effectiveness must be due to some other process factors. They suggest that the type of the inspected documents or the inspectors might explain the variation better. Individual performance was already discussed earlier, but the effects of *document types* have not been systematically studied in the surveyed articles. Nevertheless, Christenson *et al.* [62] proposed that *complexity of the documents* has an effect on the effectiveness of inspections.

Carver *et al.* [70] have received some interesting results in their recent study on the effectiveness of requirements inspection. They found in their experiment that the employees, who did not have their educational background in computer science, found more defects. Prior working experience generally did not have any impact, but experience in writing requirements had a positive effect on finding defects.

Processes

Several different kinds of modifications of the inspection process have been proposed over the years. This class includes 22 (14%) articles, which mostly represent the earliest phase of the inspection research. Later studies have focused more on *in-process factors* as described in the previous subsections.

After Fagan's [2] original inspection paper presenting the general form of the process, Runge [71] presented i*nspection adaptation for small projects*. Later Parnas & Weis [28] introduced active design reviews, which we already discussed related to the reading techniques. Bisant & Lyle [72] developed a *two-person inspection method* based on the given critique that traditional inspections allegedly require too much resources. Later Kusumoto *et al.* [73] picked up the same idea and received some supporting evidence for its usefulness in practice.

There are only a few really new methods presented in the surveyed articles. Schneider *et al.* [74] studied *n-fold inspection*, which has been created especially for the needs of critical projects. Their idea is based on several parallel teams doing the same inspection. According to Schneider at al., several *independent teams* find more defects than a single team. The n-fold inspection was originally presented a couple of years earlier by Martin & Tsai [75].

Knight & Myers [51] have presented *phased inspection*, which they created for software code inspections. Consistent with its name, it includes several phases, which are actually inspection cycles focusing on different aspects. For example, language, source code layout and programming constructs could each be inspected in a separate phase.

Thelin *et al.* [76] bring forth a new perspective with their idea of *sampling-based inspections*. Their basic idea is to inspect a sample of documents instead of all documents produced during the software development process. First, one inspector does a so-called pre-inspection phase and checks about 20-30% of the documents. This phase is used to determine which documents need the most inspection time. Then, the main inspection is focused on these documents.

None of these few inspection variations have yet inspired other researchers to do further studies. Instead, the later studies are mostly focused on the in-process factors of the traditional process. However, some of these studies suggest some changes into the traditional inspection process. Most of these suggestions are related to a debate on traditional inspection meetings. Sauer *et al.* [48] state that it may not be necessary to have the whole inspection team in a meeting, but a *couple of experts* could go through the defects found in the preparation phase. Mishra & Mishra [77] have refined the purpose of inspection meeting. They suggest that the found defects should be logged in advance and the common time in the meeting be used effectively in the discussion about the findings. Kelly & Shepard [46] have left out the whole meeting from their method. However, they suggest arranging a *start-up meeting*, which is usually an optional phase in the process.

Rigby *et al.* [78] raise up a current topic in their recent paper discussing on review practices in open source development. They report experiences from a large open source project, which differs much from traditional in-house software development. A different kind of aspect in inspection process has been the support of software reviews with *formal methods*. Some degree of formality has been aimed at in reviewing specifications by Jackson & Hoffman [79] and Polack [80] and software code by van Emden [81].

One additional clearly process-related viewpoint is tool support for inspections. The debate on the inspection meeting is one important theme, which has inspired some re-

searchers to develop tool support. This aspect is discussed separately in Section 7.

Other Technical Topics

There were 6 (4%) articles, which clearly represented the technical view, but did not fit into the subclasses presented here. These studies are typically somehow focused on the question: "What should be inspected related to the documents?" Macdonald *et al.* [82] and Dunsmore *et al.* [83] studied issues which are specific to the inspection of *object-oriented code*. Tervonen [84] presented some *principles for supporting inspectors* with relevant materials and tools. Chernak [85] introduced a model for the *systematic improvement of checklists*. Traore and Aredo [86] discussed the relationship between *automatic verification* and technical reviews. Finally, De Almeida *et al.* [87] presented *the best practices* for code inspections.

6. MANAGEMENT VIEW

The management view certainly represents a practically useful area of inspection research, but for some reason it does not include as much research as could be expected. Only 27 (18%) of the surveyed articles were primarily focused on management view. However, as noted earlier, many studies with a more technical focus have discussed topics that are also relevant to management. Inspection impact on the development process was the only clearly consistent theme under the management view. 14 (9%) papers were related to inspection impact on development process and the other 13 (9%) papers represented miscellaneous related topics.

Inspection Impact on Development Process

This class has been emphasized at the early stages of inspection research. The typical article is a case study in some specific company. For example, already Bush [88] described how the Jet Propulsion Laboratory calculated the *benefits gained* by using inspections. According to Bush, correcting one defect later in the process would cost US$1,700 and therefore one inspection saves on average US$25,000.

Russell [5] reported that every inspection hour saved 33 hours of maintenance work at Bell-Northern Research. Doolan [3] found in his research that every hour invested in inspections paid itself back 30 times. Grady & Van Slack [4] have also reported similar results at Hewlett-Packard.

Some studies in this class have compared the effectiveness of code reviews and *testing methods*. Basili & Selby [89] noticed in their early experiment that more defects were found and more effectively by reading code than by testing it. Jalote & Haragopal [90] also received similar results from their case study. So *et al.* [91] did not find a difference in the number of identified defects, but their study also verified inspections as being more cost-effective than the testing methods which they studied. Houdek *et al.* [92] studied different defect detection techniques for executable specifications. They did not find a notable difference in the effectiveness of testing and inspections. Roper *et al.* [93] did not find any difference in effectiveness between code reading and different testing methods. They also refer to several other studies that have produced inconsistent results. Runeson *et al.* [94] have recently published a survey focused on this topic. Their conclusion based on the past research was that there is no clearly superior technique for finding defects.

It can be concluded that the earlier research does not include consistent evidence of inspection effectiveness as compared to testing methods. Instead, several studies; including [93], have concluded that code reading and different testing methods have their own strengths in effectively identifying different kinds of defects. These studies usually suggest *using a combination of different methods* to find all kinds of defects effectively.

Müller's [95, 96] studies represented a different kind of theme. He compared pair programming and individual programming as supported with peer reviews. Student experiments did not reveal significant differences in effects in terms of code quality and development costs.

Zheng *et al.* [97] brought a new perspective to inspection research in their study on *static analysis*. Their study compared the defects found based on automated static analysis and code inspections. Based on the result they suggest using them both as complementary methods.

Other Management Topics

This class included other articles, which are primarily focused on the management view. Some of the articles discuss metrics. Barnard & Price [98] present various *metrics* relevant to code inspections. Madachy [99] presents a model which can be used to measure inspection impact on the software development process. Briand *et al.* [100] have created a model for creating *inspection efficiency benchmarks*. Recently Freimut *et al.* [101] introduced a method to determine cost-effectiveness of inspections in an organization.

Chatzigeorgiou & Antoniadis [102] focused on *inspection scheduling*. In a case organization they revealed an important phenomenon: Inspections tended to be postponed and accumulated towards internal project deadlines leading to excess overtime costs, quality degradation, and difficulties in reaching milestones. Related to this same issue, Kusumoto *et al.* [103] introduced earlier a *time allocation procedure* for reviews.

Jakob & Pillai [104] presented a *statistical process control method*, which can be used to improve coding as well as code reviews. Their process control is based on monitoring defects found in reviews during the process. Chaar *et al.* [105] discussed evaluating and improving inspection and testing activities.

Jalote & Haragopal [90] represented a somewhat different viewpoint. They discussed *inspection adaptation* in organizations and are focused on the so-called "not-applicable-here (NAH)" syndrome, which hinders effective adoption and application of inspections in organizations. They introduced a simple approach to overcome this kind of resistance and reported promising results from one case organization.

Surprisingly, only a couple of papers clearly focused on *organizational inspection improvement* issues. The significance of inspections in software engineering is generally acknowledged, but their effective implementation in the

software industry is a problem. Some studies have created methods that provide support for organizational inspection improvements [106-108]. Ideas about *capability or maturity models* for inspections have also been presented [4, 14, 109], but there exists only very few significant empirical studies on them.

7. OTHER MAIN TOPICS

This section introduces the other classes identified in the survey material. These classes are defect estimation, inspection tools and comprehensive views of inspections. These classes comprise 50 (33%) of the papers in the surveyed material. Seven articles remained unclassified.

Defect Estimation

This class includes the research that aims at finding a way to reliable *estimatation of the defect content* of a document after inspection. The usually proposed starting point is that this information is useful for project management while making decisions regarding further actions after the inspections.

One of the important themes in the surveyed articles is *estimation of the defect amounts*. 23 (15%) of the surveyed articles discuss it. Moreover, there is an increase in the relative amount of these kinds of studies during the latest years within this survey. Since 2000 it has even been the most popular individual inspection research issue.

Eick et al's [110] article is a starting point in this subarea. That study applied *capture-recapture sampling*, which is well known in the field of ecology. The principle behind that approach is that an estimate of the total amount of defects within an analyzed document is made based on the amount of the found defects. Most of the approaches in this area attempt to form a maximally reliable model based on this method. Another early study on this field was Vander Wiel & Votta [111].

This approach requires prior knowledge of the way inspections are applied in the user organization. For example, Padberg [112] shows how to form a sort of *organization-specific profile* of the found and the not found defects based on inspection history.

Petersson *et al.* [113] have written a comprehensive survey of the studies conducted on this subarea by then. Different studies regarded different kinds of estimation models as promising. However, these models are still too imprecise to be successfully applied in practice.

Research on this area has separated into two branches during the latest years. Earlier estimation models have been characterized as objective. Another newer branch is research on *subjective estimation*. These estimation models are simply based on the subjective views, concerning the amount of found defects, of the participants of the inspections.

Biffl & Grossman [114] claimed that the then current objective estimation models were actually quite inaccurate, especially when defect detection effectiveness was low. The objective methods appeared to perform relatively well under some conditions. Therefore, Biffl & Grossman claimed that the real question here is about when the estimates can be trusted. They suggested complementing objective estimation with other indicators like subjective estimation.

Subjective estimation has been studied by El Emam *et al.* [115], Thelin [116], and Yin *et al.* [117]. El Emam *et al.* [115] arranged an experiment with professional software engineers as subjects and found out that the median relative error of the received subjective estimates of defect content is zero. Yin *et al.* [117] reported similar results based on their experiment with students. In these experiments, the participants usually performed quite well. The problem in practice was that the reliability of the estimates was weak. Thelin [116] compares objective and subjective estimation and considers the objective method to be more reliable.

Inspection Tools

All the articles related to computer aided *inspection tools* were included in this class. This has clearly been one interesting topic in inspection research, because 16 articles (11%) dealt with this class. The specific interest in this area emerged during the late 1990's after the rapid growth of the Internet. The typical aspect in this class is *computer aided distribution* of inspection tasks. The articles surveyed here offer only a limited view of inspection tools, but Hedberg [118] has presented a more extensive review of the previous research on the field.

The earliest studies in this area emerged already in the beginning of the 1990's. Within our survey Mashayekhi *et al.* [119] were the first ones to introduce a tool for supporting inspections. Johnson [120] also presented similar ideas about tool support. The basic idea was *asynchronous inspection*, where the inspectors do not have to be in the same place at the same time. The goal of this approach is to save costs and increase the flexibility of the inspection process.

Various inspection tools have been presented, e.g. by Macdonald & Miller [121, 122], Perry *et al.* [123], and Tervonen [124]. All these tools were primarily created to support the inspection process. Instead, Anderson *et al.* [125] focused on program comprehension support in their study.

Macdonald & Miller [126] conducted a student experiment to compare tool supported inspection and paper-based inspection. The only difference revealed between the two procedures was the defect finding task. They did not find a significant difference between these two groups and they regarded this as a promising result for tool supported inspection. Tyran & George [127] conducted a similar kind of study, but in their procedure the tool-supported group interacted only by writing through the tool interface. The tool-supported group found more defects than the other group. Tyran & George gave several explanations for that result. For example, traditional meetings often had one dominant member, restricting open discussion. In addition, sidetracking was significantly lower in the tool supported group.

The research on tool support is related to the nature of the traditional inspection meeting. Perpich *et al.* [128] discuss implementation of tool-supported distributed inspection at Lucent Technologies. The key idea in their new process was the *tool-supported asynchronous meeting*. They found out

that the number of identified defects was the same as in traditional inspections, but the new method reduced costs. Stein *et al.* [129] reported similar results. However, they discovered that traditional inspections were more effective in identifying certain types of defects. They emphasized also that meetings produce other benefits in addition to the mere identification of defects. They do not recommend replacing inspection meetings with asynchronous protocol without a good reason.

Vitharana & Ramamurthy [130] found out in their recent student experiment that anonymity in tool-supported inspection may affect inspection effectiveness. They compared groups with *anonymity tool-support* with groups where the participants knew each others' identities. The groups with anonymity support were more effective in identifying the seeded errors in relatively more complex tasks.

Comprehensive Views

This class includes the articles that discuss issues within many classes of our taxonomy. Only four papers were classified into this class, but they clearly formed a class of their own.

Our sample of articles included two real literature surveys. Laitenberger & DeBaud [7] have published an extensive *life cycle centric survey*, as mentioned earlier. It included a very comprehensive view of the inspection-related research but only dealt with the literature prior to 1998. Laitenberger and DeBaud provided a taxonomy of the completed inspection studies. The taxonomy contains technical, managerial, organizational, assessment, and tool dimensions. The taxonomy helps to relate inspection studies and approaches to particular life-cycle stages, to structure the inspection field, to compare and assess inspection methods, and to identify areas where little work has been done so far. They have also suggested causal models for explaining inspection quality, effort, and duration.

Aurum *et al.* [6] published another survey, which has a narrower scope; *inspection method variations*. They studied Fagan's original inspection method and its variations during the previous 25 years, but they did not focus on other aspects of inspection research. The survey included a classification of the studies using three dimensions: 1) Change of the basic inspection structure vs. support of the inspection process structure, 2) phases of the inspection process (preparation, meeting, reinspection), and 3) existence of the evaluative elements. The support includes models, methods, and tools, whereas evaluations relate to empirical testing of the inspection tools and techniques.

Additionally, Ciolkowski *et al.* [27] conducted a relatively extensive inspection study. Although it uses the term survey, it is not a literature survey, but an empirical study, which investigated software reviews and their *state of the practice in the industry*. There were over 200 involved respondents, of which about 30-40% conducted reviews regularly. The variation on using the reviews was due to the type of the inspected documents. Requirements were reviewed slightly more regularly than software code.

Another small-scale attempt to study the state of the practice was an informal empirical survey focusing on *reengineering inspections* as reported by Johnson [59]. There were 90 respondents, of which 80% practiced inspections irregularly or not at all. However, the real contribution of that paper was to consider how inspection practices should be improved in the light of the earlier research.

8. DISCUSSION

In this section we provide a brief synthesis of the main results of the surveyed studies and the suggested further research. The discussion is organized based on our research taxonomy.

Synthesis of the Results of the Studies

Technical View

There are several variations of inspection processes for different environments. For example, these variations are created for small projects, critical solutions, and distributed environments. In addition to the process aspect, a number of studies focus on different reading techniques. Several special techniques have been introduced to effectively find defects from requirements, design documents, or software code. Most of these studies have received promising results as compared to ad hoc reading or traditional checklists. However, a weakness related to the different processes and reading techniques is that there is, for example, no research comparing the different new reading techniques. The surveyed papers include some practical results of the different effectiveness factors of the inspection process. Several researchers agree that the performance of individual team members is the most important factor in inspection effectiveness. Preparation for the inspection meeting is obviously the most important phase in the inspection process. Some researchers even consider meetings as a waste of time, because most of the defects are found already during the preparation.

Management View

There is evidence of inspection benefits in several case studies in the surveyed papers. Fagan's [2] first experiences with inspections already included improvement in quality and productivity, when they replaced informal walkthroughs with inspections. The focus in the case reports is typically on presenting savings or return on investment (ROI) gained in implementing inspections in the company. Multiple organizations have reported ROI to be at least 30, and there was no papers in the selected publication series that would have questioned the overall inspection benefits. One topic discussed in several papers is the comparison between code review and different testing techniques. There is no clear evidence for the superiority of any single technique. Instead, different techniques are generally seen as complementary to each other.

Other Main Topics

Several aspects related to inspection tools have been covered in the surveyed papers. The introduced tools are primarily focused on inspection process support. For example, they usually support distributed asynchronous inspection, which frees the participants from specific meeting times and places. Defect estimation is an area, which has inspired a significant number of studies. The key idea is to estimate the number of defects remaining in the document after the inspection. Good

estimates would provide useful decision support for project management, but unfortunately the research on this area has not at least yet succeeded to fulfill the expectations.

Suggested Further Research

Technical View

Overall, the suggestions for the needed further research as presented earlier in the surveyed studies are heavily focused on the issues within the *technical view* class. Many researchers, including Aurum *et al.* [6], Laitenberger *et al.* [7], and Parnas & Lawford [131], have suggested further research related to the technical view, and especially on the *effectiveness factors*.

Laitenberger & DeBaud [7] identified the then relevant further research avenues. On the general level the goals included: Assurance of software quality, reduction of development costs, and keeping software projects within schedule. On the more specific level there were multiple issues. They included issues such as: Determining the most cost-effective inspection variations, the proper amount of effort to be allocated to inspections, the effects of the number and experience of the inspectors, the adequacy of different reading techniques, the effects of the inspected software artifact types, and the proper amount of the inspected materials.

The survey of Aurum *et al.* [6] similarly addressed the then most relevant oncoming research avenues. These included issues such as: Determining the efficiency of the different kinds of reading techniques, efficient ways of supporting inspections, measuring the effectiveness of inspections, possible synergy of inspections and testing, and possible differences of the proper form of inspections in case of different kinds of applications and systems such as object-oriented systems, real-time systems, and web-based systems. The nature of the further research issues suggested in [6, 7] is mainly conducting more research on the technical issues which have already been studied earlier in some extent.

Parnas and Lawford [131] considered the future of inspections and brought forth from the technical view point a need to *specify inspection processes*. By this they mainly mean the development of different kinds of reading techniques. It is noteworthy that studies on determining the adequacy and effectiveness of different reading techniques are suggested by many researchers, including Aurum *et al.* [6], Laitenberger & DeBaud [7], and Parnas & Lawford [131]. Thereby, the area of reading techniques still requires active further research.

These kinds of developments can and should be linked to the studies of inspection efficiency. A clear deficiency affecting these kinds of studies is the currently modest level of understanding issues related to practically applying inspections in industrial settings. Generally, the conducted studies have only considered factors affecting inspection efficiency in a limited context. It is, however, also necessary to understand what *environmental factors* mainly affect efficiency.

Management View

Laitenberger & DeBaud [7] have suggested more research on *scaling up inspections*, and Aurum *et al.* [6] on management and *inspection process control*. It is difficult to predict the orientation of future research based on earlier studies, but some observations can be made regarding weaknesses in the current body of knowledge. From the practical point of view, the biggest issue is still the relatively weak state of *implementing inspections in organizational settings*.

Organizations have not yet widely adopted inspection techniques and their *maturity level* for adopting those techniques probably still has to be much improved [16]. Implementation and management issues need to be studied more extensively since very few of the papers covered in this survey discuss them. So, this probably is the most important gap area in the current research and also in the suggestions of the needed further research in most of the earlier studies.

Other Main Topics

It is noteworthy that many researchers, including Aurum *et al.* [6], Laitenberger & DeBaud. [7], and Parnas & Lawford [131], have suggested that more *tool support* should be developed. Anderson *et al.* [125] have presented a tool that includes support for its users in identifying defects. However, generally it seems to be that interest in studying inspection tools has unfortunately rather diminished than increased during this decade. Nevertheless, there will undoubtedly be a need for tools in the future since they potentially enable improving both the efficiency and the attractiveness of inspections to commercial organizations.

For example, Xu [132] has presented a method that can be used to improve the inspectability of real-time systems during the system design phase. These kinds of *special questions* will undoubtedly be increasingly important in the future. After the basic processes of inspections have reached a mature level, a further improvement in efficiency can be attempted by also improving the understanding of the specificities of the *inspected document contents*. Currently there are only a few examples of this kind of research. Macdonald *et al.* [82] and Dunsmore *et al.* [83] have studied the specificities of inspecting object-oriented code.

Software engineering is an evolving discipline. Some changes in software engineering practices may affect the role of software inspections. For example, implementing *agile methods* or *domain-specific modelling* may lead to an extensive change in the entire software development process. Therefore, the relationship between inspections and these new development models have to be understood. Serious research on this field is only just emerging. For example, in our survey, Müller's [95, 96] papers were the only ones discussing agile methods.

9. CONCLUSIONS

This paper has reported a comprehensive survey of software inspection literature published in the selected high-impact publication series during the selected time period (i.e. 1980-2008). The survey includes most of the software inspection studies which have ever been published. It has fully covered 16 central publication series including 153 identified and studied articles. The paper has reported the following three main results related to the survey: 1) Trends in inspection research. 2) An emergent taxonomy of the inspection research. 3) A summary of the most important research results in each class of the taxonomy.

The survey revealed the continuously increasing trend of software inspection research till the recent years. During the early years (i.e. 1980-1993) the inspection research focused on different variations of inspection processes. However, the core of the original inspection ideas has not changed.

Later research has been strongly focused on different in-process details like reading techniques and different effectiveness factors. For some reason, the volume has significantly decreased during the four last years (i.e. 2005-2008). This might be mostly explained by the personal interests of some key researchers, who appear to have directed their work to other areas. However, there would still be a lot of important work to be done in the inspection research area.

Overall, the conducted research is relatively scattered relative to its volume. There are many adapted variants of the inspection methods for different needs. The studied materials present clear evidence of inspection benefits in several cases. Some understanding has been gained regarding several effectiveness factors such as team size and the proper amount of materials to be inspected. Also tool support for the inspection process has produced notable results. We thus have a fairly good understanding of how to run effective inspections in limited controlled research settings, in which most of the empirical studies have been conducted.

The real issue with inspections is still their weak implementation in the software industry. Only relatively few studies within our survey were related to the implementation and management of inspections in practice. Empirical studies are needed especially to better understand these issues. From our point of view this is the most important direction for future inspection research.

The gained body of knowledge creates a good basis for further studies. Now there is a need to transfer the results achieved in the controlled research settings to the industry and to study how inspections can be effectively implemented in different organizational environments.

SUMMARY (MAX. 50 WORDS)

This paper presents a comprehensive survey of the software inspection research focusing on high-impact scientific publication series. The main results include a description of the research trends during 1980-2008 and a description of the main results of the identified 153 articles.

APPENDIX 1: COMPLETE SURVEY CLASSIFICATION DATA

Table 4. Complete List of References in Each Research Class

Technical view (76)	Reading techniques (25)	Basili *et al.* (1996) [35], Berling & Runeson (2003) [133], Dunsmore *et al.* (2001;2002;2003) [43-45], Fusaro *et al.* (1997) [33], Hatton (2008) [49], Hungerford *et al.* (2004) [134], Höst & Johansson (2000) [135], Kelly & Shepard (2004) [46], Laitenberger & DeBaud (1997) [37], Laitenberger *et al.* (2000;2001) [38, 136], Maldonado *et al.* (2006) [36], Miller *et al.* (1998) [32], Porter *et al.* (1995) [30], Porter & Votta (1994;1998) [29, 31], Regnell *et al.* (2000) [137], Sabaliauskaite *et al.* (2003) [39], Sandahl (1998) [34], Thelin *et al.* (2001;2003;2004) [40, 41, 42], Zhang *et al.* (1999) [138].
	Effectiveness factors (23)	Biffl *et al.* (2001) [69], Biffl & Halling (2003) [139], Briand *et al.* (2004) [140], Carver *et al.* (2008) [70], Dunsmore *et al.* (2000) [141], Ebert *et al.* (2001) [58], Halling & Biffl (2002) [142], Johnson & Tjahjono (1997;1998) [143, 60], Kelly & Shepard (2004) [144], Laitenberger *et al.* (2002) [61], Land *et al.* (2000) [67], Miller & Yin (2004) [145], Porter *et al.* (1997;1997;1998) [52, 146, 67], Porter & Johnson (1997) [54], Porter & Votta (1997) [147], Raz & Yaung (1997) [148], Sabaliauskaite *et al.* (2004) [55], Sauer *et al.* (2000) [48], Seaman & Basili (1997;1998) [149, 64].
	Processes (22)	Ackerman *et al.* (1989) [150], d'Astous (2001) [151], Bias (1991) [152], Bisant & Lyle (1989) [72], van Emden (1992) [81], Fagan (1986) [9], Gantner & Barth (2003) [153], Jackson & Hoffman (1994) [79], Knight & Myers (1993) [51], Kosman (1997) [154], Kusumoto *et al.* (1998) [73], Martin ja Tsai (1990) [75], Meyer (2008) [155], Mishra & Mishra (2007) [77], Parnas & Weis (1985;1987) [28, 156], Polack (2001) [80], Rigby *et al.* (2008) [78], Runge (1982) [71], Schneider *et al.* (1992) [74], Thelin *et al.* (2004) [76], Weinberg ja Freedman (1984) [157].
	Other technical topics (6)	de Almeida *et al.* (2003) [87], Chernak (1996) [85], Dunsmore *et al.* (2000) [83], Macdonald *et al.* (1996) [82], Tervonen (1996) [84], Traore & Aredo (2004) [86].
Management view (27)	Inspection impact on the development process (14)	Basili & Selby (1987) [89], Bush (1990) [88], Doolan (1992) [3], Grady & Van Slack (1994) [4], Houdek *et al.* (2002) [92], Maranzano *et al.* (2005) [158], Müller (2004;2005) [95, 96], Roper *et al.* (1997) [93], Runeson *et al.* (2006) [94], Russel (1991) [5], So *et al.* (2002) [91], Weller (1993) [159], Zheng *et al.* (2006) [97].
	Other management topics (13)	Barnard & Price (1994) [98], Briand et al (1998) [100], Chaar *et al.* (1993) [105], Chatzigeorgiou & Antoniadis (2003) [102], Denger & Shull (2007) [106], Freimut *et al.* (2005) [101], Hall & Fenton (1996) [160], Harjumaa (2005) [107], Iisakka & Tervonen (1998) [108], Jacob & Pillai (2003) [104], Jalote & Haragopal (1998) [90], Kusumoto *et al.* (1996) [103], Madachy (1996) [99].
Other main topics (50)	Defect estimation (23)	Biffl (2000;2003) [161, 162], Biffl & Grossmann (2001) [114], Biffl & Gutjahr (2002) [163], Briand *et al.* (2000) [164], Cockram (2001) [165], Ebrahimi (1997) [166], Eick *et al.* (1992) [110], El Emam *et al.* (2000) [115], El Emam & Laitenberger (2001) [167], Miller (1999;2002) [168, 169], Padberg (2002) [112], Padberg *et al.* (2004) [170], Petersson *et al.* (2004) [113], Runeson & Wohlin (1998) [171], Thelin (2004) [116], Thelin & Runeson (2000;2002) [172, 173], Vander Wiel & Votta (1993) [111], Wohlin *et al.* (1995) [174], Wohlin & Runeson (1998) [175], Yin *et al.* (2004) [117].
	Inspection tools (16)	Anderson *et al.* (2003) [125], van Genuchten *et al.* (2001) [176], Johnson (1994) [120], Lanubile *et al.* (2003) [177], Macdonald & Miller (1997;1998;1999) [121, 126, 122], Mashayekhi *et al.* (1993) [119], Miller & Macdonald (2000) [178], Perpich *et al.* (1997) [128], Perry *et al.* (2002) [123], Stein *et al.* (1997) [129], Tervonen (1996) [124], Tervonen & Oinas-Kukkonen (1996) [179], Tyran & George (2002) [127], Vitharana & Ramamurthy (2003) [130].
	Comprehensive views (4)	Aurum *et al.* (2002) [6], Ciolkowski *et al.* (2003) [27], Johnson (1998) [59], Laitenberger & DeBaud (2000) [7].
	Unclassified topics (7)	d'Astous & Robillard (2000;2002) [57, 180], Carver ym. (2006) [181], Kazman & Bass (2002) [182], Kelly *et al.* (1992) [183], Näslund & Löwgren (1999) [184], Xu (2003) [132].

REFERENCES

[1] L. Osterweil, L. Clarke, R. DeMillo *et al.*, "Strategic directions in software quality", *ACM Computing Surveys*, vol. 28(4), pp. 738-750, 1996.

[2] M.E. Fagan, "Design and code inspection to reduce errors in program development", *IBM Systems Journal*, vol. 15(3), pp. 182-211, 1976.

[3] E. Doolan, "Experience with Fagan's inspection method", *Software Practice and Experience*, vol. 22(2), pp. 173-182, 1992.

[4] R. Grady, & T. Van Slack, "Key lessons in achieving widespread inspection use", *IEEE Software*, vol. 11(4), pp. 46-57, 1994.

[5] G.W. Russell, "Experience with inspection in ultra large-scale developments", *IEEE Software*, vol. 8(1), pp. 25-31, 1991.

[6] A. Aurum, H. Petersson, & C. Wohlin, "State-of-the-art: Software inspections after 25 years", *Software Testing, Verification and Reliability*, vol. 12(3), pp. 133-154, 2002.

[7] O. Laitenberger, & J.M. DeBaud, "An encompassing life cycle centric survey of software inspection", *Journal of Systems and Software*, vol. 50(1), pp. 5-31, 2000.

[8] A.A. Porter, H.P. Siy, & L.G. Votta, "A review of software inspections", *Advances in Computers*, vol. 42, pp. 39-76, 1996.

[9] M.E. Fagan, "Advances in software inspections" *IEEE Transactions on Software Engineering*, vol. 12(7), pp. 744-751, 1986.

[10] R. G. Ebenau, & S.H. Strauss. "*Software Inspection Process*", New York: McGraw-Hill, 1994.

[11] T. Gilb, & D. Graham, "*Software Inspection*", Wokingham, England: Addison-Wesley, 1993.

[12] K. Wiegers, *Peer Reviews in Software: A Practical Guide*. Boston: Addison-Wesley, 2002.

[13] P. Brereton, B. Kitchenham, D. Budgen *et al.*, "Lessons from applying the systematic literature review process within the software engineering domain", *Journal of Systems and Software*, vol. 80(4), pp. 571-583, 2007.

[14] S. Kollanus, "ICMM - Inspection capability maturity model", in IASTED International Conference on Software Engineering, pp. 372-377.

[15] S. Kollanus, "Issues in software inspection practices", in Product Focused Software Process Improvement - 6th International Conference, LNCS 3547, 2005, pp. 429-442.

[16] S. Kollanus, "Experiences from using ICMM in inspection process assessment", *Software Quality Journal*, 2009 (In press, available online).

[17] S. Kollanus, & J. Koskinen, "Software inspections in practice: Six case studies", in Product Focused Software Process Improvement - 7th International Conference, LNCS 4034, pp. 377-382, 2006.

[18] ISI, "ISI Web of Knowledge. Journal Citation Reports", JCR Science Edition, 2009, categories: Computer Science, Software Engineering (URL: http://www.isiknowledge. com/) [Accessed: Feb. 20, 2009].

[19] A. Abran, J.W. Moore, P. Bourque, *et al.* (Eds.), *SWEBOK: Guide to the Software Engineering Body of Knowledge*. IEEE Computer Society, 2004. (URL: http://www.swebok.org/htmlformat.html) [Accessed: Feb. 20, 2009].

[20] CiteSeer, "Estimated impact of publication venues in computer science (Based on CiteSeer database and DBLP)", 2009, (URL: http://citeseer.ist.psu.edu/impact.html) [Accessed: Feb. 20, 2009].

[21] DBLP, "Computer Science Bibliography - University of Trier", 2009, (URL: http://dblp.uni-trier.de/) [Accessed: Feb. 20, 2009].

[22] IEEE, "IEEE Xplore (Release 2.5)", 2009, (URL: http://ieeexplore.ieee.org) [Accessed: Feb. 20, 2009].

[23] ACM, "The ACM Digital Library", 2009, (URL: http://www.acm.org/dl/) [Accessed: Feb. 20, 2009].

[24] F. Shull, F. Lanubile, & V.R. Basili, "Investigating reading techniques for object-oriented framework learning", *IEEE Transactions on Software Engineering*, vol. 26(11), pp. 1101-1118, 2000.

[25] R. Chillarege, I.S. Bhandari, J.K. Chaar *et al.*, "Orthogonal defect classification - A concept for in-process measurements", *IEEE Transactions on Software Engineering*, vol. 18(11), pp. 943-956, 1992.

[26] V. Basili, & S. Green, "Software process evolution at the SEL", *IEEE Software*, vol. 11(4), pp. 58-66, 1994.

[27] M. Ciolkowski, O. Laitenberger, & S. Biffl, "Software reviews, the state of the practice", *IEEE Software*, vol. 20(6), pp. 46-51, 2003.

[28] D.L. Parnas, & D.M. Weiss, "Active design reviews: Principles and practices", In 10th International Conference on Software Engineering, pp. 132-136, 1985.

[29] A. Porter, & L.G. Votta, "An experiment to assess different defect detection methods for software requirements inspections", in Proceedings of the 16th International Conference on Software Engineering, pp. 103-112, 1994.

[30] A.A. Porter, L.G. Votta, & V.R. Basili, "Comparing detection methods for software requirements inspections: A replicated experiment", *IEEE Transactions on Software Engineering*, vol. 21(6), pp. 563-575, 1995.

[31] A. Porter, & L. Votta, "Comparing detection methods for software requirements inspections: A replication using professional subjects", *Empirical Software Engineering*, vol. 3(4), pp. 355-379, 1998.

[32] J. Miller, M. Wood, & M. Roper, "Further experiences with scenarios and checklists", *Empirical Software Engineering*, vol. 3(1), pp. 37-64, 1998.

[33] P. Fusaro, F. Lanubile, & G. Visaggio, "A replicated experiment to assess requirements inspection techniques", *Empirical Software Engineering*, vol. 2(1), pp. 39-57, 1997.

[34] K. Sandahl, O. Blomkvist, K. Karlsson, C. *et al.*, "An extended replication of an experiment for assessing methods for software requirements inspections", *Empirical Software Engineering*, vol. 3(4), pp. 327-354, 1998.

[35] V.R. Basili, S. Green, O. Laitenberger, F. *et al.*, "The empirical investigation of perspective-based reading", *Empirical Software Engineering*, vol. 1(2), pp. 133-164, 1996.

[36] J. Maldonado, J. Carver, F. Shull *et al.*, "Perspective-based reading: A replicated experiment focused on individual reviewer effectiveness. An empirical investigation", *Empirical Software Engineering*, vol. 11(1), pp. 119-142, 2006.

[37] O. Laitenberger, & J.-M. DeBaud, " Perspective-based reading of code documents at Robert Bosch GmbH", *Information and Software Technology*, vol. 39(11), pp. 781-791, 1997.

[38] O. Laitenberger, C. Atkinson, M. Schlich *et al.*, "An experimental comparison of reading techniques for defect detection in UML design documents", *Journal of Systems and Software*, vol. 53(2), pp. 183-204, 2000.

[39] G. Sabaliauskaite, F. Matsukawa, S. Kusumoto *et al.*, "Further investigations of reading techniques for object-oriented design inspection", *Information and Software Technology*, vol. 45(9), pp. 571-585, 2003.

[40] T. Thelin, P. Runeson, & B. Regnell, "Usage-based reading - An experiment to guide reviewers with use cases", *Information and Software Technology*, vol. 43(15), pp. 925-938, 2001.

[41] T. Thelin, P. Runeson, & C. Wohlin, "An experimental comparison of usage-based and checklist-based reading", *IEEE Transactions on Software Engineering*, vol. 29(8), pp. 687-704, 2003.

[42] T. Thelin, P. Runeson, C. Wohlin *et al.*, "Evaluation of usage-based reading - Conclusions after three experiments", *Empirical Software Engineering*, vol. 9(1-2), pp. 77-110, 2004.

[43] A. Dunsmore, M. Roper, & M. Wood, "Systematic object-oriented inspection technique", In Proceedings of the 23rd International Conference on Software Engineering, pp. 123-144, 2001.

[44] A. Dunsmore, M. Roper & M. Wood, "Further investigations into the development and evaluation of reading techniques for object-oriented code inspection", In Proceedings of the 24th International Conference on Software Engineering, pp. 47-57, 2002.

[45] A. Dunsmore, M. Roper & M. Wood, "The development and evaluation of three diverse techniques for object-oriented code inspection", *IEEE Transactions on Software Engineering*, vol. 29(8), pp. 677-686, 2003.

[46] D. Kelly, & T. Shepard, "Task-directed software inspection", *Journal of Systems and Software*, vol. 73(2), pp. 361-368, 2004.

[47] L. Votta, "Does every inspection need a meeting?", *ACM Software Engineering Notes*, vol. 18(5), pp. 107-114, 1993.

[48] C. Sauer, D.R. Jeffery, L. Land *et al.*, "The effectiveness of software development technical reviews: A behaviorally motivated program of research", *IEEE Transactions on Software Engineering*, vol. 26(1), pp. 1-14, 2000.

[49] L. Hatton, "Testing the value of checklists in code inspections", *IEEE Software*, vol. 25(4), pp. 82-88, 2008.

[50] B.W. Boehm, *Software Engineering Economics*. Englewood Cliffs, NJ: Prentice-Hall, 1981.

[51] J.C Knight, & E.A. Myers, "An improved inspection technique", *Communications of the ACM*, vol. 36(11), pp. 51-61, 1993.

[52] A.A. Porter, H.P. Siy, C.A. Toman *et al.*, "An experiment to assess the cost-benefits of code inspections in large scale software development", *IEEE Transactions on Software Engineering*, vol. 23(6), pp. 329-346, 1997.

[53] A. Bianchi, F. Lanubile, F.G. Visaggio, "A controlled experiment to assess the effectiveness of inspection meetings", in International Symposium on Software Metrics, pp. 42-50, 2001.

[54] A.A. Porter, & P.M. Johnson, "Assessing software review meetings: Results of a comparative analysis of two experimental studies", *IEEE Transactions on Software Engineering*, vol. 23(3), pp. 129-145, 1997.

[55] G. Sabaliauskaite, S. Kusumoto, & K. Inoue, "Assessing defect detection performance of interacting teams in object-oriented design inspection", *Information and Software Technology*, vol. 46(13), pp. 875-886, 2004.

[56] B.A. Kitchenham, S.L. Pfleeger, L.M. Pickard *et al.*, "Preliminary guidelines for empirical research in software engineering", *IEEE Transactions on Software Engineering*, vol. 28(8), pp. 721-734, 2002.

[57] P. d'Astous, & P.N. Robillard, "Characterizing implicit information during peer review meetings", In: Proceedings of the 22nd International Conference on Software Engineering, pp. 460-466, 2000.

[58] C. Ebert, C.H. Parro, R. Suttels *et al.*, "Improving validation activities in a global software development", *In Proceedings of the 23rd International Conference on Software Engineering*, pp. 545-554, 2001.

[59] P.M. Johnson, "Reengineering inspection", *Communications of the ACM*, vol. 41(2), pp. 49-52, 1998.

[60] P.M. Johnson, & D. Tjahjono, "Does every inspection really need a meeting?", *Empirical Software Engineering*, vol. 3(1), pp. 9-35, 1998.

[61] O. Laitenberger, T. Beil, & T. Schwinn, "An industrial case study to examine a non-traditional inspection implementation for requirements specifications", *Empirical Software Engineering*, vol. 7(4), pp. 345-374, 2002.

[62] D.A. Christenson, S.T. Huang, & A.J. Lamperez, "Statistical quality control applied to code inspections", *IEEE Journal of Selected Areas of Communication*, vol. 8(2), pp. 196-200, 1990.

[63] A. Dunsmore, "*Survey of Object-oriented Defect Detection Approaches and Experience in Industry*", Technical Report - EFoCS-37-2000, University of Strathclyde, Scotland, UK, July 2000.

[64] C. Seaman, & V. Basili, "Communication and organization: An empirical study of discussion in inspection meetings", *IEEE Transactions on Software Engineering*, vol. 24(7), pp. 559-572, 1998.

[65] K.V. Bourgeois, "Process insights from a large-scale software inspections data analysis", *CrossTalk*, Oct. 1996.

[66] S. Rifkin, & L. Deimel, "Applying program comprehension techniques to improve software inspections", In: Proceedings of the 19th Annual NASA Software Engineering Workshop, pp. 115-126, 1994.

[67] L. Land, C. Sauer & R. Jeffery, "The use of procedural roles in code inspections: An experimental study", *Empirical Software Engineering*, vol. 5(1), pp. 11-34, 2000.

[68] A. Porter, H. Siy, A. Mockus *et al.*, "Understanding the sources of variation in software inspections", *ACM Transactions on Software Engineering and Methodology*, vol. 7(1), pp. 41-79, 1998.

[69] S. Biffl, B. Freimut, & O. Laitenberger, "Investigating the cost-effectiveness of reinspections in software development", in Proceedings of the 23rd International Conference on Software Engineering, 2001, 155-164.

[70] J. Carver, N. Nagappan, & A. Page, "The impact of educational background on the effectiveness of requirements inspections: An empirical study", *IEEE Transactions on Software Engineering*, vol. 34(6), pp. 800-812, 2008.

[71] B. Runge, "The Inspection Method applied to small projects", In: Proceedings of the 6th International Conference on Software Engineering, 416-417, 1982.

[72] D.B. Bisant, & J.R. Lyle, "A two-person inspection method to improve programming productivity", *IEEE Transactions on Software Engineering*, vol. 15(10), pp. 1294-1304, 1989.

[73] S. Kusumoto, A. Chimura, T. Kikuno, *et al.*, "A promising approach to two-person software review in educational environment", *Journal of Systems and Software*, vol. 40(2), pp. 115-123, 1998.

[74] M. Schneider, J. Martin, & W. Tsai, "An experimental study of fault detection in user requirements documents", *ACM Transactions on Software Engineering and Methodology*, vol. 1(2), pp. 188-204, 1992.

[75] J. Martin, & W.T. Tsai, "N-fold inspection: A requirements analysis technique", *Communications of the ACM*, vol. 33(2), pp. 225-232, 1990.

[76] T. Thelin, H. Petersson, P. Runeson *et al.*, "Applying sampling to improve software inspections", *Journal of Systems and Software*, vol. 73(2), pp. 257-269, 2004.

[77] D. Mishra, & A. Mishra, "Efficient software review process for small and medium enterprises", *IET Software*, vol. 1(4), pp. 132-142, 2007.

[78] P. Rigby, D. German & M. Storey, "Open source software peer review practices: A case study of the Apache server", In: Proceedings of the 30th International Conference on Software Engineering, pp. 541-550, 2008.

[79] A. Jackson, & D. Hoffman, "Inspecting module interface specifications", *Software Testing, Verification, and Reliability*, vol. 4(2), pp. 133-153, 1994.

[80] F. Polack, "A case study using lightweight formalism to review an information system specification", *Software: Practice and Experience*, vol. 31(8), pp. 757-780, 2001.

[81] M. van Emden, "Structured inspections of code", *Software Testing, Verification, and Reliability*, vol. 2(2), pp. 133-153, 1992.

[82] F. Macdonald, J. Miller, A. Brooks *et al.*, "Applying inspection to object-oriented code", *Software Testing, Verification and Reliability*, vol. 6(2), pp. 61-82, 1996.

[83] A. Dunsmore, M. Roper, & M. Wood, "Object-oriented inspection in the face of delocalization", In: Proceedings of the 22nd International Conference on Software Engineering, pp. 467-476, 2000.

[84] I. Tervonen, "Consistent support for software designers and inspectors", *Software Quality Journal*, vol. 5(3), pp. 221-229, 1996.

[85] Y. Chernak, "A statistical approach to the inspection checklist - Formal synthesis and improvement", *IEEE Transactions on Software Engineering*, vol. 22(12), pp. 866-874, 1996.

[86] I. Traore, & D.B. Aredo, "Enhancing structured review with model-based verification", *IEEE Transactions on Software Engineering*, vol. 30(11), pp. 736-753, 2004.

[87] J.R. de Almeida, J.B. Camargo, B.A. Basseto *et al.*, "Best practices in code inspection for safety-critical software", *IEEE Software*, vol. 20(3), pp. 56-63, 2003.

[88] M. Bush, "Improving software quality: The use of formal inspections at the JPL", In: Proceedings of the 14th International Conference on Software Engineering, pp. 196-199, 1990.

[89] V.R. Basili & R. Selby, "Comparing the effectiveness of software testing strategies", *IEEE Transactions on Software Engineering*, vol. 13(12), pp. 1278-1296, 1987.

[90] P. Jalote & M. Haragopal, "Overcoming the NAH syndrome for inspection deployment", in Proceedings of the 20th International Conference on Software Engineering, 1998, pp. 371-378.

[91] S. So, S. Cha, T. Shimeall, *et al.*, "An empirical evaluation of six methods to detect faults in software", *Software Testing, Verification and Reliability*, vol. 12(3), pp. 155-171, 2002.

[92] F. Houdek, T. Schwinn, & D. Ernst, "Defect detection for executable specifications - an experiment", *International Journal of Software Engineering and Knowledge Engineering*, vol. 12(6), pp. 637-655, 2002.

[93] M. Roper, M. Wood, & J. Miller, "An empirical evaluation of defect detection techniques", *Information and Software Technology*, vol. 39(11), pp. 763-775, 1997.

[94] P. Runeson, C. Andersson, T. Thelin *et al.*, "What do we know about defect detection methods?", *IEEE Software*, vol. 23(3), pp. 82- 90, 2006.

[95] M. Müller, "Are reviews an alternative to pair programming?", *Empirical Software Engineering*, vol. 9(4), pp. 335-351, 2004.

[96] M. Müller, "Two controlled experiments concerning the comparison of pair programming to peer review", *Journal of Systems and Software*, vol. 78(2), pp. 166-179, 2005.

[97] J. Zheng, L. Williams, N. Nagappan *et al.*, "On the value of static analysis for fault detection in software", *IEEE Transactions on Software Engineering*, vol. 32(4), pp. 240-253, 2006.

[98] J. Barnard, & A. Price, "Managing code inspection information", *IEEE Software*, vol. 11(2), pp. 59-69, 1994.

[99] R. Madachy, "System dynamics modeling of an inspection-based process", In: Proceedings of the 18th International Conference on Software Engineering, pp. 376-386, 1996.

[100] L. Briand, K. El Emam, O. Laitenberger, *et al.*, "Using simulation to build inspection efficiency benchmarks for development projects", In: Proceedings of the 20th International Conference on Software Engineering, pp. 340-349, 1998.

[101] B. Freimut, L.C. Briand, & F. Vollei, "Determining inspection cost-effectiveness by combining project data and expert opinion", *IEEE Transactions on Software Engineering*, vol. 31(12), pp. 1074-1092, 2005.

[102] A. Chatzigeorgiou, & G. Antoniadis, "Efficient management of inspections in software development projects", *Information and Software Technology*, vol. 45(10), pp. 671-680, 2003.

[103] S. Kusumoto, T. Kikuno, K. Matsumoto e al.,"Experimental evaluation of time allocation procedure for technical reviews", *Journal of Systems and Software*, vol. 35(2), pp. 119-126, 1996.

[104] A.L. Jacob, & S.K. Pillai, "Statistical process control to improve coding and code review", *IEEE Software*, vol. 20(3), pp. 50-55, 2003.

[105] J.K. Chaar, M.J. Halliday, I.S. Bhandari, *et al.*, "In-process evaluation for software inspection and test", *IEEE Transactions on Software Engineering*, vol. 19(11), pp. 1055-1070, 1993.

[106] C. Denger, & F. Shull, "A Practical Approach for Quality-Driven Inspections", *IEEE Software*, vol. 24(2), pp. 79-86, 2007.

[107] L. Harjumaa, "A pattern approach to software inspection process improvement", *Software Process: Improvement and Practice*, vol. 10(4), pp. 455-465, 2007.

[108] J. Iisakka, & I. Tervonen, "Painless improvements to the review process", *Software Quality Journal*, vol. 7(1), pp. 11-20, 1998.

[109] L. Harjumaa, I. Tervonen, & P. Vuorio, "Using software inspection as a catalyst for SPI in a small company", In: Proceedings of the 5th International Conference on Product Focused Software Process Improvement, LNCS 3009, pp. 62-75, 2004.

[110] S.G. Eick, C.R. Loader, M.D. Long, *et al.*, "Estimating software fault content before coding", In: Proceedings of the 14th International Conference on Software Engineering, 1992, pp. 59-65.

[111] S. Vander Wiel, & L. Votta, "Assessing software designs using capture-recapture methods", *IEEE Transactions on Software Engineering*, vol. 19(11), pp. 1045-1054, 1993.

[112] F. Padberg, "Empirical interval estimates for the defect content after an inspection", In: Proceedings of the 24th International Conference on Software Engineering, 2002, pp. 58-68.

[113] H. Petersson, T. Thelin, P. Runeson *et al.*, "Capture–recapture in software inspections after 10 years research - Theory, evaluation and application", *Journal of Systems and Software*, vol. 72(2), pp. 249-264, 2004.

[114] S. Biffl, & W. Grossmann, "Evaluating the accuracy of defect estimation models based on inspection data from two inspection cycles", In: Proceedings of the 23rd International Conference on Software Engineering, 2001, pp. 145-154.

[115] K. El Emam, O. Laitenberger & T. Harbich, "The application of subjective estimates of effectiveness to controlling software inspections", *Journal of Systems and Software*, vol. 54(2), pp. 119-136, 2000.

[116] T. Thelin, "Team-based fault content estimation in the software inspection process", in Proceedings of the 26th International Conference on Software Engineering, 2004, pp. 263-272.

[117] Z. Yin, A. Dunsmore & J. Miller, "Self-assessment of performance in software inspection processes", *Information and Software Technology*, vol. 46(3), pp. 185-194, 2004.

[118] H. Hedberg, "Introducing the next generation of software inspection tools", in Proceedings of the 5th International Conference on Product Focused Software Process Improvement, LNCS 3009, 2004, pp. 234-247.

[119] V. Mashayekhi, J.M. Drake, W.-T. Tsai, *et al.*, "Distributed, collaborative software inspection", *IEEE Software*, vol. 10(5), pp. 66-75. 1993.

[120] P.M. Johnson, "An instrumented approach to improving software quality through formal technical review", in Proceedings of the 16th International Conference on Software Engineering, 1994, pp. 113-122.

[121] F. Macdonald, & J. Miller, "A software inspection process definition language and prototype support tool", *Software Testing, Verification and Reliability*, vol. 7(2), pp. 99-128, 1997.

[122] F. Macdonald & J. Miller, "ASSIST - A tool to support software inspection", *Information and Software Technology*, vol. 41(15), pp. 1045-1057, 1999.

[123] D.E. Perry, A. Porter, M.W. Wade *et al.*, "Reducing inspection interval in large-scale software development", *IEEE Transactions on Software Engineering*, vol. 28(7), pp. 695-705, 2002.

[124] I. Tervonen, "Support for quality-based design and inspection", *IEEE Software*, vol. 13(1), pp. 44-54, 1996.

[125] P. Anderson, T. Reps, & T. Teitelbaum, "Design and implementation of a fine-grained software inspection tool", *IEEE Transactions on Software Engineering*, vol. 29(8), pp. 721-733, 2003.

[126] F. Macdonald, & J. Miller, "A comparison of tool-based and paper-based software inspection", *Empirical Software Engineering*, vol. 3(3), pp. 233-253, 1998.

[127] C.K. Tyran, & J.F. George, "Improving software inspections with group process support", *Communications of the ACM*, vol. 45(9), pp. 87-92, 2002.

[128] J.M. Perpich, D.E. Perry, A.A. Porter, *et al.*, "Anywhere, anytime code inspections: Using the web to remove inspection bottlenecks in large-scale software development", in Proceedings of the 19th International Conference on Software Engineering, 1997, pp. 14-21.

[129] M. Stein, J. Riedl, S. Harner, *et al.*, "A case study of distributed, asynchronous software inspection", in Proceedings of the 19th International Conference on Software Engineering, 1997, pp. 107-117.

[130] P. Vitharana, & K. Ramamurthy, "Computer-mediated group support, anonymity, and the software inspection process: An empirical investigation", *IEEE Transactions on Software Engineering*, vol. 29(2), pp. 167-180, 2003.

[131] D.L. Parnas, & M. Lawford, "The role of inspection in software quality assurance", *IEEE Transactions on Software Engineering*, vol. 29(8), pp. 674-676, 2003.

[132] J. Xu, "On inspection and verification of software with timing requirements", *IEEE Transactions on Software Engineering*, vol. 29(8), pp. 705-720, 2003.

[133] T. Berling, & P. Runeson, "Evaluation of a perspective based review method applied in an industrial setting", *IEE Proceedings - Software*, vol. 150(3), pp. 177-184, 2003.

[134] B. Hungerford, A. Hevner, & R. Collins, "Reviewing software diagrams: A cognitive study", *IEEE Transactions on Software Engineering*, vol. 30(2), pp. 82-96, 2004.

[135] M. Höst, & C. Johansson, "Evaluation of code review methods through interviews and experimentation", *Journal of Systems and Software*, vol. 52(2-3), pp. 113-120, 2000.

[136] O. Laitenberger, K. El Emam, & T.G. Harbich, "An internally replicated quasi-experimental comparison of checklist and perspective based reading of code documents", *IEEE Transactions on Software Engineering*, vol. 27(5), pp. 387-421, 2001.

[137] B. Regnell, P. Runeson, & T. Thelin, "Are the perspectives really different? – Further experimentation on scenario-based reading of requirements", *Empirical Software Engineering*, vol. 5(4), pp. 331-356, 2000.

[138] Z. Zhang, V. Basili, & B. Shneiderman, "Perspective-based usability inspection: An empirical validation of efficacy", *Empirical Software Engineering*, vol. 4(1), pp. 43-69, 1999.

[139] S. Biffl, & M. Halling, "Investigating the defect detection effectiveness and cost benefit of nominal inspection teams", *IEEE Transactions on Software Engineering*, vol. 29(5), pp. 385-397, 2003.

[140] L. Briand, B. Freimut, & F. Vollei, "Using multiple adaptive regression splines to support decision making in code inspections", *Journal of Systems and Software*, vol. 73(2), pp. 205-217, 2004.

[141] A. Dunsmore, M. Roper, & M. Wood, "The role of comprehension in software inspection", *Journal of Systems and Software*, vol. 52(2-3), pp. 121-129, 2000.

[142] M. Halling, & S. Biffl, "Investigating the influence of software inspection process parameters on inspection meeting performance", *IEE Proceedings – Software*, vol. 149(5), pp. 115-121, 2002.

[143] P. Johnson, & D. Tjahjono, "Assessing software review meetings: A controlled experimental study using CSRS", in 19th International Conference on Software Engineering, 1997, pp. 118-127.

[144] D. Kelly, & T. Shepard, "Eight maxims for software inspectors", *Software Testing, Verification and Reliability*, vol. 14(4), pp. 243-256, 2004.

[145] J. Miller, & Z. Yin, "A cognitive-based mechanism for constructing software inspection teams", *IEEE Transactions on Software Engineering*, vol. 30(11), pp. 811-825, 2004.

[146] A. Porter, H. Siy. & L. Votta, "Understanding the effects of developer activities on inspection interval", in Proceedings of the 19th International Conference on Software Engineering, 1997, pp. 128-138.

[147] A. Porter, & L. Votta, "What makes inspections work?", *IEEE Software*, vol. 14(6), pp. 99-102, 1997.

[148] T. Raz, & A. Yaung, "Factors affecting design inspection effectiveness in software development", *Information and Software Technology*, vol. 39(4), pp. 297-305, 1997.

[149] C. Seaman, & V. Basili, "An empirical study of communication in code inspections", in Proceedings of the 19th International Conference on Software Engineering, 1997, pp. 96-106.

[150] A.F. Ackerman, L.S. Buchwald,, & F.H. Lewski, "Software inspections: An effective verification process", *IEEE Software*, vol. 6(3), pp. 31-36, 1989.

[151] P. d'Astous, P. Robillard, F. Détienne *et al.*, "Quantitative measurements of the influence of participant roles during peer review meetings", *Empirical Software Engineering*, vol. 6(2), pp. 143-159, 2001.

[152] R. Bias, "Interface-walkthroughs: Efficient collaborative testing", *IEEE Software*, vol. 8(5), pp. 94-95, 1991.

[153] T. Gantner, & T. Barth, "Experiences on defining and evaluating an adapted review process", in Proceedings of the 25th International Conference on Software Engineering, 2003, 506-511.

[154] R. Kosman, "A two-step methodology to reduce requirement defects", *Annals of Software Engineering*, vol. 3, pp. 477-494, 1997.

[155] B. Meyer, "Design and code reviews in the age of the internet", *Communications of the ACM*, vol. 51(9), pp. 66-71, 2008.

[156] D.L. Parnas, & D.M. Weis, "Active design reviews: Principles and practices", *Journal of Systems and Software*, vol. 7(4), pp. 259-265, 1987.

[157] G.M. Weinberg, & D.P. Freedman, "Reviews, walkthroughs, and inspections", *IEEE Transactions on Software Engineering*, vol. 10(1), pp. 68-72, 1984.

[158] J. Maranzano, S. Rozsypal, G. Zimmerman *et al.*, "Architecture reviews: Practice and experience", *IEEE Software* 22(2), pp. 34-43, 2005.

[159] E. Weller, "Lessons from three years of inspection data", *IEEE Software*, vol. 10(5), pp. 38-45, 1993.

[160] T. Hall, & N. Fenton, "Software quality programmes: A snapshot of theory versus reality", *Software Quality Journal*, vol. 5(4), pp. 235-242, 1996.

[161] S. Biffl, "Using inspection data for defect estimation", *IEEE Software*, vol. 17(6), pp. 36-43, 2000.

[162] S. Biffl, "Evaluating defect estimation models with major defects", *Journal of Systems and Software*, vol. 65(1), pp. 13-29, 2003.

[163] S. Biffl, & W. Gutjahr, "Using a reliability growth model to control software inspection", *Empirical Software Engineering*, vol. 7(3), pp. 257-284, 2002.

[164] L. Briand, K. El Emam, B. Freimut *et al.*, "A comprehensive evaluation of capture-recapture models for estimating software defect content", *IEEE Transactions on Software Engineering*, vol. 26(6), pp. 518-540, 2000.

[165] T. Cockram, "Gaining confidence in software inspection using a Bayesian belief model", *Software Quality Journal*, vol. 9(1), pp. 31-42, 2001.

[166] N. Ebrahimi, "On the statistical analysis of the number of errors remaining in a software design document after inspection", *IEEE Transactions on Software Engineering*, vol. 23(8), pp. 529-532, 1997.

[167] K. El Emam, & O. Laitenberger, "Evaluating capture-recapture models with two inspectors", *IEEE Transactions on Software Engineering*, vol. 27(9), pp. 851-864, 2001.

[168] J. Miller, "Estimating the number of remaining defects after inspection", *Software Testing, Verification and Reliability*, vol. 9(3), pp. 167-189, 1999.

[169] J. Miller, "On the independence of software inspectors", *Journal of Systems and Software*, vol. 60(1), pp. 5-10, 2002.

[170] F. Padberg, T. Ragg, & R. Schoknecht, "Using machine learning for estimating the defect content after an inspection", *IEEE Transactions on Software Engineering*, vol. 30(1), pp. 17-28, 2004.

[171] P. Runeson, & C. Wohlin, "An experimental evaluation of an experience-based capture-recapture method in software code inspections", *Empirical Software Engineering*, vol. 3(4), pp. 381-406, 1998.

[172] T. Thelin, & P. Runeson, "Robust estimations of fault content with capture–recapture and detection profile estimators", *Journal of Systems and Software*, vol. 52(2-3), pp. 139-148, 2000.

[173] T. Thelin, & P. Runeson, "Confidence intervals for capture–recapture estimations in software inspections", *Information and Software Technology*, vol. 44(12), pp. 683-702, 2002.

[174] C. Wohlin, P. Runeson, & J. Brantestam, "An experimental evaluation of capture-recapture in software inspections", *Software Testing, Verification, and Reliability*, vol. 5(4), pp. 213-232, 1995.

[175] C. Wohlin, & P. Runeson, "Defect content estimations from review data", in Proceedings of the 20th International Conference on Software Engineering, 1998, pp. 371-378.

[176] M. van Genuchten, C. van Dijk, H. Scholten *et al.*, "Using group support systems for software inspections", *IEEE Software*, vol. 18(3), pp. 60-65, 2001.

[177] F. Lanubile, T. Mallardo, & F. Calefato, "Tool support for geographically dispersed inspection teams", *Software Process: Improvement and Practice*, vol. 8(4), pp. 217-231, 2003.

[178] J. Miller, & F. Macdonald, "An empirical incremental approach to tool evaluation and improvement", *Journal of Systems and Software*, vol. 51(1), pp. 19-35, 2000.

[179] I. Tervonen, & H. Oinas-Kukkonen, "Reorganizing the inspection process: Problems encountered and resolved", *Software Process: Improvement and Practice*, vol. 2(2), pp. 97-110, 1996.

[180] P. d'Astous, & P. Robillard, "Empirical study of exchange patterns during software peer review meetings", *Information and Software Technology*, vol. 44(11), pp. 639-648, 2002.

[181] J. Carver, F. Shull, & V. Basili, "Can observational techniques help novices overcome the software inspection learning curve? An empirical investigation", *Empirical Software Engineering*, vol. 11(4), pp. 523-539, 2006.

[182] R. Kazman, & L. Bass, "Making architecture reviews work in the real world", *IEEE Software*, vol. 19(1), pp. 67-73, 2002.

[183] J. Kelly, J. Sherif, & J. Hops, "An analysis of defect densities found during software inspections", *Journal of Systems and Software*, vol. 17(2), pp. 111-117, 1992.

[184] T. Näslund, & J. Löwgren, "Usability inspection in contract-based systems development - A contextual assessment", *Journal of Systems and Software*, vol. 45(3), pp. 233-240, 1999.

Received: March 16, 2009 Revised: April 14, 2009 Accepted: April 16, 2009