Guidance for Efficiently Implementing Defect Causal Analysis

Marcos Kalinowski [1, 2], Guilherme H. Travassos [1], David N. Card [3]

[1] COPPE/UFRJ – Federal University of Rio de Janeiro
Caixa Postal 68511 – CEP 21.945-970 – Rio de Janeiro – Brazil

[2] Bennett – Methodist University of Rio de Janeiro
Rua Marquês de Abrantes, 55 - Flamengo - CEP 22.230-060 – Rio de Janeiro – Brazil

[3] Det Norske Veritas - 115 Windward Way - Indian Harbour, FL 32937 - USA

{mkali, ght}@cos.ufrj.br , card@computer.org

Abstract. *Defect causal analysis has shown itself to be a cheap and high return means of product-focused software process improvement. However, despite its advantages and wide industry adoption little academic research is being done in this area. Thus, professionals face several questions when implementing it in software organizations. Aiming to provide unbiased and evidence-based answers to those questions, a systematic review has been conducted. Based on the results of the systematic review, better guidance for implementing defect causal analysis efficiently in software organizations can be elaborated.*

1. Introduction

Causal analysis and resolution is a means of identifying causes of defects and other problems and taking action to prevent them from occurring in the future. It is considered in many software process improvement models and approaches, such as MPS [SOFTEX, 2007a], CMMI [SEI, 2006], ISO/IEC 12207 [ISO/IEC, 2004], Lean [Poppendieck and Poppendieck, 2003], and Six Sigma [Eckes, 2000]. Robitaille (2004) highlights causal analysis as a way to identify opportunities for improving organizational process assets based on experience with the projects' defined processes.

Defect causal analysis (or defect prevention[1]), comprises applying causal analysis and resolution to a specific type of problem: the defects introduced in software artifacts throughout the software lifecycle. Given this initial definition, defect causal analysis can be seen as a systematic process to identify and analyze causes associated with the occurrence of specific defect types, allowing the identification of improvement opportunities for organizational process assets and the implementation of actions to prevent the occurrence of that same defect type in future projects. Thus defect causal analysis provides an opportunity to enable product-focused software process improvement, based on data about the products' defects [Kalinowski, 2007]. Other problems, such as schedule flaws, are not considered directly by defect causal analysis.

Accomplishing defect causal analysis activities throughout the software development lifecycle has shown to reduce defect rates by over fifty percent in different organizational contexts, such as IBM [Mays et al., 1990], Computer Science Corporation [Dangerfield et al., 1992 apud Card, 1993], HP [Grady, 1996], and InfoSys

[1] More precisely, defect causal analysis can be considered part of defect prevention. The latter also addresses the implementation of actions and the communication of changes to the development team explicitly.

[Jalote and Agrawal, 2005]. As a consequence, it helps to diminish the rework effort [Jalote and Agrawal, 2005] and increases the probability of achieving other process quality and performance goals [SEI, 2006]. Moreover, defect causal analysis is a means for communicating lessons learned among projects [SEI, 2006].

However, as mentioned by Card (2005), despite its benefits and the wide industry adoption of causal analysis, little research is being done in this area, and little scientific knowledge has been generated and published. Thus many doubts and questions remain when implementing defect causal analysis in software organizations. For instance: Is my organization ready for defect causal analysis? What approach[2] should be followed? How should defects be categorized? How should causes be categorized? What are the expected costs and results of implementing defect causal analysis?

Aiming to provide unbiased and evidence-based answers to these questions, a systematic review has been conducted. This systematic review identifies the defect causal analysis state of the art and provides guidance on how to efficiently implement it in software organizations. The systematic review and the resulting guidance are the focus of this paper. Further information and details on the systematic review can be found in a technical report [Kalinowski and Travassos, 2008] that supports this paper.

The remainder of this paper is organized as follows. In section 2 the systematic review is described. In section 3 the guidance and suggestions obtained from analyzing the results of the systematic review are presented in the form of evidence-based and unbiased answers to common questions. Section 4 concludes the paper.

2. A Systematic Review on Defect Causal Analysis

A systematic review is a means of identifying, evaluating and interpreting research relevant to research questions in an unbiased and fair way [Brereton et al., 2007]. Systematic reviews tend to be more reliable because it makes use of a rigorous methodology that is open to auditing and replication.

A systematic review was conducted to identify the state of the art regarding defect causal analysis and to provide guidance on how to effectively implement defect causal analysis in software organizations. An unbiased review protocol, focusing on the research questions was developed to guide the literature review according to the systematic review process. This process involves three main activities: planning, execution and result analysis [Biolchini et al., 2005].

The next subsection presents an introduction to defect causal analysis, in order to provide the basis for understanding of the remaining subsections, in which a description of the systematic review activity and results is presented.

2.1. Defect Causal Analysis

To have a clear understanding of what defect causal analysis represents in the scope of this paper it is important to understand precisely what we mean by the term defect, since its interpretation often depends on the context in which it is being used. For instance, when a defect is found through peer reviews it is related to a fault in the artifact being reviewed. When a defect is found through testing activities, on the other hand, usually it is related to a failure in the software product being tested. These definitions follow the IEEE standard terminology for software defects [IEEE 610.12, 1990]:

[2] Hereafter we refer to "defect causal analysis approach" as a strategy for an investigative process and to "defect causal analysis techniques" as a specific tool or method used in a defect causal analysis process.

- Error: a mistake committed by a person while trying to understand given information, solve a problem, or using a method or tool.
- Fault: the concrete manifestation of an error in a software artifact. An error may result in many faults.
- Failure: the operational behavior different from the one expected by the user. A failure may be caused by many faults, one fault may cause many failures, and some faults may never cause failures.

This paper uses the term defect to represent the IEEE definition of fault. Thus, in the case of failures it will be necessary to find the related defects (faults) by analyzing the artifacts (for instance, by debugging source code) before starting defect causal analysis.

Analyzing causes of software defects has been discussed since the seventies [Endres, 1975] and is cited by Boehm (2006), in addition to software inspections, as one of the main contributions of that decade to software engineering. Since then, defect causal analysis processes have been implemented in industry for large projects involving hundreds of employees [Mays et al., 1990] [Leszak et al., 2002] as well as smaller projects [Dangerfield et al., 1992] [Yu, 1998] [Jalote and Agrawal, 2005].

Card (2005) summarizes the defect causal analysis process in six steps: (1) select a sample of the defects, (2) classify selected defects, (3) identify systematic errors, (4) determine principal cause, (5) develop action proposals, and (6) document meeting results. In this context a systematic error is an error that results in the same or similar defects being repeated in different occasions [Card, 2005]. Finding systematic errors indicates the existence of significant improvement opportunities for the project or organizational process assets. Besides listing these six steps, Card (2005) highlights the importance of managing the implementation of the action proposals until their conclusion and communicating the implemented changes to the development team.

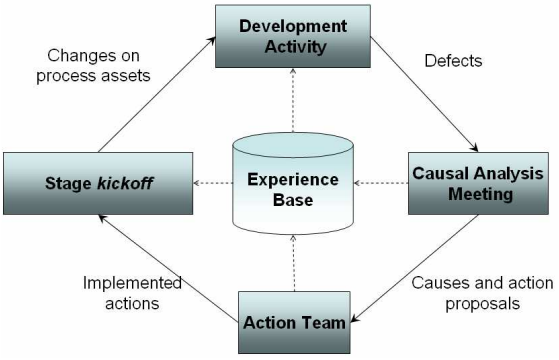


Figure 2. Defect prevention process. Adapted from [Mays et al, 1990 apud Rombach and Endres, 2003].

A representation of the traditional software defect prevention process [Mays et al., 1990], consistent with the defect causal analysis process described above, is shown in Figure 2. Besides the causal analysis meeting, the figure shows a specific activity for implementing the action proposals by an action team and the communication of the implemented changes before starting the development activity. Moreover, the position of the experience base shows that defect causal analysis is a mean for communicating lessons learned among projects as well as a way to disseminate knowledge in the organization, as suggested by the SEI (2006).

Given this brief introduction to defect causal analysis, the remaining subsections describe the defect causal analysis systematic literature review.

2.2. Planning the Systematic Review

The goal of the systematic review was to conduct an unbiased and fair review regarding the state of the art of defect causal analysis, more specifically aiming at:

- Summarizing the processes, approaches, and guidance that have been proposed for defect causal analysis.

- Additionally, summarize and analyze the defect and cause classification schemes used. Note that the goal here isn't to analyze all the existing classification schemes for defects and causes, but to focus on those from sources related to defect causal analysis.

Two research questions, Q0 and Q1, where formulated to address the goals listed previously. The question Q0 relates to a broader scope, analyzing causes of defects. The question Q1, on the other hand, is specifically related to defect causal analysis (involving the prevention in future projects). This strategy was adopted so that generic knowledge regarding the analysis of causes of defects could also be identified, since this knowledge can be used as a starting point towards defect causal analysis (including prevention). The description of the research questions, in the format suggested in [Biolchini et al., 2005], follows:

- Q0: Which processes, approaches and knowledge have been proposed and/or used for analyzing causes of software defects?
    - (P) Population: scientific publications and experience reports from software development projects, environments or organizations.
    - (I) Intervention: processes, approaches and knowledge for analyzing causes of software defects.
    - (C) Comparison: does not exist.
    - (O) Output: identification of processes, approaches and knowledge for analyzing causes of software defects.
- Q1: Which processes, approaches and knowledge have been proposed and/or used for software defect causal analysis (or defect prevention)?
    - (P) Population: scientific publications and experience reports from software development projects, environments or organizations.
    - (I) Intervention: processes, approaches and knowledge for software defect causal analysis (or defect prevention).
    - (C) Comparison: does not exist.
    - (O) Output: identification of processes, approaches and knowledge for software defect causal analysis (or defect prevention).

Based on these research questions, search strings could be derived and adjusted so that they could be executed on different digital libraries. Below we present the search string S0, derived for question Q0 by listing the keywords identified for the (P) and (I) and (C) and (O) structure.

- S0: (P) Population **and** (I) Intervention **and** (O) Output

  ("software development" or "software project" or "software process" or "software engineering" or "software organization" or "software environment" or "experience factory" or "software factory") **and** (("causal analysis" or "cause analysis" or "defect analysis" or "error analysis" or "failure analysis" or "fault analysis") and (defect or error or failure or fault) and (causal or cause)) **and** (process or approach or method or methodology or technique or knowledge or tool or paradigm or strategy).

The chosen digital libraries were: ACM Digital Library, EI Compendex, IEEE, Inspec, and Web of Science. The search strings were tested against a set of seven control papers dealing with defect causal analysis, read before the systematic review. The strings were able to retrieve five of them and, of course, many other defect causal analysis papers from the digital libraries. The two control papers that were not retrieved both satisfied the strings search criteria, however, in our opinion they were not retrieved because one of them was really not indexed in the chosen digital libraries and the other one was probably indexed improperly. Thus, our strings seemed to fit their purpose.

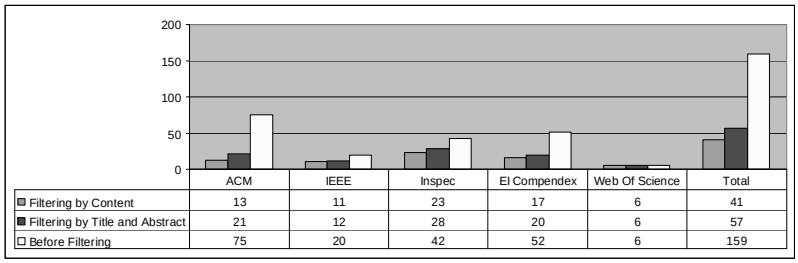
The criteria for including a retrieved paper in our review was that it must comprise processes, approaches or knowledge regarding analyzing causes of defects or defect causal analysis (defect prevention). For each of the selected papers the following

information was extracted: title, complete reference, source, defect classification scheme, cause classification scheme, identification of the process or approach, description of the process or approach, identification of knowledge, description of the knowledge, and type of study.

2.3. Execution of the Systematic Review

Once the review protocol was finished, the execution of the systematic review could be started. It was planned to occur on a yearly basis, thus avoiding becoming outdated over the years. A summary of the executions in August 2006 and August 2007 follows.

August 2006 Execution. The search strings were executed on the selected digital libraries, and 203 papers were retrieved. After eliminating replications (the same paper retrieved from more than one digital library) the total number of retrieved papers fell to 159. Those 159 papers were filtered further. The initial filtering was based on the title and abstract. Only those papers that were surely not related were excluded. As a result, the number of papers to read and evaluate decreased to 57. After reading the papers and trying to extract the information from them, some more papers were excluded, based on the criteria defined previously. Finally, 41 papers were selected as part of the first execution of our systematic review. The number of papers retrieved from each digital library during the filtering process is shown in Figure 3.

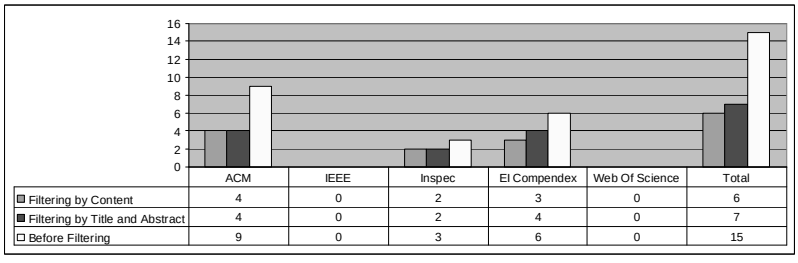


	ACM	IEEE	Inspec	EI Compendex	Web Of Science	Total
Filtering by Content	13	11	23	17	6	41
Filtering by Title and Abstract	21	12	28	20	6	57
Before Filtering	75	20	42	52	6	159

Figure 3. Number of papers retrieved in august 2006.

In addition to the 41 papers obtained by applying the review protocol to the digital libraries, information was extracted from the two control papers that were not retrieved and from one other paper: [Endres, 1975]. This paper was included because it was referenced by some of the selected papers as a seminal paper in the area. Again, we believe that the paper was indexed improperly in the IEEE digital library, since its content satisfies the strings search criteria. Therefore, as a result of applying the systematic review in August 2006, information was extracted from 44 research papers.

August 2007 Execution. In 2007 the same filtering process was performed. As a result 15 new papers were identified. After filtering by title and abstract the number of new papers decreased to 7. Finally, after filtering by content the number decreased to 6. The quantity of new papers from each digital library during the filtering process is shown in Figure 4.



	ACM	IEEE	Inspec	EI Compendex	Web Of Science	Total
Filtering by Content	4	0	2	3	0	6
Filtering by Title and Abstract	4	0	2	4	0	7
Before Filtering	9	0	3	6	0	15

Figure 4. Number of additional papers retrieved in august 2007.

All the analyzed papers and the way they cite each other, considering the August 2006 and August 2007 executions, are shown in Figure 5. This graph representation allowed

us to identify the most influential papers and research trends. The papers cited by more than three of the remaining papers are shown in light grey. The additional papers retrieved in the August 2007 execution are shown in grey.

The information extracted from each of the papers is registered in [Kalinowski and Travassos, 2008].

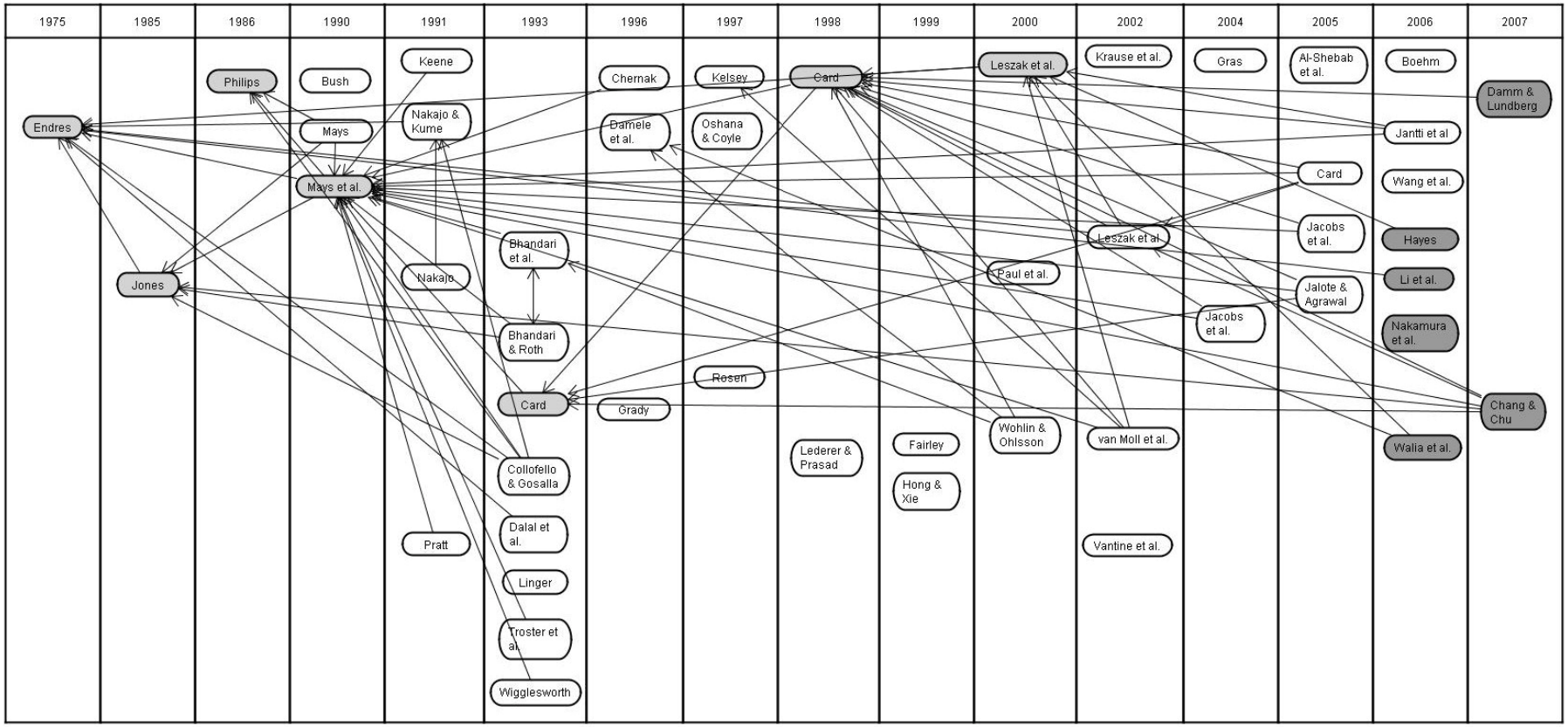


Figure 5. Citations between papers selected in the august 2006 and august 2007 reviews.

2.4. Results of the Systematic Review

The following information, regarding the goals of our systematic review, was extracted and grouped in tables, organized by publication date:

- processes and approaches used for defect causal analysis;
- defect classification schemes used by the papers;
- cause classification schemes used by the papers, and;
- knowledge regarding defect causal analysis generated or cited by those papers.

In the following subsections a summary of the analyses that performed based on those tables, aiming at obtaining the state of the art for each of the systematic review goals, is presented.

2.4.1. Processes and Approaches

The first approach found was the one described by Endres (1975), at IBM. This approach deals with individual analysis of software defects, in such a way that they can be categorized and their causes identified, allowing actions to be taken to prevent their occurrence in future projects or at least to assure their detection. The analysis of the defects occurs occasionally, as well as the corrective actions. This approach is still referenced in recent papers that involve retrospective defect analysis, such as [Li et al., 2006].

The difference between this approach and the traditional defect prevention process, also established at IBM [Jones, 1985] [Philips, 1986] [Mays, 1990] [Mays et al., 1990], about ten years later, is that the latter one is integrated into the development process. This integration occurs by establishing two additional tasks: (1) a causal analysis meeting after the rework activity is finished (which means that the defects at this point were already removed), and (2) a kickoff meeting, where the actions implemented since the last causal analysis meeting are communicated to the personnel involved in performing the activity.

The approach presented by Endres (1975) and the defect prevention process [Jones, 1985] analyzes defects exhaustively and involves in the analysis, if possible, the people responsible for introducing the defects. When appropriate, the meeting leader can skip some of the defects during the meeting without them being discussed.

Card (1993), describes a more flexible process involving the same concepts. The apparent flexibility comes from the fact that the causal analysis meeting isn't coupled to a specific development activity, but occurs periodically. Card (1993) also highlights, based on the experience at Computer Science Corporation referenced in the paper, that it is not necessary to analyze all defects exhaustively, a sample will suffice, and that the action proposals should treat classes of defects and not each defect separately.

Most of the papers analyzed follow processes congruent to the traditional defect prevention process [Jones, 1985] or the corresponding defect causal analysis process [Card, 1993]. Among the papers that explicitly follow or suggest this process are [Jones, 1985], [Philips, 1986], [Mays, 1990], [Mays et al., 1990], [Pratt, 1991], [Card, 1993], [Collofello and Gosalla, 1993], [Damele et al., 1996], [Grady, 1996], [Card, 1998], [Leszak et al., 2000], [Leszak et al., 2002], [van Moll et al., 2002], [Jalote and Agrawal, 2005], and [Card, 2005].

Grady (1996) describes defect causal analysis experiences at Hewlett Packard. The periodicity varied: one shot causal analysis (performed randomly), post project causal analysis (performed after the conclusion of the project), and a continuous process improvement cycle (performed after each development phase).

Besides the defect causal analysis approaches, some specific techniques to support causal analysis could be identified. Even though they are not defect causal analysis approaches, we considered them in our review. They represent techniques to identify cause-effect relations: [Nakajo, 1991], [Nakajo and Kume, 1991], [Wohlin et al., 2000], [Krause et al., 2002], and [Gras, 2004]; statistical control techniques: [Hong et al., 1999], and [Wang et al., 2006]; and defect analysis techniques: [Bhandari et al., 1993], [Bhandari and Roth, 1993], [Kelsey, 1997], [Nakamura et al., 2006], and [Damm and Lundberg, 2007]. A summary of those techniques is described in [Kalinowski and Travassos, 2008].

Based on the literature review, two techniques that seem to show themselves particularly useful to support defect causal analysis activities are Pareto charts and cause-effect (or Ishikawa) diagrams [Ishikawa, 1976]. They support, respectively, identifying the most common defect types and finding the causes for specific defect classes. Although cause-effect diagrams were cited by [Mays et al., 1990] as used in the manufacturing area, the first of the analyzed papers to use those diagrams in the context of software defect causal analysis appeared in 1996. The cause-effect diagram is used and/or suggested in many of the analyzed papers, including [Chernak, 1996], [Damele et al., 1996], [Grady, 1996], [Card, 1998], [van Moll et al., 2002], [Jacobs et al., 2004], [Card, 2005], [Jacobs et al., 2005] [Jalote and Agrawal, 2005], and [Wang et al., 2006]. Among the papers that cite pareto charts as a useful technique are [Card, 1998], [Hong et al., 1999], [van Moll et al., 2002], [Card, 2005] [Jalote and Agrawal, 2005], and [Wang et al., 2006].

2.4.2. Defect Classification Schemes

Two types of information about classification were extracted from the papers: defect information to be collected and defect types.

Defect information to be collected. We noted a certain consensus among the authors on the relevant information to be collected about software defects in order to support defect causal analysis. Among the data considered relevant since the initial papers [Endres, 1975] [Jones, 1985] [Mays, 1990] and continuing to the more recent ones [Agrawal and Jalote, 2005] [Jantti et al., 2006] [Chang and Chu, 2007] [Damm e Lundberg, 2007] are:

- The moment (or phase) in which the defect was introduced.
- The moment (or phase) in which the defect was detected.
- Defect type.

This information matches the three dimensions to classify defects suggested in [Card, 1998] and [Card, 2005]. None of the analyzed papers argued against the use of any of these dimensions to classify defects.

However, other information is considered in some papers, according to the specific goals of the defect causal analysis approaches presented in them:

- Correction (effort and time) or severity (impact), is considered in many papers, for example [Endres, 1975] [Collofello e Gosalla, 1993] [Fairley, 1999] [Leszak et al., 2000] [Leszak et al., 2002] [Jantti et al., 2006] [Nakamura et al., 2006] and [Chang and Chu, 2007]. Moreover, severity is also considered in the papers that use ODC (Othogonal Defect Classification): [Bhandari et al., 1993] [Chernak, 1996].
- Location (or module), is considered by many papers, such as [Endres, 1975], [Bhandari et al., 1993], [Bhandari e Roth, 1993], [Kelsey, 1997], [Leszak et al., 2000], [Leszak et al., 2002], [van Moll et al., 2002], [Nakamura et al., 2006], and [Chang and Chu, 2007].

Aditional information found in two papers that use the ODC [Chillarege et al., 1992] classification scheme [Bhandari et al., 1993] [Chernak, 1996] includes:

- Trigger, which indicates how the defect was detected.
- If the defect was introduced while developing new functionality or while maintaining existing functionality.

Defect types. Regarding the defect types, several different taxonomies could be found. The complete list of taxonomies in chronological order is reported in [Kalinowski and Travassos, 2008]. An excerpt of that list, with three most cited of the nine taxonomies found in the papers, follows:

- ODC taxonomy [Chillarege et al., 1992], with the following types: interface, function, build/package/merge, assignment, documentation, checking, algorithm, timing/serialization. Of the analyzed papers this taxonomy was used by [Bhandari et al., 1993] and [Chernak, 1996] it was also mentioned in some other papers, like [Card, 1998] [Card, 2005].
- Taxonomy for defects used at Hewlett-Packard [Grady, 1996], containing several types of defects organized by development phase. For each type its nature or mode (missing, unclear, wrong, changed, or better way) are also determined.
- Taxonomy for requirements defects used at NASA, containing 13 types of requirements defects, used in [Hayes et al., 2006]. It represents a tailoring of the taxonomy described by Shull (1998) (which includes the defect types: ambiguity, omission, inconsistent information, incorrect fact, and extraneous information), reflecting the reality of NASA. In Leszak et al (2002) similar types of defects are referred to as the nature of the defect, and aggregated to more refined defect types.

The great variety of taxonomies found reflects the argument of Card (2005), that the defect taxonomy should be created in such a way that it supports the specific analysis interests of the organization that is implementing defect causal analysis. This argument is reinforced by Jalote and Agrawal (2005), for instance, who used the same argument to justify the elaboration of a specific taxonomy for InfoSys.

2.4.3. Cause Classification Schemes

Analyzing the different cause classification schemes listed in [Kalinowski and Travassos, 2008] it is possible to identify a consensus among the authors about the categories that should be considered for defect causes. Moreover, most of the categories contained in the different classification schemes can be easily mapped to the categories initially proposed by Ishikawa (1976) for manufacturing. The Ishikawa categories are suggested as a starting point for defect causal analysis approaches in [Card, 1998] and [Card, 2005]. They are the following: tools, input, people, and methods.

However, it is important to highlight that the remaining schemes, considering the specific context of software, can bring contribute additional information by refining the Ishikawa scheme. For instance, an additional category considered in many schemes developed for analyzing software defects is “lack of understanding of the requirements”. This category can be considered a subcategory of “people”[3], but since in several cases [Endres, 1975] [Nakajo and Kume, 1991] [Hayes, 2006] many causes matched that category it could be helpful to consider it separately. Further suggestions

[3] It could also be mapped to the “input” category, depending on the nature of the misunderstanding.

on subcategories identified in other schemes can be found in [Kalinowski and Travassos, 2008].

An interesting classification scheme is that cited in [Jalote and Agrawal, 2005], the five "Ms" and one "E" scheme [Robitaille, 2004]. The five "Ms" refer to the categories "Material", "Method", "Manpower", and "Measurement", and "Machinery". The "E" refers to the "Environment". According to Robitalle (2004) this scheme is commonly used in manufacturing. However, Jalote and Agrawal (2005) decided to adapt it to the context of software, and suggest the following categories: process, people, and technology. They mention that those are the factors with the highest impact on quality and productivity [Jalote, 2000] and [SEI, 1995]. Those categories again can be mapped to the Ishikawa categories, removing just the "input" category.

Finally, the papers [Jacobs et al., 2004] and [Jacobs et al., 2005] present a category that is not obviously included in the Ishikawa categories, the "organization". This category is related to organizational causes and not specifically to its processes. For instance, inadequate or unavailable organizational policies, improper geographical distribution of departments (for instance, communication problems do to different time zones), improper organizational structure (for instance, no employee responsible for configuration management), among others. It is important to mention that those papers focus on projects with globally distributed teams, and that this category seems to be especially useful in that context.

2.4.4. Knowledge Regarding Defect Causal Analysis

In order to facilitate the application of the knowledge gathered from analyzing the papers, it was structured to separate generic knowledge about the entire process from the knowledge limited to specific activities and tasks. The following tables were generated:

- A table with generic knowledge regarding the entire process.
- One table for each of the defect causal analysis activities of the defect prevention process described in [Jones, 1985], [Mays, 1990], and [Mays et al., 1990]. As described before, those four activities are: (1) causal analysis meeting, (2) implementation of the actions, (3) stage kickoff, and (4) data collection and monitoring. Inside the causal analysis meeting group of knowledge subgroups were created for each of the six steps mentioned in [Card, 2005]: (1) selecting the sample, (2) categorizing the defects, (3) finding the systematic error, (4) finding the main causes, (5) elaborating action proposals, and (6) documenting meeting results.

The complete tables of knowledge, organized by publication date, are available in the report that supports this paper [Kalinowski and Travassos, 2008]. It is important to mention that the knowledge described in the tables should be used considering the context in which it was generated, as described in the appendix of the report. The appendix lists the information extracted from each of the papers, together with a description of the knowledge and the type of study the paper presents.

3. Guidance to Efficiently Implement Defect Causal Analysis

Based on the knowledge acquired and the analyses performed as a result of the systematic review some guidance to implement defect causal analysis could be elaborated. The guidance aims at providing reasonable answers to the questions listed in the introduction. These are questions commonly faced by professionals when trying to implement defect causal analysis in software organizations. Is my organization ready for defect causal analysis? What approach should be followed? How should defects be

categorized? How should causes be categorized? What are the expected costs and results of implementing defect causal analysis?

The guidance is composed of unbiased evidence-based answers to those questions and is complemented by the tables of knowledge regarding the process and each of its activities, listed in [Kalinowski and Travassos, 2008].

Is my organization ready for defect causal analysis?

The main requirement for applying defect causal analysis is collecting data regarding defects. However, as argued by Card (1993) and van Moll et al (2002), defect causal analysis benefits from having a defined process established for the projects.

Given the potential benefits and return of investment [Mays et al., 1990] [Card, 1993] [Leszak et al., 2000] [Jalote and Agrawal, 2005], we recommend the implementation of defect causal analysis approaches even for lower maturity organizations, even though it is only required at the highest maturity levels of models such as MPS and CMMI. Defect causal analysis can help to improve the performance and capability of processes before those high maturity levels are achieved [Card, 1998] [Card, 2005].

What approach should be followed?

Based on the scenario described in the previous section we believe that an efficient defect causal analysis approach could follow the traditional defect prevention approach [Jones, 1985], which has been used in early and recent approaches. As mentioned before, this process has shown itself of low cost and efficient in reducing defect rates in different organizational contexts. As suggested by Card (1993) the causal analysis meeting should focus on classes of defects, sampled before the causal analysis meeting. Among the specific techniques we suggest Pareto charts to identify the main defect classes to be analyzed and cause-effect diagrams to support identifying the main causes for the systematic errors that lead to those defect classes.

Among the metrics to be collected to support defect causal analysis and understanding its results, based on the systematic review, we suggest the number of defects found per unit of size, the mean number of defects found per inspector per hour (if available), and the PIQ (Phase Input Quality) and POQ (Phase Output Quality) metrics defined by Damm and Lundberg (2007). The number of defects indicates efficiency in preventing defects, the PIQ and POQ metrics, on the other hand, provide further insight into the defect detection mechanisms efficiency. Those metrics could be placed under statistical control integrating defect causal analysis with statistical process control, providing a more precise and quantitative way of determining how the process performance and capability are affected by defect causal analysis.

Defect causal analysis helps accomplish two goals of statistical process control: (1) controlling and stabilizing processes and (2) improving their performance and capability. As mentioned in [Florac and Carleton, 1999] the first of these goals deals with assignable (or special) causes. The existence of assignable causes can be detected by applying process stability tests, such as the ones described by Wheeler et al., (1992) to statistical process control charts. If the process is not stable, then defect causal analysis can help to find the assignable causes and implement actions that stabilize it. The second goal addresses the common causes for the process' current performance and capability [Florac and Carleton, 1999]. This can be accomplished by applying defect causal analysis even if it the process is stable, for instance, after each development phase. Relating this to the capability maturity models MPS and CMMI, controlling and stabilizing processes is related to MPS maturity level B and CMMI maturity level 4.

Improving the performance and capability on the hand is related to MPS maturity level A and CMMI maturity level 5.

Our systematic review indicates that the most appropriate control charts for controlling defect data and supporting defect causal analysis are the U charts [Hong et al., 1999][Card, 2005]. U charts apply to data that has a Poisson distribution, with a varying area of opportunity (for instance, the size of a document) for an event to occur (for instance, detecting a defect) [Montgomery, 1991 *apud* Hong et al., 1999]. Individuals or XmR charts may be used, but are less sensitive.

How should defects be categorized?

Defect information to collect. Based on the information collected from the retrieved approaches and papers, we believe that an efficient defect causal analysis approach should consider at least the consensus information, which is: moment of introduction, moment of detection, and type of defect. Additionally, in our point of view, knowing the severity of the defects (or their impact) and their location could support defect causal analysis by identifying clusters of defects to support the selection of the samples to be analyzed and monitoring the efficiency of the defect causal analysis process itself. However, severity is largely an accident and the same mistake made in different contexts can have very different impacts. Finally, additional information, such as the trigger and if the defect is related to new functionality, can be used for specific situations, such as trying to improve detection mechanisms (when prevention is difficult to achieve), as done by [Chernak, 1996].

Moreover, once the cause of the defect is determined, a link between the defect and its cause should be established and the cause should be stored according to its own classification scheme.

Defect types. The results of the systematic review suggest that an efficient defect causal analysis approach should use a taxonomy that considers the nature and the type of defect. The nature can be expressed using the categories described by Shull (1998), which can be tailored to specific contexts, as in [Hayes, 2006]. These categories (omission, ambiguity, inconsistent information, incorrect fact, extraneous information, and other) can be used in a generic way for the different artifact types generated throughout he software development lifecycle, as observed by Travassos et al (2001).

For the defect type, a taxonomy should be created considering the specific goals of the organizations' defect causal analysis approach, as suggested by Card (2005). Not having a defect type taxonomy in place at the organization the types considered by ODC [Chillarege et al., 1992], commonly used [Card, 2005], could serve as a starting point for further tailoring.

How should causes be categorized?

Given the consensus scenario regarding defect cause categories, we believe that the Ishikawa categories represent a good starting point [Ishikawa, 1976], as suggested by Card [Card, 1998] [Card, 2005], adding for the sake of completeness the category "organization" [Jacobs et al., 2004] [Jacobs et al., 2005]. The tailoring of the classification scheme can be done afterwards, by creating specific subcategories, based on the reality of the organization itself (taken for instance from the results of prior defect causal analyses).

What are the expected costs and results of implementing defect causal analysis?

The answer to this question helps to establish quantitative improvement goals for implementations of defect causal analysis. Based on the generic knowledge gathered regarding defect causal analysis approaches, its cost varies from 0.5 to 1.5 percent of the projects budget (including the implementations of the proposed actions), according

to data taken from IBM and Computer Science Corporation [Card, 2005]. In our point of view this is a very low cost, especially when considering the benefits, which have been reducing defect rates by over 50 percent (again considering data from IBM and Computer Science Corporation) [Card, 2005]. As a consequence rework effort and costs will be reduced while productivity is increased.

Even though they measured the benefits in different ways (for instance, reduction of rework effort, reduction of specific defect types, among others), all of the analyzed papers showed positive results from implementing defect causal analysis. The papers describe experiences from a great range of software organizations, including: AG Communication Systems Corporation, Computer Science Corporation, Hewlett-Packard, IBM (several organizational units), InfoSys, Italtel SIT BUCT Línea UT, Lucent Technology, Motorola, and Philips Software Centre.

4. Conclusions

Defect causal analysis has shown itself an inexpensive and high return means of product focused software process improvement [Kalinowski, 2007]. However, as mentioned by Card [2005], despite its advantages and wide industry adoption little academic research is being done in this area. Thus little knowledge has been generated and published. As a consequence professionals face many questions when trying to implement defect causal analysis in software organizations.

A systematic review was conducted in order to provide unbiased and evidence-based answers to some of those questions [Kalinowski and Travassos, 2008]. Based on the results of the systematic review we were able to identify the defect causal analysis state of the art in an unbiased way and to provide some guidance on how to efficiently implement it in software organizations.

Moreover, some opportunities for further investigation were identified. For instance, we found no approach allowing dynamic updates to the cause-effect relationships that reflect the reality of the organizational context, based on the results of the causal analysis itself. According to the results of the systematic review and to our knowledge this is first being addressed in the proposal described in [Kalinowski et al., 2008].

We believe that making the resulting guidance and knowledge available in the academic and software practitioners' community could facilitate the implementation of defect causal analysis, complementing other valuable sources of information, such as [Card, 2005] and [SOFTEX, 2007b]. As the main contributions of this guidance we highlight the unbiased compilation of answers to commonly faced questions and lists of knowledge regarding the defect causal analysis process and its activities.

References

Al-Shehab, A. J., Hughes, R. T., Winstanley, G., Facilitating Organisational Learning through Causal Mapping Techniques in IS/IT Project Risk Management, LNCS, 3782 NAI, 145 - 154, 2005.

Bhandari, I., Halliday, M., Tarver, E., Brown, D., Chaar, J., Chillarege, R., A Case Study of Software Process Improvement During Development, IEEE Trans. on Soft. Eng., 19 (12), 1157 – 1170, 1993.

Bhandari, I., Roth, N., Post-process Feedback with and without Attribute Focusing: a Comparative Evaluation, in 'ICSE '93: Proceedings of the 15th Intl' Conf. on Soft. Eng.', IEEE Computer Society Press, Los Alamitos, CA, USA, pp. 89-98, 1993.

Biolchini, J.; Mian, P.G.; Natali, A.C.; & Travassos, G.H. (2005), "Systematic Review in Software Engineering: Relevance and Utility", Technical Report ES-679/05, PESC-COPPE/UFRJ. Available at http://www.cos.ufrj.br .

Boehm, B., A view of 20th and 21st century software engineering, in 'ICSE '06: Proceeding of the 28th international conf. on Soft. Eng.', ACM Press, New York, NY, USA, pp. 12-29, 2006.

Brereton, P., Kitchenham, B. A., Budgen, D., Turner, M., Khalil, M., Lessons from applying the systematic literature review process within the software engineering domain. Journal of Systems and Software. Volume 80, Issue 4. Pages 571-583, 2007.

Bush, M., Getting started on metrics - JPL productivity and quality, in 'ICSE '90: Proceedings of the 12th international conf. on soft. eng.', IEEE Comp. Soc. Press, Los Alamitos, CA, USA, pp. 133-142, 1990.

Card, D., “Defect Analysis: Basic Techniques for Management and Learning”, Advances in Computers, vol. 65, chapter 7, pp. 259-295, 2005.
Card, D., Learning from our mistakes with defect causal analysis, IEEE Software, 15(1), pp. 56-63, 1998.
Card, D., Defect Causal Analysis Drives Down Error Rates, IEEE Software 1/1993, Volume 10, Issue 4, 7/1993, p.98-99, 1993.
Chang, C., Chu, C., "Defect Prevention in Software Processes: An Action-Based Approach", The Journal of Systems and Software, Vol. 80, issue 4, April 2007, pp. 559-570, 2007.
Chernak, Y., A statistical Approach to the Inspection Checklist Formal Synthesis and Improvement, IEEE Transactions on Software Engineering, 22(12), 866-874, 1996.
Chillarege, R., Bhandari, I., Chaar, J., Halliday, M., Moebus, D., Ray, B., Wong, M.Y., Ortogonal Defect Classification – A Concept for In-Process Measurement, IEEE Transactions on Software Engineering, vol. 18, pp. 943-956, 1992.
Collofello, J., Gosalla, B., Application of Causal Analysis to the Software Modification Process, Software Practice and Experience, 23(10), pp. 1095–1105, 1993.
Dalal, S. R., Horgan, J. R., Kettenring, J. R., Reliable Software and Communication: Software Quality, Reliability, and Safety, in 'ICSE '93: Proceedings of the 15th International Conference on Software Engineering', IEEE Computer Society Press, Los Alamitos, CA, USA, pp. 425-435, 1993.
Damele, G., Bazzana, G., Andreis, F., Aquilio, S., Process Improvement through Root Cause Analysis, in ‘Proceedings of Third International Conference on Achieving Quality in Software’, pp. 35--47, 1996.
Damm, L., Lundberg, L., Company-wide Implementation of Metrics for Early Software Fault Detection, Proc. of the 29th International Conference on Software Engineering (ICSE’07), Minneapolis, 2007.
Dangerfield, O., Ambardekar, P., Paluzzi, P., Card, D., Giblin, D., Defect Causal Analysis: A Report from the Field, Proceedings of International Conference of Software Quality, American Society for Quality Control, 1992.
Eckes, G., The Six Sigma Revolution: How General Electric and Others Turned Process Into Profits, John Wiley and Sons, 2000.
Endres, A., “An Analysis of Errors and Their Causes in Systems Programs", IEEE Transactions on Software Engineering, SE-1, 2, June 1975, pp. 140-149, 1975.
Fairley, R. E., Managing by the Numbers: a tutorial on quantitative measurement and control of software projects, in 'Proceedings of the 21st International Conference on Software Engineering, ICSE 99', IEEE Computer Society Press, Los Alamitos, CA, USA, pp. 677- 678, 1999.
Florac, A.W., Carleton A.D., Measuring the Software Process: Statistical Process Control for Software Process Improvement, Pearson Education, 1999.
Grady, R. B., Software Failure Analysis for High-Return Process Improvement Decisions, Hewlett-Packard Journal, 47 (4), 15 - 24, 1996.
Gras, J.J., End-to-End Defect Modeling, IEEE Software, 21(5), 98-100, 2004.
Hayes, J.H., Raphael, I., Holbrook, E.A., Pruett, D.M., A case history of International Space Station requirement faults, Proceedings of the11th IEEE International Conference on Engineering of Complex Computer Systems, Stanford University, California, 2006.
Hong, G., Xie, M., Shanmugan, P., A Statistical Method for Controlling Software Defect Detection Process, Computers and Industrial Engineering 37 (1-2), pp. 137-140, 1999.
IEEE, 1990, IEEE Standard Glossary of Software Engineering Terminology, Standard 610, IEEE Press.
ISO/IEC, ISO/IEC 12207: Information technology - Software life-cycle processes – Amendment 2, Geneve: ISO, Int. Org. for Standardization and the Int. Electrotechnical Commission, 2004.
Jacobs, J.C., van Moll, J.H., Krause, P.J., Kusters, R.J., Trienekens, J.J.M., Brombacher, A., Exploring Defect Causes in Products Developed by Virtual Teams, Journal on Information and Software Technology, 47(6), 399 – 410, 2005.
Jacobs, J.C., van Moll, J.H., Krause, P.J., Kusters, R.J., Trienekens, J.J.M., Effects of Virtual Development on Product Quality: Exploring Defect Causes, Proceedings of the 11th Annual International Workshop on Software Technology and Engineering Practice (STEP’04), 2004.
Jalote, P., Agrawal, N., “Using Defect Analysis Feedback for Improving Quality and Productivity in Iterative Software Development”, (Invited paper, 3rd International Conference on Information and Communication Technology, ICICT, 2005.), pp. 701 – 713, Cairo, 2005.
Jantti, M., Toroi, T., Eerola, A., Difficulties in establishing a defect management process: A case study, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 4034 NCS, pp. 142-150, 2006.
Jones, C.L., A process-integrated approach to defect prevention, IBM Systems Journal, 24(2), 150-67, 1985.
Kalinowski, M., Travassos, G.H., Uma revisão sistemática a respeito de análise causal de defeitos de software, technical report available at http://www.cos.ufrj.br , COPPE/UFRJ, 2008.
Kalinowski, M., Travassos, G.H., Card, D.N., Towards a Defect Prevention Based Process Improvement Approach, 34th EUROMICRO Conference, IEEE Computer Society, 2008.
Kalinowski, M., Defect Causal Analysis: An Opportunity for Product Focused Software Process Improvement, Invited paper at the V CICIS (Congreso Internacional de Computación y Ingenieria de Sistemas), Moquegua, Peru, 2007.
Keene, J., Managing Software Reliability and Support Costs, in 'Proceedings of the Leesburg Workshop on Reliability and Maintainability Computer-Aided Engineering in Concurrent Engineering', pp. 201-205, 1991.

Kelsey, R.B., Integrating a Defect Typology with Containment Metrics, ACM SIGSOFT Software Engineering Notes, 22(2), 64-67, 1997.
Kitchenham, B.A., Procedures for Performing Systematic Reviews, Joint Technical Report Keele University and National ICT Australia Ltd., 2004.
Krause, P., Freimut, B., Suryn, W., New Directions in Measurement for Software Quality Control, in 'Proc. of the 10th Int. Workshop on Soft. Tech. and Eng. Practice, STEP 2002', pp. 129-143, 2002.
Lederer, A.L., Prasad, J., A Causal Model for Software Cost Estimating Error, IEEE Transactions on Software Engineering, 24 (2), pp. 137-148, 1998.
Leszak, M., Perry, D., Stoll, D., A Case Study in Root Cause Defect Analysis, in 'International Conference on Software Engineering, ICSE 2000', pp. 428-437, 2000.
Leszak, M., Perry, D. E., Stoll, D., Classification and evaluation of defects in a project retrospective, Journal of Systems and Software, 61(3), 173 - 187, 2002.
Li, Z., Tan, L., Wang, X., Lu, S., Zhou, Y., Zhai, C., Have things changed now? An empirical study of bug characteristics in modern open source software, Proceedings of ASID'06: 1st Workshop on Architectural and System Support for Improving Software Dependability, pp. 25-33, 2006.
Linger, R. C., Cleanroom Software Engineering for Zero-Defect Software, in 'ICSE '93: Proceedings of the 15th International Conference on Software Engineering', IEEE Computer Society Press, Los Alamitos, CA, USA, pp. 2-13, 1993.
Mays, R.G., Applications of Defect Prevention in Software Development, IEEE Journal on Selected Areas in Communications, Vol.8, No.2, pp. 164-168, February 1990.
Mays, R.G., Jones, C.L., Holloway, G.J., Studinski, D.P., Experiences with Defect Prevention, IBM Systems Journal, 29(1), pp. 4-32, 1990.
van Moll, J., Jacobs, J., Freimut, B., Trienekens, J., The Importance of Life Cycle Modeling to Defect Detection and Prevention, in '10th International Workshop on Software Technology and Engineering Practice, 2002. STEP 2002', pp. 144-155, 2002.
Nakajo, T. & Kume, H. (1991), A Case History Analysis of Software Error Cause-Effect Relationships, IEEE Transactions on Software Engineering, 17(8), 830-838, 1991.
Nakajo, T., Foolproofing and Quality Feedback: Keys of Process-Based Management, Proceedings of the Fifteenth Annual International Computer Software and Applications Conference, COMPSAC'91, IEEE Computer Society Press, 390-391, 1991.
Nakamura, T., Hochstein, L., Basili, V.R., Identifying Domain-Specific Defect Classes Using Inspections and Change History, Proceedings of the 2006 International Symposium on Empirical Software Engineering (ISESE), Sept. 21-22, Rio de Janeiro, Brazil. p. 346-355, 2006.
Oshana, R., Coyle, F.P., Implementing Cleanroom Software Engineering into a Mature CMM-Based Software Organization, Proceedings of the 19th International Conference on Software Engineering, ICSE'97, ACM Press, New York, NY, USA, pp. 572-573, 1997.
Paul, R. A., Bastani, F.; Yen, I., Challagulla, V. U., Defect-based Reliability Analysis for Mission-Critical Software, Proceedings of the IEEE Computer Society's International Computer Software and Applications Conference, pp. 439 - 444, 2000.
Philips, R.T., An approach to software causal analysis and defect extinction, in 'Proceedings of the IEEE Globecom Conference, pp. 412-416, 1986.
Poppendieck, M., Poppendieck, T., Lean Software Development: An Agile Toolkit, Addison Wesley Professional, 2003.
Pratt, W. M., Experiences in the Application of Customer-Based Metrics in Improving Software Service Quality, Conf. Record of the International Conference on Communications, pp. 1459 - 1462, 1991.
Robitaille, D., Root Cause Analysis – Basic Tools and Techniques, Paton Press, 2004.
Rombach, D., Endres, A., A Handbook of Software and Systems Engineering – Empirical Observations, Laws and Theories, Pearson Addison Wesley, 2003.
Rosen, C., PLUNGE DA: A Case Study, ACM SIGSOFT Soft. Eng. Notes, 22(2), pp. 82-83, 1997.
SEI, CMMI for Development (CMMI-DEV), Version 1.2, Technical report CMU/SEI-2006-TR-008. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2006.
Shull, F., Developing Techniques for Using Software Documents: A Series of Empirical Studies, Ph.D. thesis, University of Maryland, College Park, 1998.
SOFTEX, MPS.BR – Melhoria de Processo do Software Brasileiro – Guia Geral, version 1.2, June 2007, available at http:www.softex.br/mpsbr , 2007a.
SOFTEX (Rocha, A.R.C., Montoni, M.A., Souza, G.S., Kalinowski, M., Scalet, D.), MPS.BR – Melhoria de Processo do Software Brasileiro – Guia de Implementação do Nível A, version 1.0, June 2007, available at http:www.softex.br/mpsbr , 2007b.
Travassos, G.H., Shull, F., Carver, J., "Working with UML: A Software Design Process Based on Inspections for the Unified Modeling Language", in: Advances in Computers, vol. 54, Academic Press, 2001.
Troster, J., Henshaw, J., Buss, E., Filtering for Quality, in 'Proceedings of the 1993 Conf. of the Centre for Advanced Studies on Collaborative Research, CASCON 93', IBM Press, pp. 429-449, 1993.
Vantine, W., Benfield, K., Pritts, D., Ballard, K., Evaluating and Incorporating New Age Software Technology for Identifying Systemic Root Causes, in: 'Proceedings of the Joint ESA-NASA Space Flight Safety Conference, ESTEC', ESA SP-486, Noordwijk (NL), pp. 369 - 376, 2002.
Walia, G., Carver, J., Philip, T., Requirements Error Abstraction and Classification: An Empirical Study, Proc. Int. Symposium on Empirical Soft. Eng. (ISESE), Sept. 21-22, Rio de Janeiro, Brazil, 2006.

Wang, Q., Jiang, N., Gou, L., Liu, X., Li, M., Wang, Y., BSR: A Statistic-based Approach for Establishing and Refining Software Process Performance Baseline, in 'ICSE '06: Proceeding of the 28th Int. Conf. on Software Engineering', ACM Press, New York, NY, USA, pp. 585-594., 2006.

Wheeler, D.J., Chambers, D.S., Understanding Statistical Process Control, 2ed, Knoxville, Tenn., SPC Press, 1992.

Wigglesworth, J., Surveys as a Method for Improving the Development Process, in 'CASCON '93: Proc. conf. of the Centre for Advanced Studies on Collaborative research', IBM Press, pp. 337-355, 1993.

Wohlin, C., Host, M., Ohlsson, M., Understanding the Sources of Software Defects: A Filtering Approach, in '8th Int. Workshop on Program Comprehension, 2000, IWPC 2000', pp. 9-17, 2000.