Publication VI

Mika V. Mäntylä, Juha Itkonen, and Joonas Iivonen. 2012. Who tested my software? Testing as an organizationally cross-cutting activity. Software Quality Journal, volume 20, number 1, pages 145-172.

Software Qual J (2012) 20:145–172
DOI 10.1007/s11219-011-9157-4

Who tested my software? Testing as an organizationally cross-cutting activity

Mika V. Mäntylä · Juha Itkonen · Joonas Iivonen

Published online: 21 August 2011

Abstract There is a recognized disconnect between testing research and industry practice, and more studies are needed on understanding how testing is conducted in real-world circumstances instead of demonstrating the superiority of specific methods. Recent literature indicates that testing is a cross-cutting activity that involves various organizational roles rather than the sole involvement of specialized testers. This research empirically investigates how testing involves employees in varying organizational roles in software product companies. We studied the organization and values of testing using an exploratory case study methodology through interviews, defect database analysis, workshops, analyses of documentation, and informal communications at three software product companies. We analyzed which employee groups test software in the case companies, and how many defects they find. Two companies organized testing as a team effort, and one company had a specialized testing group because of its different development model. We found evidence that testing was not an action conducted only by testing specialists. Testing by individuals with customer contact and domain expertise was an important validation method. We discovered that defects found by developers had the highest fix rates while those revealed by specialized testers had the lowest. The defect importance was susceptible to organizational competition of resources (i.e., overvaluing defects of reporter's own products or projects). We conclude that it is important to understand the diversity of individuals participating in software testing and the relevance of validation from the end users' viewpoint. Future research is required to evaluate testing approaches for diverse organizational roles. Finally, to improve defect information, we suggest increasing automation in defect data collection.

Keywords Industrial case study · Testing · Roles · Testers · Defect reporters · Values · Defect fix rate · Defect data analysis · Interviews

M. V. Mäntylä (✉) · J. Itkonen · J. Iivonen
Department of Computer Science and Engineering, School of Science, Aalto University,
P.O. Box 19210, 00076 Aalto, Finland
e-mail: mika.mantyla@aalto.fi

J. Itkonen
e-mail: juha.itkonen@aalto.fi

J. Iivonen
e-mail: joonas.iivonen@iki.fi

1 Introduction

Software testing is a widely researched quality assurance topic. However, a prominent gap exists between academic research and the problems that are encountered by practitioners—an issue acknowledged on both sides (Glass et al. 2006). For example, the scientific community has a vision of 100% automatic testing (Bertolino 2007), whereas credible practitioners' reports claim that automated testing can never completely replace manual testing (Berner et al. 2005; Martin et al. 2007). Furthermore, some reports have claimed that the problems associated with software testing in the industry are not primarily technical; rather, they originate from other sources, such as the socio-technical environment and organizational structure of the company (Martin et al. 2007; Ahonen et al. 2004).

System-level testing is traditionally seen as a separate action that is executed by testing specialists or even by a separate quality assurance organization. Myers (1979) states that programmers should avoid testing their own programs and that programming organizations should not test their own programs. However, there are several hints in the literature that, in many cases, testing is actually a cross-cutting activity that involves knowledge and people from different organizational roles and functions. First, at Microsoft, people from various roles participate in internal system-level testing in a practice known as "eating your own dog food" (Cusumano and Selby 1995). Second, extreme programming (Beck 2000) recognizes the onsite customer as the key player in system-level testing. Third, empirical studies of testing in the industry highlight the importance of domain knowledge (Beer and Ramler 2008; Itkonen and Rautiainen 2005; Iivonen et al. 2010; Kettunen et al. 2010), which suggests that there could be benefits in using domain experts, such as sales people, in system-level testing.

Based on this idea of software testing as a cross-cutting activity, the research objective of this study is to provide empirical evidence on how testing involves different groups of employees in varying organizational roles in software product development companies. Another objective is to add to the shallow body of empirical reports that describe how testing is conducted in practice and in realistic environments (Martin et al. 2007; Beer and Ramler 2008; Rooksby et al. 2009; Juristo et al. 2009). First, we wish to understand the organization of testing in companies, and we aim to describe not only the organization and process but also the underlying values that affect the testing work in the companies. It is believed that the values can explain the organization of software testing. The values upon which the testing builds within the organizations can be related, for example, to technical excellence, knowledge, and experience, or priorities and desired quality characteristics. Second, we assess what types of employees in different organizational roles are involved in software testing, because we have noticed that the conventional approach in which testing is a separate organizational unit does not seem to hold with the medium-sized and small software product companies with whom we have mainly collaborated. Our goal is to understand the various employee groups actually carrying out testing activities and any differences in their results.

Our work is an industrial case study of employees performing software testing and reporting defects. The study was performed in three successful medium-sized software product companies. In all cases, the software products are used by domain experts in the field of engineering. From each company, we collected both qualitative and quantitative data from several sources: defect database, interviews, quality goal workshop results (Vanhanen et al. 2009), defect reporting guidelines, organizational charts, and informal discussions with company personnel. This paper is structured as follows: In the next section, we summarize the relevant related work for our study. We describe our research

methodology in Sect. 3, and the results are presented in the Sect. 4. Section 5 includes the discussion and answers to the research questions, followed by conclusions and directions for the future work in Sect. 6.

2 Related work

Although there is no shortage of software testing research, limited research has been done on industrial case studies of software testing. In this section, we summarize the most important studies related to our work according to the following research topics: organization of testing, values related to software testing, the employee groups and roles of those performing software testing within the organization, and the effects of employee groups or roles on the defects that are detected.

2.1 Organization of testing

Martin et al. (2007) argue that software testing is a socio-technical, rather than purely technical, process that is greatly influenced by organizational realities and constraints. They studied the constraints in a small and agile start-up software product company in relation to software testing practice. They describe how organizational reasons and reality shapes how the testing is organized in the company and highlight practical reasons for "bad" testing. Rooksby et al. (2009) studied testing as a social and organizational real-world practice. In their ethnography of testing in four organizations, they describe, with detailed examples, how testing work is carried out in different organizational, technical, and business contexts. They conclude that there is a correspondence between the act of testing and the organizational structure. Ahonen et al. (2004) studied software testing organization in three industrial cases and concluded that most challenges were not technical; instead, they related more to the process of test case design and the organizational model of the companies. Taipale et al. (2007) studied four industrial cases and found that the business organization affected the testing organization. Andersson and Runeson (2002) conducted an industrial qualitative survey of the state of the practice in the industry. They studied test processes, corporate values, tools, and environment. They found substantial differences between small and large companies especially related to the processes and the type of tools used.

2.2 Values of software testing

Regarding the values related to software testing, Martin et al. (2007) emphasized the socio-technical nature of software testing. In their study in a small and agile software product company, they describe how the organizational priorities affect the fundamental values of testing and, thus, how testing in practice is carried out. They concluded that a disconnect exists between software testing research and practice in small companies making software products. The researchers call for more work on "understanding testing as it happens." In their ethnographic descriptions, Rooksby et al. (2009) indicate how organizations' values or priorities affect the ways that they organize testing. Andersson and Runeson (2002) give general statements of the values and indicate that values toward verification and validation are ambiguous, but the choice is most often between quality and time. They also state that companies do not consider inexperienced staff suitable for verification and validation activities. Beer and Ramler (2008) studied the importance of experience in a multiple-case

study of three industrial software projects. They found that, in addition to testing knowledge, domain knowledge was deemed crucial in testing. Itkonen and Rautiainen (2005) studied three companies that had chosen to adopt an exploratory testing approach and found that such companies value the utilization of testers' application domain knowledge, testing from the end user's viewpoint, and fast feedback from testing to development.

2.3 Employee groups performing testing

We could not find direct studies of roles, or of employee groups, that perform software testing. However, Rooksby et al. (2009), as part of their ethnography of testing work in four organizations, describe the different organizational roles performing testing, e.g., programmers doing their own testing, (proxy) users doing testing, and full-time testers. Furthermore, in our prior work, we observed experienced testers (Itkonen et al. 2009) as they conducted exploratory testing. In that study, four of the eleven observed subjects were primarily testers, and the rest were from management, customer support, and development positions. This paper builds on this prior finding.

Research on the effects of groups or roles on detected defects includes research on test teams, organizational and geographical factors, and the life cycle of defects. Studies of software test teams (Andersson and Runeson 2007a; Jalote et al. 2007) have found that defects of components are also found by other test teams in addition to the one responsible for that particular component. Unfortunately, both studies omitted a detailed description of the test teams or the defects they found (e.g., the defect importance or fix rate). Furthermore, these works focus on different test teams rather than different employee groups.

Recently, Guo et al. (2010) studied organizational and geographical factors that affect the defect fix ratio. They found that organizational distance, reporter reputation, and whether or not the reporter was a permanent employee affected the defect fix ratio. However, the study did not consider the different employee groups (e.g., product manager, developer, and specialized tester) as a factor.

Aranda and Venolia (2009) analyzed the life cycle of defects at Microsoft. They studied how a defect was found and described the way in which it was resolved. Their results show that individual bug histories depend on social, organizational, and technical factors that cannot be found solely in defect databases. Our work has many similarities, but their work is more focused on the defect life cycle and coordination in the software teams than on understanding the organization of software testing or employee groups performing tests.

2.4 Summary

Based on prior work, we draw the following conclusions: The organization of testing has been studied in several works (Martin et al. 2007; Ahonen et al. 2004; Rooksby et al. 2009; Taipale et al. 2007; Andersson and Runeson 2002). Values that relate to software testing have been covered by (Martin et al. 2007; Beer and Ramler 2008; Itkonen and Rautiainen 2005; Rooksby et al. 2009). The employee groups that perform software testing in an organization and the effects of those groups or roles on the detected defects have been left unexplored. However, related work has covered software testers (Martin et al. 2007; Beer and Ramler 2008; Rooksby et al. 2009; Itkonen et al. 2009), industrial defect data (Jalote et al. 2007; Guo et al. 2010; Aranda and Venolia 2009; Andersson and Runeson 2007b), and the roles of employee groups in the context of SPI (Baddoo and Hall 2002; Jönsson and Wohlin 2005). To our knowledge, the distribution of defects between employee groups or roles has not been studied in an industrial context.

3 Methodology

This section presents the methodology of this study. First, we present the research questions that this study is about to answer. Second, we provide the case study method that was followed in this work. Third, we describe the case companies. Finally, we present our framework for defect database analysis and the rationale and practicalities behind it.

3.1 Research questions

The research objective of this study is to investigate software testing as a cross-cutting activity in software product development companies and provide empirical evidence on the matter. Additionally, we wish to provide credible insight of software testing performed in the industry.

The research questions are as follows:

- RQ1: How is testing organized in software product companies?
- RQ2: What is valued in software testing by the employees in software product companies?
- RQ3: What are the organizational roles of the employees who test software?
- RQ4: What differences exist in the number, importance, and fix rates of the reported defects based on these organizational roles?

In the literature, many definitions and goals for software testing can be found, and in most of them, defect detection plays a key role. In practice, various means of quality assurance are applied to detect defects in running software. In this paper, whenever we use the generic term *testing*, we include all QA practices that reveal defects that are reported to the defect tracking system. Our data include all reported defects from different sources in a development organization.

3.2 Case study method

This study was conducted as an exploratory case study (Runeson and Höst 2009) at three software product companies. It is considered an embedded case study that contains three units of analysis (i.e., each company represents a unit of analysis). The purpose of our case study was to provide initial answers to our research questions and to generate hypotheses for future studies. The research was conducted in six sequential phases: initial analysis, semi-structured interviews, interview data analysis, defect database analysis, case description reviews, and validating interviews. In addition to sequential phases, we also had other sources of information. This section describes the research phases and the other information sources.

In the initial analysis phase, we collected written documentation (e.g., defect reporting guidelines and the organizational charts that describe the roles of each of the companies' employees) and retrieved the snapshots of the defect databases at the case companies. The goal of the initial analysis was to obtain a general picture of the defect reporters, including the kinds of roles they had within the company and the distributions of defect types, and to get a general picture of the testing processes and the defect data that were available in the companies. We decided that the database analysis would focus only on data from the most recent year (2008), as older information could not be reliably discussed in the interviews. We selected the database sample based on the date a defect report was submitted in the database (i.e., included all reports that were submitted during year 2008). From one

company (case A), we had data from only 6 months because the company had switched to a new defect database.

Semi-structured interviews were conducted after forming an initial understanding of the case companies. We interviewed one product development manager and three employees who performed testing at each company in order to better understand the quality assurance process, values that drive the testing, the backgrounds of the people who perform testing, and the use of the defect database. The interviews were conducted with an interview guide approach (Patton 1990) with questions such as, "Which employees test the product?," "What other duties do these employees have?," "What are the backgrounds of the employees who perform testing?," "What type of testing processes do you have?," and "How is the defect database used?" The length of the product development manager interviews ranged from 86 to 130 min. The interviews with the employees who performed the testing had fewer questions and were 38 to 63 min in duration. The interviews with the managers covered a larger set of topics, and some issues were covered in more detail than those in the tester interviews (e.g., more time was spent on understanding the software testing process and the different employee groups). This approach was appropriate since the manager interviews determined the individuals who were selected for the tester interviews. All the interviews covered the planned topics, and the variation in the interview times is explained by whether or not the interviewee was talkative. We asked questions about the defect database in order to resolve any ambiguities that arose during the initial analysis of the defect database. At this stage, we found that the defects that were detected during development and unit testing are seldom entered into the database because developers fix them immediately after finding them. The defects that were not entered into a database are not included in this study.

In analysis of the interview data, the transcribed interviews were coded by using the topics of the interview guides as preformed codes as well as codes emerging from the data. In (Strauss and Corbin 1990), this step is referred to as open coding. Codes were then analyzed, and similar codes were combined into more general ones called axial coding (Strauss and Corbin 1990). This unification of coding was conducted in all cases in order to ease cross-case analysis. Coded transcriptions were then again analyzed.

After the interview analysis, the comprehensive database analysis was conducted according to what we learned from the interviews. A detailed framework for the database analysis is presented in Sect. 3.4.

Finally, after the analyses, case description reviews and validating interviews were conducted. The case descriptions were reviewed by the product development managers who were interviewed, as well as by the testers, when possible, in order to verify our interpretations. In the validating interviews, we discussed our findings, many of which are included in this paper, and presented some hypotheses regarding our findings to the development and quality managers of the case companies in order to get their insights into the phenomena that we observed. The validating interviews were conducted as both face-to-face discussions and e-mail interviews.

In addition, in order to support the analysis, we also incorporated other sources of information before and during the research. We had previously run quality goal workshops in the companies (Vanhanen et al. 2009). These workshops gave us an understanding of the developed products and quality requirements of the case companies prior to this research. We had several informal discussions with personnel in the companies through a research partnership before and after the interviews. The fact that we had several data sources and close collaboration with companies strengthens our results and reduces the possibility of misunderstandings.

3.3 Case companies

All of the companies included in our case study were successful medium-sized software product companies: Companies A and B had a strong market position in Scandinavia and Baltic countries and customers in other European countries. Company C was a worldwide market leader of software systems in its engineering domain. All of the companies were profitable and growing.

Each of the companies had more than 10 years of experience in their domains and had both domestic and foreign customers. The products in all companies were relatively mature as they all had been under development for 10 years or more. In all cases, the customers of the studied companies were companies or organizations in engineering, and the products were used by domain experts. Information about the case companies and the cases is summarized in Table 1. In the table and in this description, some information is purposely non-specific in order to ensure the anonymity of the cases.

The software products that each of the three companies developed were highly business-critical for their customers. Failures in such software could result in major financial losses for their customers and severely damage their core business processes. The product of company C was design software for a specific engineering field that was indirectly life critical (i.e., faults in the designs that were created and which used the system could have life-critical consequences).

The development processes in the case companies were iterative and incremental. All case companies issued periodic releases. Cases B and C also generated periodic *external*

Table 1 Summary of the case companies

	Company A	Company B	Company C
Personnel	>110 employees	>60 in the studied division (>300 in the whole company)	>70 in the studied divisions (>100 in the whole company)
Customers	>200	>80 for the studied products	>300
Company Age	>10 years	>20 years	>20 years
Studied product	Single product Business software of specific industry Integrated directly into the customers' other business systems Many customization opportunities	Two products for engineering in different fields The products share a common technological core architecture Integrated directly into the customers' other systems	Single product for engineering design Product has a separate core that is also used for another product COTS type of software (i.e., not heavily integrated or customized)
Release process	Internal monthly mainline release Majority of software development done in customer branch and later imported back to mainline Projects encouraged to frequently update to the latest mainline release	External main release two times a year External minor release four times a year Majority of software development conducted in main branch	External main release once a year External minor release once a year Majority of software development done in main branch

releases. In case A, the company issued monthly internal releases, but the majority of software development was done in customer projects from which new features were later merged back into the main line. In cases B and C, instead, the development was mainly conducted at the main branch. Even though their development processes were iterative and incremental, the companies did not follow any specific agile development process and their overall development approaches were more plan-driven than agile. The existing customer base was highly important for all of the case companies. For companies A and B, the deployment of new versions of the products to existing customers was a significant business issue, due to customer-specific integrations and customizations. The product development at the companies was mainly organized at one site, and no specific issues with global software development could be identified.

In this research, we wanted to select case companies that represent successful medium-sized software product companies. As the companies were similar in terms of their offerings (i.e., software products for domain experts with a rich user interface) and organization size, our case selection should be viewed as typical (Runeson and Höst 2009; Benbasat et al. 1987). We felt that typical cases would yield the most interesting answers to our research questions, as the literature currently offers an imperfect view of software testing in the industry. We had ongoing research cooperation with four such companies. We selected these three particular companies according to the availability of their defect databases. These organizations also represented some interesting differences (e.g., the use of specialized testers and the amount of customer-specific customization), which enabled comparisons to be drawn between the otherwise similar cases.

3.4 Defect databases and a framework for their analysis

The data sources in the defect database analysis were the operational defect databases in each company. These databases were used to record all defects or issues that were found in the developed software products, except for those identified during the development phases when developers fixed them right away during their programming work without the overhead of reporting to the defect database. The defect reports in the databases included defects on many types and levels, such as, for example, both omission defects (i.e., something missing) and commission defects (i.e., something wrong), as well as technical, verification-type defects, validation-type defect-related real users' processes, serious defects (e.g., application crashes and data loss), and less serious defects with minor consequences. One important aspect of defect database analysis is to distinguish feature requests and defects. In our case, only one of the companies, case A, used the same database for both defects and feature requests. In this case, we filtered out the feature request entries before analysis. However, the difference between a defect and feature request is not always clear cut. Consequently, in the majority of the real-world defect data sets, there are certainly issues that some might classify as a feature request and others that might classify as a defect. This means that our data set represents a holistic view of all reported defects, including reports that are related to users' viewpoints, not just discrepancies against technical specifications.

Various dimensions of testers' characteristics can be studied. Our study on employees who perform software testing comprises two dimensions, including an employee's primary title and duties in the organization (e.g., a sales representative) and the employee's software testing effectiveness (i.e., measured in terms of detected defects and defect characteristics). The results, in accordance with this framework, are shown in Tables 3, 4, 5).

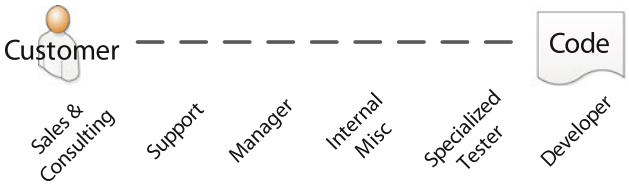

Fig. 1 Employee groups on the bipolar scale from customer to code

As the organizational structure of each company was different, we have classified employees who report defects, based on the interviews and organizational charts, into six groups. To make comparisons between companies possible, we derived categories that allowed us to use the same groups and classification in all three case companies. Classifications are based on employees' locations on a bipolar scale, in which the opposite ends are "customers" who use the program and "code" crafted by the developers, as illustrated in Fig. 1. This means that, in this classification, people are not matched to the groups directly and mechanically according to their job titles. Instead, we studied the actual roles, tasks, and organizational units of the employees and then classified them into the corresponding groups. In Table 2, the different job titles or organizational roles, whichever was accessible to us, are listed for each of the employee groups in our classification. It is clearly seen in the table that the same job titles can appear under different groups in our classification based on the employees' real roles and tasks in the organization. For example, in case A, there was an employee who worked in a specialized testing organization as a tester whose job title was software engineer.

Next, we describe the six groups of our classification in more detail. *Sales and consulting* roles are seen as being the closest to the customer because such positions require constant interaction and face-to-face communication with customers. *Support* roles are also constantly in direct contact with customers, but the information channel is e-mail or phone, rather than the richer face-to-face contact. Additionally, support personnel are usually contacted when a problem occurs, rather than when there is a need for something new, which is the case for sales and consulting. Next in line are people in roles that are not at the front lines of customer contact, but who are not primary programmers or testers of the software. Such people can have titles such as product manager, project engineer, usability expert, and technical writer. As many of these types of roles can be found in a company, those who fill them are classified as either *managers* or *internal misc.* due to their power and responsibility in the organization. *Developers* are seen as being the furthest from the customer[1] and closest to the code. Additionally, company A had specialized testers, and company C had used two hired testing consultants. Such people are classified under the group *specialized testers*. Finally, defects from customers are classified as *customer (ext).*

Several measures can be considered when evaluating individual testing performance. Undoubtedly, one can claim that the number and importance of the defects that were detected are measures of testing performance. Additionally, when considering the true contribution of testing efforts to software quality, the fix rate is an essential measure. If reporting findings in the defect database does not get defects fixed, the product quality is not improved; thus, the classification of the status of all of the reports that were submitted in the database (i.e., open, fixed, no fix) is considered important. Finally, the time it takes to

[1] This is not always the case, but our case companies preferred to protect the developers from direct customer demands.

Table 2 Employee groups and roles

Group	Company A	Company B	Company C
Sales and consulting	Customer consultant, Customer support engineer, System engineer, Sales manager, Senior consultant	Customer consulting, Sales services, Marketing	Sales manager, Key customer manager
Support	Customer support engineer, Quality assurance and project engineer	Customer support	Customer service, Application domain area architect, Customer service manager
Manager	Software development manager, Project manager, Product business director, Customer service manager	Product management	Manager of product quality, Product manager, Head of business unit, Business development director, Managing director, Research and development manager
Internal misc	Project engineer, Quality assurance and project engineer, Customer support engineer	Documentation, testing and Release	Customer service and product development
Specialized tester	Software test engineer, Software engineer, Test manager, Quality assurance engineer, Quality assurance and project engineer	–	External consultant
Developer	Software engineer, System engineer, Product manager	Software development	Application domain area engineer, Systems analyst, Senior systems analyst, Software architect

resolve a defect can be seen as a measure of defect importance, although it is affected by other things, such as the fixing process and the difficulty of fixing the defect.

Next, we explain the selected measures of testing effectiveness (columns in Tables 3, 4, 5) of the employee groups and the reasons for their existence. The first is the *number of employees* (n) from each group who reported defects. The second is the *total number of defects* from each group and the *median number of defects* per person. The defect count data were skewed (i.e., Pareto like); thus, the mean values would not describe the typical tester's results and are not presented. The third is the *importance of defects* found, according to role. The measured importance in this work is either severity or priority, whichever a company considered to be more reliable. The authors are aware that severity and priority measure different aspects of a defect, but since such separation was not available in the data, it made sense to use both as a measure of defect importance. The importance was decided by an individual reporting or handling the defect in cases A and C and jointly agreed in case B. Fourth, we present the *status of each defect*, whether it be open, fixed, or no fix. "Open" defects have not yet been resolved, "fixed" are defects that have been addressed by repairing them, and "no fix" defects have been resolved without a repair. The "open" class represents reports that are new and in progress; in other words, both defects that will eventually be fixed and defects that will not be fixed, or those that are not actual defects at all. "No fix" includes various reasons why a defect was not fixed: not able to reproduce, duplicate, not a defect, not able to fix, and fixing too expensive. The defect status data represent a snapshot of the defect data for the year 2008. For two companies (A and B), the date of the snapshot was 2 weeks after the end of the year. As those companies released products several times in the year (e.g., A issued twelve releases and B issued six releases), the release date is always close, but the effect of the release in

Table 3 Defect reports in case A (see Sect. 3.4 for detailed explanation)

Reporters		Defects		Importance (priority) (%)			Status (%)			Resolved
Group	*n*	Total	Median	Extra hot	Hot	Normal	Open	Fixed	No fix	Days median
Sales and consulting	9	145	15.0	14.5	35.9	49.7	16.6	69.0	14.5	14.0
Support	5	111	4.0	1.8	42.3	55.9	19.8	73.9	6.3	25.0
Manager	7	108	8.0	13.9	32.4	53.7	15.7	70.4	13.9	11.0
Internal misc	21	348	13.0	17.0	39.7	43.4	16.1	71	12.1	8.0
Specialized tester	9	367	50.0	2.2	25.3	72.5	24.3	58.0	17.7	12.5
Developer	22	134	3.0	9.7	11.2	79.1	17.9	77.6	4.5	14.0
Customer (ext)		458		7.2	7.4	85.4	11.8	76.4	11.8	3.0
Total	73	1,671	6.5	9.0	24.8	66.2	17.1	70.3	12.6	9.0

Table 4 Defect reports in case B (see Sect. 3.4 for detailed explanation)

Reporters		Defects		Importance (priority) (%)			Status (%)			Resolved
Group	*n*	Total	Median	Fatal (Now)	Next release	Later	Open	Fixed	No fix	Days median
Sales and consulting	10	117	10.5	2.6	61.5	35.9	13.7	73.5	12.8	6.0
Support	7	79	8.0	1.3	73.4	25.3	21.5	67.1	11.4	6.0
Manager	6	247	30.0	2.0	74.1	23.9	20.2	72.9	6.9	1.0
Internal misc	5	89	18.0	2.2	74.2	23.6	27.0	60.7	12.4	3.0
Developer	11	419	39.0	0.7	87.1	12.2	12.9	81.6	5.5	1.0
Customer (ext)		431		0.9	52.9	46.2	25.8	65.0	9.3	6.0
Total	39	1,382	17.0	1.3	70.3	28.4	19.7	72.0	8.3	3.0

Table 5 Defect reports in case C (see Sect. 3.4 for detailed explanation)

Reporters		Defects		Importance (severity) (%)			Status (%)			Resolved
Group	*n*	Total	Median	Fatal	High	Normal	Open	Fixed	No fix	Days median
Sales and consulting	4	136	34.5	0.0	25.7	74.3	34.6	44.9	20.6	16.0
Support	4	239	49.5	0.8	8.4	90.8	25.5	55.6	18.8	18.5
Manager	12	476	14.5	1.3	13.7	85.1	27.3	44.7	27.9	19.0
Internal misc	8	620	77.5	1.1	14.7	84.2	24.0	50.0	26.0	18.5
Specialized tester	2	117	58.5	1.7	3.4	94.9	44.4	35.9	19.7	31.0
Developer	17	282	12.0	2.1	8.9	89.0	22.3	57.1	20.6	10.0
Total	47	1,870	18.0	1.2	12.8	85.9	26.8	49.2	24.0	18.0

relation to the amount of open defects is smaller. For company C, the data were obtained 3 months after the end of the year. Because this company only released twice a year, there is considerable variation in the amount of open defects, depending on how close the release is. This 3 month buffer minimized this undesirable variation, which could have otherwise occurred. Finally, we present the *median time from defect report to resolution*. The distribution of the resolution time was also skewed (i.e., Pareto like), which is common for repair time distributions (Gokhale and Mullen 2010); thus, the median is favored over the mean value.

All of these measures were available from all the companies, and company representatives considered them to be the most reliable. However, concerning the quality of defect data, we confirmed that databases are susceptible to heavy individual variation as emphasized by the work of Aranda and Venolia (2009). That is, we were only able to analyze a strictly limited set of fields, since all databases in the case companies showed an unsystematic use of various fields. For example, case A previously had a defect database that contained all the defect data fields that a researcher required, but we quickly learned that the data were largely unusable because the majority of the data fields were sporadically utilized and, in many fields, the default values represented 90% of the data. For this reason, some fields in all cases were excluded from the defect database analyses. Using such data for research would be pointless; thus, only the most reliable fields were included in the analysis.

4 Results

In this section, we present the study results organized according to the case companies. For each company, we start by examining the organization of testing and the values behind it. After that, we proceed to the analysis of which employees test the software at each of the companies and what defects they found. Finally, we look at the distribution of defects among employees who perform software testing in terms of the number and importance of the defects and fix rates. Tables 3, 4, 5, which are included next to each case description, show how the reported defects are distributed to various employee groups.

4.1 Case A

4.1.1 RQ1: How is testing organized?

In case A, organized testing is conducted in two forms. First, the testing group, which consists of specialized testers in software development, conducts system and regression testing once a month for the internal mainline release of the product. Testing is based on test cases and is mostly manual, while some is partially or completely automated. Currently, these test cases lag behind the development of new features, as the majority of the testing group's time is spent on test executions and an analysis of results, rather than on the design of new test cases. The company plans to shorten the test execution cycle by automation and by hiring new employees.

Second, the new features that are developed in a customer project are tested by the project team as part of the system and acceptance testing of the project. This testing is conducted against each customer's requirements. These new features are later imported into the main product line.

4.1.2 RQ2: What is valued in testing?

The company values the discovery of fatal defects (e.g., crashes, data losses, and incorrect calculations), testing in a realistic environment (i.e., real customer data and real customer processes), the testing of new features by domain experts, and good-enough quality of mainline releases.

4.1.3 RQ3: Employees performing testing?

As discussed in Sect. 4.1.1, project engineers conducted testing on customer projects, while the testing group was responsible for mainline testing by maintaining and executing the system and regression test suite. Although formal responsibility for testing was assigned to some employees more often than others, a wide variation among the people who reported the defects was found. The employees who reported more than 15 defects had the following titles: customer consultant, quality assurance and project engineer, software development manager, project manager, software test engineer, and software engineer. In Table 3 and Fig. 2, we can see that people whose primary task was not testing (i.e., managers, sales and consulting, support, and development) found a considerable number of defects.

4.1.4 RQ4: Differences in the number, importance, and fix rates?

From the results, we can learn the following (see Table 3; Fig. 2): The median number of reported defects was highest in the *specialized testers* group; however, their defect fix rate was the lowest in the company (58% when the company average was 70%). Still, the testing effort of the specialized testers was valued in the company, according to the interviews, because it assured the stability of mainline releases and thus allows customer projects to migrate to the latest mainline version even in the middle of a project.

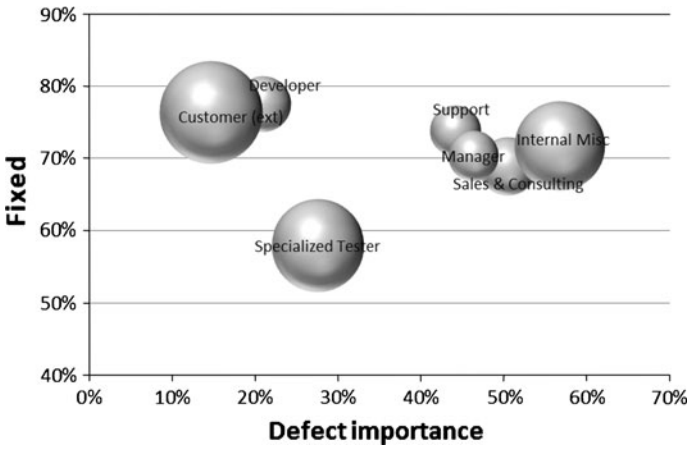

Fig. 2 Percentage of defects above normal importance (*x-axis*), fix rate (*y-axis*), and the total number of defects found by each reporter group (*bubble size*) in case A. Caveat: To show between-group differences, the *y*- and *x-axis* do not run from 0 to 100%. For exact data, see Table 3

Developers reported the smallest median number of defects and their defect importance. For example, the priority in case A was the lowest in the company, in terms of the percentage of "normal" priority defects in comparison with "hot" priority defects. The developers, on the other hand, seemed to be the best at getting their defects fixed (e.g., a fix rate of 78%), despite low defect priorities. *Support* also had a low median of detected defects, but further investigation of the group revealed that two individuals were responsible for 90% of the defects detected in that group. One should also notice that reports by customers have been separated from those issued by support.

The second highest defect priority, in terms of the percentage of "extra hot" and "hot" defects, came from the *sales and consulting* group; however, the great importance of defects for this group was not reflected in their defect fix rate or resolution time.

Internal misc had the highest priority for defects (i.e., the percentage of "extra hot" and "hot"). Project engineers, who are responsible for technical product setup and functional testing of customer projects, reported the majority of defects in this group; however, in the interviews, some company individuals also thought that the high defect priority was sometimes unnecessarily used to get more resources for the defect reporters' own projects.

Furthermore, the company's ability to charge a customer is often based on the amount of delivered functionality; thus, defects detected by *internal misc.* that prevent billing were immediately given high importance, and this observation is also reflected in the short resolution time. Occasionally, the importance of billing blocker defects was increased by the CEO of the company, who could demand immediate defect fixes in order to clear a pending invoice that was to be sent to the customer.

When looking at *external customer* defects, we can see that their priorities were the lowest in the company, in terms of the percentage of "normal" priority defects. This is explained by the fact that 75% of reports came from customers directly to the defect database, and customers do not have the ability to set defect importance. However, customers' defects clearly had top priority, as evidenced by the fact that their fix rate is among the highest and resolution time the lowest.

4.2 Case B

4.2.1 RQ1: How is testing organized?

In case B, all releases end with a testing period in which the system testing that is based on test cases and ad hoc regression testing is conducted. Testing is seen as a team effort, and the company does not have a separate testing organization or specialized testers in their organization.

The system testing based on test cases is meant to cover all the new or changed functionality that is produced during a release project. System test cases are designed by the same software developers who design and implement the functionality during the project. System testing is arranged, so that all cases are run by individuals other than those who designed them. The objective is to have personnel who interface with the customer (e.g., support, consultation, or product management) run as many test cases as possible.

The regression testing is conducted to ensure that all old functionality remains intact after development. The amount of work invested in this testing varies. Typically, regression and acceptance testing start when it is known which customers are affected and when they will start using the new version. The company reports that many defects are detected prior to release when customer consultants prepare for product demonstration and try to use the new features from the customer's viewpoint.

4.2.2 RQ2: What is valued in testing?

The company values realistic testing from the customers' viewpoint and against customers' processes, testing of high system availability (i.e., system reliability and updatability), and testing of performance under critical situations (e.g., when big data volumes are input into the system). Although the company has considered hiring specialized testers, it has purposely decided against such a possibility because of the deep requisite knowledge of the products' technical details and the necessary expertise in the domain and understanding of the customers' needs and processes are valued and critical for successful testing.

4.2.3 RQ3: Employees performing testing?

Case B had no separate testing group; instead, testing was seen as a team effort. A wide variety of people within the company reported defects. Table 4 indicates that, in terms of the number of reported defects, the most active testing groups were comprised of developers and managers. The employees with the following roles were found among those reporting 15 or more defects: sales services, product management, customer consulting, support manager, and software development.

4.2.4 RQ4: Differences in the number, importance, and fix rates?

As shown in Table 4 and Fig. 3, *developers* reported the highest number of defects in case B (excluding external customer group). We can see that the developer group had the lowest rate of defects prioritized in "later" releases. Developers' defects were also fixed, as they had the highest fix ratio (82%), the lowest no-fix ratio, and the lowest resolution time.

Although the *customer* (*ext*) group had the highest total amount of reported defects, it exhibited the lowest priority in terms of the percentage of defects in the "later" category, as well as a low fix rate. In case B, external customer defects were related to version upgrades, not tailored customer projects. This means that these defects were individual

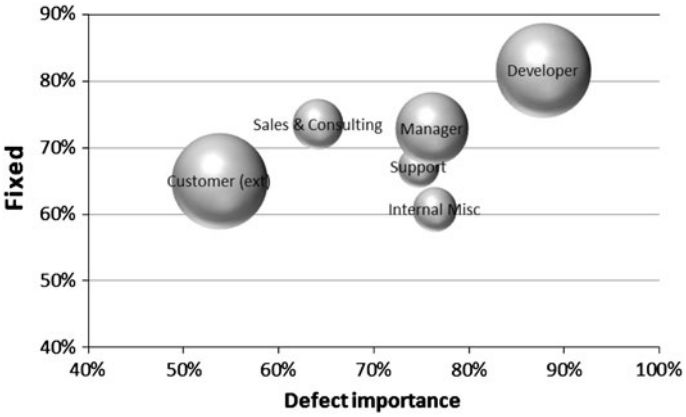

Fig. 3 Percentage of defects above later importance (*x-axis*), fix rate (*y-axis*), and the total number of defects found by each reporter group (*bubble size*) in case B. Caveat: To show between-group differences, the *y*- and *x-axis* do not run from 0 to 100%. For exact data, see Table 4

customers' problems with new versions that did not directly affect billing or contractual obligations.

During the interviews and quality goal workshops, we found that the company highly valued testing by people who had close contact with customers (i.e., *Sales and Consulting* and *Support* groups); however, from a numerical viewpoint, their testing was not very effective. Despite the evidence that their defects were often prioritized to be fixed in "later" releases, and their "no-fix" ratio was among the highest in the company, much of the testing that these individuals conducted occurred very late in the release project. Such a timeline made it more likely that their defects would be prioritized to be fixed in the later release rather than in the next one. Yet, they had the highest ratio of fatal defects in the company, although the absolute number of fatal defects was very small.

Managers and *internal misc* groups were similar in terms of the importance of defects. Even though managers had a higher median number of defects, they also seemed to be better at ensuring that the defects they detected would be fixed. Such an outcome is not surprising, as managers have more power and more to lose because they are responsible for a product or process.

4.3 Case C

4.3.1 RQ1: How is testing organized?

In case C, a quality manager leads the testing of the new or fixed functionality per release and is responsible for the overall quality of the product. The company has no specialized tester on the payroll but has occasionally hired external testing consultants. The quality manager has permission to use any of the department's personnel as testers according to their experience and knowledge.

The company's goal is to have features tested as soon as possible after they are completed. In some projects, the company has experimented and obtained positive testing results with the *demo-by-customer* practice (in this case, "customer" refers to internal customer). The idea is that, after each iteration, the customer tests the product by presenting a demo session to the development team. The company feels that this practice is superior to the demo-for-customer practice because it does not allow developers to hide quality problems, and it forces the customer to be more involved. The company also reported that many defects are found when consultants prepare for a presentation or demonstration of new product releases—a discovery that later influenced the creation of the demo-by-customer practice. However, it is a challenge to get internal customers involved and committed to the testing process because of their busy schedules and attention to other tasks that have priority.

The test cases have no detailed execution steps; instead, they focus more on the kinds of task that the user would be performing and the actual implementation is left to the individual who is executing the test. The product's user interface provides several ways to perform the same tasks. For an important function, several test cases will be given to different people in order to run to get different kinds of workflows tested.

In case C, the responsibility for regression testing of products is divided among all department personnel. By the end of a release, everyone should have tested the part of the product for which he or she is responsible. Each employee's responsibility also includes building test scripts for the automated regression testing of his or her part of the software.

4.3.2 RQ2: What is valued in testing?

The company values testing by internal customers from the end users' viewpoint, finding fatal defects (e.g., crashes, data losses, and accuracy of calculations), and team effort in testing. The company has also deliberately not formed a separate testing group because it values the domain and technical expertise more than specialized testing expertise.

4.3.3 RQ3: Employees performing testing?

Case C had no separate testing group; rather, testing was considered a team effort. However, as discussed in Sect. 4.3.1, the company had used external consultants to assist in testing efforts. In Table 5, it can be seen that managers and the *internal misc* group were clearly the most active testers in terms of the number of reported defects. A wide variation was found in the titles of the employees who reported defects. For example, sales representatives, managers, heads of business units, managers of key customers, quality managers, testers (i.e., external testing consultants), and systems analysts were the titles that reported more than 15 defects.

4.3.4 RQ4: Differences in the number, importance, and fix rates?

As shown in Table 5 and Fig. 4, people in the *internal misc.* group reported the highest median number of defects. This group included a group entitled "customer service and product development" that was comprised of individuals who were programmers with customer interaction responsibilities. They used the company's proprietary macrolanguage as they programmed features to the user interface level. The macrolanguage is comparable to the Visual Basic script in Microsoft Office products. The severity of the discovered issues, fix rate, and resolution time of the *internal misc.* group was very close to the company average.

The *developers* had the lowest median of defects per person. When looking at importance (i.e., severity in case C), we see that developers found the highest percentage of

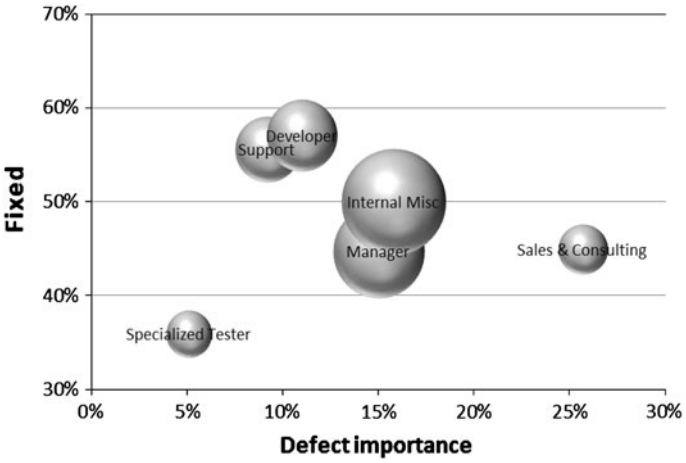

Fig. 4 Percentage of defects above normal importance (*x-axis*), fix rate (*y-axis*), and the total number of defects found by each reporter group (*bubble size*) in case C. Caveat: To show between-group differences, the *y*- and *x-axis* do not run from 0 to 100%. For exact data, see Table 5

"fatal" defects but, because of the low absolute number of defects in the "fatal" category, generalization is difficult. The severity of the developers' defects was lower than the company's average, in terms of the percentage of "normal" compared to "high" severity defects. Finally, the status of the developers' defects tells us that they had the highest fix rate (57%) and clearly the lowest resolution time in the company.

The two external test consultants, the *specialized tester* group, exhibited high defect detection effectiveness (median) even though they were hired for only a few months; however, their defect severities, in terms of the percentage of "normal" compared to "high," and their fix rate (36%) were the lowest in the company. The number of open defects identified by consultants is strikingly high, and the resolution times are also much longer than they are in other groups. The analyzed defect database snapshot was taken roughly 8 months after the consultants had stopped their testing efforts; thus, the company had had plenty of time to resolve and fix their findings if it had found the detected defects to be important.

The highest severities came from *sales and consulting*, but that did not help the group achieve a higher defect fixing rate or a short resolution time. Representatives of the company also thought that some people tended to overrate their defect severities, either unintentionally or intentionally. In case C, we could not separate out defects that were specifically reported by customers. Customer-reported defects are included in support's defects.

5 Discussion

In this section, we summarize our results and answer the research questions. We discuss our findings in contrast to related work and present the limitations of this study.

5.1 Answering the research questions

5.1.1 RQ1: How is testing organized in software product companies?

The main findings that relate to research question 1 are outlined in Table 6 (further details of the cases were presented in Table 1). All of the case companies issued periodic iterative releases: however, only cases B and C made periodic *external* releases. In case A, the company issued internal releases on a monthly basis, but the majority of software development was accomplished in customer projects from which new features were later merged back into the main line. Perhaps, stemming from that background, companies B and C both viewed testing as a team effort and did not employ specialized testers. Company A, on the other hand, needed a testing group to ensure the quality of internal mainline releases, so they could offer a reliable baseline for customer projects. Additionally, company A

Table 6 Organization of testing in the case companies

	Company A	Company B	Company C
Testing organization	Specialized testing group	No specialized testing group	No specialized testing group
	Feature testing in customer projects	Systematic testing at the end of each release	Systematic testing at the end of each release

conducted the testing of customer projects in which new features were first developed. In companies B and C, systematic testing was conducted at the end of every release. Company B reported no problems with this approach because it released the software every 2 months. Company C, on the other hand, had experienced problems with a 6-month release cycle when the testing was started and critical defects were discovered too late, forcing the company to postpone releases.

Previous work proposes that larger companies have a separate test group, whereas for smaller companies, testing is more a team effort led by an individual (Andersson and Runeson 2002). Our results suggest that the existence of a test group is dependent on the software product development model (e.g., periodic external releases vs. customer projects) rather than solely upon company size. For example, company B was clearly the largest (when considering the entire company), but neither of its two departments had a separate test group. Comparison against Ahonen et al. (2004) is difficult, as we did not assess the effect of the organizational model on the applicability of testing methods.

5.1.2 RQ2: What is valued in software testing by the employees in software product companies?

All of the companies seemed to value validation from the end users' perspective. We believe that this appreciation originated from several sources. First, the products have rich user interfaces for end users. They are intended for complex engineering domains and are operated by expert users. Additionally, all companies had seen cases in which a feature was implemented correctly, but it did not support the customer's way of working. Thus, people with a good understanding of the product and the customer process were considered to be indispensable. To cope with delayed releases, company C highlighted close collaboration and frequent testing by the internal customer, and it obtained good results with the demo-by-customer practice (see Sect. 4.3.1.). Another example came from companies B and C, which both pointed out that many defects are detected when sales and consulting personnel prepare for a product demonstration of an upcoming release. It is at that time when they try to use the product from the customer's perspective. Both companies strived to ensure that this type of testing is conducted earlier in the release. Verification techniques and the detection of critical issues such as crashes and incorrect calculations were considered to be highly important, but such issues were so infrequent and often quickly detected that the companies did not find them to be problematic. Since other minor defects that are related to verification were not seen as a top priority, it is clear that none of the companies was aiming for zero-defect software. Instead, the aim was to provide value for their customers and users.

A comparison of values to related work reveals the following: The work of Martin et al. (2007) and this study emphasize the importance of testing from the end users' perspective. Studies of industrial testers (Beer and Ramler 2008; Itkonen et al. 2009) highlight the importance of domain knowledge in testing, which was also highly valued in our case companies. Rooksby et al. (2009) describe, in their ethnographic examples, how testing is organized in four cases. Their descriptions include examples of deciding whether to use specialized testers versus testing by developers and what and what not to test. They described similar reasoning for these decisions, based on what is valued and the priorities of the organizations, as we have found in this work. From Taipale et al. (2007), we learn that a deep knowledge of testing is needed in product organizations. Our results support this statement.

5.1.3 RQ3: What are the organizational roles of the employees who test software?

In all three companies, employees from various groups participated in the software testing process, which is clearly indicated in the summary in Table 7. Their roles varied across the organization from developers to sales, and from managers to customer consultants. Even if the formal responsibility for testing was assigned more to some group of employees than to another, a wide variation in the people who reported defects was found. It is important to note that, in all cases, we found a considerable testing contribution by all of the identified reporter groups. The existence of a separate testing group or the organizational practice of hiring specialized testing consultants did not show as a strong peak in the testing contribution. For example, in case A, the separate testing group's defect numbers were only little higher than the next group (*internal misc*). This finding emphasizes that, in real software organizations, product testing is not a separated task of specialized testers. Instead, it seems to be a team activity in which a large number of people with varying organizational roles contribute and collaborate. This result supports the findings of Rooksby et al. (2009), who describe how testing in real-world circumstances involves people in different roles.

5.1.4 RQ4: What differences exist in the number, importance, and fix rates of the reported defects based on these organizational roles?

In this subsection, we discuss our observations on the differences in the defects that were detected by various employee groups. With respect to research question 4, our results show three main findings: First, defects that are discovered by developers have the highest fix rate. Second, defects that are discovered by people whose main task is testing have the lowest fix rate. Third, people with a personal stake of the software product (e.g., due to a close connection to the customer) tend to place greater importance upon the defects that they detect. The summary of the number and importance of the defects and the fix rates across all three cases is presented in Table 8 and Fig. 5.

When analyzing the developer's defects, we found that, in all cases, defects discovered by developers had the highest fix rate, even to the point that they exceeded the fix rate of the defects detected by the customers as shown in Fig. 5 and Table 8. We should bear in mind that defects found during unit testing and development were mostly fixed immediately before there was any chance that they would be inserted into the defect database. Thus, in reality, the developers' fix rates would be even higher. No single reason for this

Table 7 Distribution of defect reports between reporter groups

	Case A		Case B		Case C		Total	
Sales and consulting	145	8.7%	117	8.5%	136	7.3%	398	8.1%
Support	111	6.6%	79	5.7%	239	12.8%	429	8.7%
Manager	108	6.5%	247	17.9%	476	25.5%	831	16.9%
Internal misc	348	20.8%	89	6.4%	620	33.2%	1,057	21.5%
Specialized tester	367	22.0%	–	–	117	6.3%	484	9.8%
Developer	134	8.0%	419	30.3%	282	15.1%	835	17.0%
Customer (ext)	458	27.4%	431	31.2%	N/A	N/A	889	18.1%
Total	1,671		1,382		1,870		4,923	

Table 8 Summary of fix and importance percentages across all cases

	Fixed (%)			Importance[a] (%)			Total		
	Case A	Case B	Case C	Case A	Case B	Case C	Case A	Case B	Case C
Sales and consulting	69.0	73.5	44.9	50.4	64.1	25.7	145	117	136
Support	73.9	67.1	55.6	44.1	74.7	9.2	111	79	239
Manager	70.4	72.9	44.7	46.3	76.1	15.0	108	247	476
Internal misc	71.8	60.7	50.0	56.7	76.4	15.8	348	89	620
Specialized tester	58.0	–	35.9	27.5	–	5.1	367	–	117
Developer	77.6	81.6	57.1	20.9	87.8	11.0	134	419	282
Customer (ext)	76.4	65.0	–	14.6	53.8	–	458	431	–
Total	70.3	72.0	49.2	33.8	71.6	14.0	1,671	1,382	1,870

[a] Percentage of defects in the two highest-importance classes

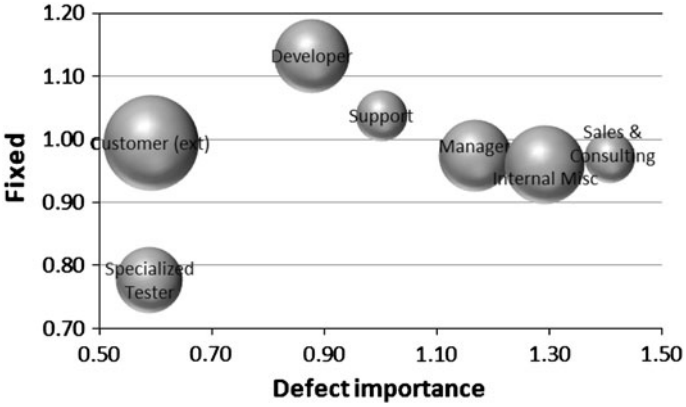

Fig. 5 Percentage of defects above normal importance (*x*-axis), fix rate (*y*-axis), and the total number of defects found by each reporter group (*bubble size*) in all cases. Data was normalized to company totals to enable cross-case merging (e.g., a value of 1.0 in both axis equals the companies' average). Caveat: To show between-group differences, the *y*- and *x-axis* do not run from 0 to 100%. For exact data, see Table 8

phenomenon was found in our study, but we generated several possible hypotheses, and they are explored with additional data in Sect. 5.2.1.

The second finding related to research question 4 was that defects detected by people whose main task is testing have the lowest fix rate. Company B did not utilize any specialized testers, so this finding is only based on data from two companies (see Table 8). We believe this phenomenon is linked to a tester's goal (i.e., to find as many defects as possible). Our interview data support this suggestion. Managers stated that specialized testers are capable of revealing and reporting large numbers of defects, but the relevancy of these defects to the end user is not always very high. Thus, based on the data, our hypothesis is: *Specialized testers are more likely to find and report defects that are unlikely to occur in real use or those that have only minor effects.*

Our third finding on research question 4 is that *people with a personal stake in the product tend to place their defects at a higher level of importance* than the company average, but it does not improve their defect fix ratio. This was noticed in cases A and C

(see Table 8). In our cases, we believe that this personal stake was created in two different ways. First, responsibility for a particular customer can explain higher defect importance. For example, in case A, the internal misc group had the highest group priority for defects, and many employees in that group were responsible for particular customers through their job as project engineers (see Sect. 4.1). Thus, the customer satisfaction was directly related to the defects that were found in this particular project. Second, the responsibility for a larger set of customers through close customer contact and product demonstration increases the personal stake and defect importance. People with close customer relationships tend to see the defects they report as higher priorities because the defects are more directly related to customer complaints or are based on their experience of what is important for the end users. Our interview data support this finding, as the interviewees agreed that, for example, the defects that were found by sales personnel during the preparation of sales demos and materials tend to be highly relevant from the customer and user viewpoints. The major challenge of utilizing the contributions of people near the customer interface was to motivate and get these people involved in testing before the last-minute demo presentations, which is far too late to have any defects that arise, repaired by any usual processes. Company C's innovative solution for this situation was the demo-by-customer practice. In case B, the importance of defects was collaboratively agreed upon in a weekly meeting. It is believed that this practice eliminated the perceived high defect importance because the personal stake of an individual had a much more limited affect.

Other studies of the roles of defect reporters in a commercial context are by Jalote et al. (2007) and Guo et al. (2010). Our result that developers' defects have the highest fix rate is similar to that of Guo et al. (2010), who found that the highest fix ratio occurs when the defect reporter and the first assignee is the same individual. This means that developers are most likely to fix their own defects. Guo et al. also found that the defect fix ratio decreases the further away the reporter is from the first assignee. Combining the findings of this paper with the results of Guo et al. suggests a hypothesis that the lowest fix ratios of defects are found by individuals who are testing specialists with a large organizational distance. However, further studies are obviously needed to confirm this hypothesis.

Comparison to the work of Jalote et al. (2007) is difficult because their organizational groups are different from ours. For example, they analyzed different types of testing teams and, overall, their contribution to this topic is brief since their paper focuses on process improvement rather than on the individuals who perform testing. However, our results on defect databases are in line with observations published by Aranda and Venolia (2009): in addition to the databases, one must understand the social, organizational, and technical context that is involved.

5.2 Hypotheses and additional findings

5.2.1 Developers' fix ratios

We also performed additional investigation to explore the possible causes for developers' high fix ratios. For this finding, we could derive four plausible hypotheses with which to explain the high fix rate of developers' defects in all three cases. In order to better understand the reasons behind this finding, we conducted a short round of interviews with the managers within the case companies. In these interviews, we asked the managers to comment on our initial hypotheses. Below, we present our hypotheses and the managers' opinions on them:

1. *Developers report only defects they know will be fixed.* Manager C did not believe this would be the case. Managers A and B agreed that, to some extent, this could be true. B also commented: "*At least developers often understand well how to fix the defect already when reporting it.*"
2. *Developers find defects in their own code and are more likely to fix their own defects than others.* Manager A agreed with this theory and added that "*it is also from their teammates' code where they are likely to find defects and whose defects they are likely to fix.*" Manager B agreed with this hypothesis and commented that, in the majority of defects, the same developer both reported and fixed the defect. Manager C also agreed and pointed out that some developers report issues directly for themselves and use the defect database as a task list.
3. *Developers work with the freshest code and are likely to find many defects that must be fixed or make further development impossible.* Manager A thought that the latter part (make further development impossible) was the best explanation of those that were offered. Manager B did not find this explanation to be plausible, and manager C agreed but commented: "*however, it is concerning that others besides developers do not run the latest code frequently enough… but, on the other hand, defects from customer service mostly originate from customers and one would think that these defects would have a higher priority.*"
4. *Developers are friends with each other and are more likely to fix each other's defects.* None of the managers agreed with this hypothesis.

In addition to interviewing managers, we analyzed the defect databases of companies B and C against hypothesis 2. From company A's database, we could not reliably determine who had fixed the defects. Defect database analysis revealed that 4 out of 11 developers in case B and 8 out of 17 developers in case C had fixed at least half of the defects that they had reported themselves. The total numbers of defects fixed by the reporter in cases B and C were 104 (25% of the defects reported by developers) and 60 (21%), respectively. Therefore, it seems that the self-reported defects only partially explain the developers' high fix ratios.

5.2.2 Internal competition and reliability of the defect importance data

Finally, during the interviews, companies A and C also described that defect importance was occasionally used as a tool in a company's internal competition for resources between products or projects. For example, a product manager is likely to put a higher priority on his or her defects in striving for more resources because he or she is responsible for the product and his or her job success is linked to product success. Whether the misuse of the importance of defects is intentional or unintentional is unclear. Another aspect we found concerning the quality of defect data, was that defect databases are susceptible to individual variation as all the databases in our analysis showed unsystematic use of various fields. Our experiences support the findings of Aranda and Venolia (2009), who report a severe unreliability within the defect database data. They identified missing and incorrect data as well as inconsistencies between electronic traces in the database and the actual activities concerning a defect. The implication of our findings concerning defect data quality is that defect importance is not only subjective, but it is also used as a mechanism for organizational politics. Both the subjectivity and this misuse of the importance data will be difficult to reduce with instructions and conventions, or automation alone. The collaborative agreement approach used by case B seems to be one good solution for this.

Furthermore, based on our experiences, we do not recommend that companies attempt to manually collect all the information on defects that might interest them. We believe that putting an additional burden on defect reporters will eventually turn into nothing more than yet another unreliable data source. Instead, we emphasize the need for automatic defect data collection. For example, the process phase in which defects are detected could be automatically recorded according to the time of entry and the project schedule, and the module or component of the application could perhaps be collected through an application plug-in. A sufficient set of such automatic features in mainstream defect databases could greatly improve the reliability of the defect data.

5.3 Limitations and validity assessment

This section presents the main limitations of the study by assessing its internal, construct, and external validity. Internal validity needs to be assessed when causal relationships are studied (Runeson and Höst 2009). The purpose of the study was not to establish causal relationships. However, we derived a few hypotheses of a causal nature, based on our observations (the development model leads to the existence of a testing group, developers' defects have the highest fix rate, etc.). However, since they came unexpectedly, we had no way to control them. We could present statistical tests of some of the findings, but we see it as misleading because it would be yet another form of fishing for statistical significance. In this type of study, we are bound to find significant differences simply due to chance. Thus, our causal findings are presented as hypotheses for future studies.

Construct validity is concerned with design of a study and whether the studied artifacts really represent what the researchers had in mind (Runeson and Höst 2009). We examined more than one company, thus, triangulation of companies is present. We had triangulation of data collection methods from all the companies: interviews, defect database, quality goal workshops (Vanhanen et al. 2009) and informal communication through our long research collaboration. This means that we understood the cases and their contexts over a longer period, which strengthens the results. The case descriptions were also verified by the companies. The research was conducted by three individuals, which, to some degree, prevented researcher bias. However, as we had close relationships with each other over a long period of time, it is possible that we are likeminded, which would increase the chance of bias.

One could question whether the defect database gives an accurate view of the detected defects. Discussions with the companies revealed that, with the exception of defects discovered before or at unit testing (which are excluded from the analysis), the databases should be as accurate as they can be. By this we mean that, in all companies, individual variations were found in defect reporting as a result of differences in work practices, politics, or personal differences. For example, some individuals would report even the smallest issues, while others, especially if working on a tight schedule, would leave some things unreported. Variations and limitations of defect database data has been reported in other studies (Aranda and Venolia 2009) and, based on our experiences, we think it is extremely difficult for a company to fully avoid such variation in defect reporting. In this study we could not affect how the data was collected in the defect databases, yet we analyzed the quality of the data of each case company and selected only the most reliable dataset for our analysis. Another threat to the validity of our data can be raised according to how we count the defect reports in the *fixed*/*open*/*no fix* classification. In our system, the classes of open and no fix also include reports that are not related to actual defects (e.g., misunderstanding, cannot reproduce, etc.), or are duplicates. This will affect the fix ratios

compared to situations in which only valid and verified defects would be counted. However, our classification reflects the realistic view of the contribution of different employee groups to the software quality. All defect reports, despite duplicates, use organizational resources; reports have to be addressed, investigated, and resolved. Thus, the number of reports in the no-fix class decreases the relative value of the testing contribution, which is reflected in the fix rate of the group in question.

Although we feel that the data sources are adequate, our analysis of the employees who perform the testing would have greatly benefited from the testing effort data. Work time reporting systems in the companies were used only for the purposes of customer billing and salary payment, and these did not contain such data. Another source of bias in our study was the problem of varying defect densities in different features and modules of the software undergoing the test. We had no way of controlling this because a defect's module information was not reliable in the databases.

External validity is concerned with the generalizability of the results (Runeson and Höst 2009). As prior studies of various software engineering topics have shown (Engström et al. 2008; Arisholm and Sjoberg 2004; Arisholm et al. 2007; Zimmermann et al. 2009), generalizing results is difficult in software engineering because of the effect of context. This is most likely the biggest shortcoming of this work as well. We carried out this research in a specific period of time and the results reflect the actual situations in the case companies at that time. It is possible that some situations we observed in some companies were temporary and exceptional, not representative of the case in question. However, based on our longer research partnership with all of the case companies over the course of several years, we feel that the situations described in our results are representative for the companies and not just temporary deviations in their development practices. We studied three companies that were similar in terms of their maturity and product type. Thus, we believe that the results are relevant to similar contexts (i.e., software *product* companies with relatively *mature* products, which has a *rich user interface* in terms of features and that is used by *domain experts*). However, it is unlikely that the results are transferable to completely different contexts, such as a company that makes real-time embedded software with no user interfaces.

6 Conclusions and future work

In this research, we studied the organization and values of testing as well as the organizational groups that contribute testing activities in small- and medium-sized software product companies. The context in our three case companies was the development of mature products with a rich interactive user interface for professional users. Based on the results of this case study, we draw the following conclusions:

Testing is not an action that is solely performed by specialists. In all our cases, people in roles that vary from sales to software development found a substantial number of defects and thus helped the companies improve their product quality. Two of the three companies deliberately did not hire specialized testers on the payroll and wanted to continue their practice of testing as a team effort. Consequently, it would be interesting to study the type of testing knowledge and support, such as the tools and techniques that could benefit these non-specialized testers when they conduct testing.

Validation from the viewpoint of end users is more valuable than verification aiming for zero-defect software, in the context of complex engineering products used by domain experts. In all companies, knowledge of the domain, the end user, and the customer process

and data was considered an indispensable part of software testing. This implies a need for further research in the area of domain-specific testing, and on the best ways to utilize the knowledge of domain experts in the testing effort. We aim to follow how the practice of keeping the internal customer responsible for testing works in the long run.

Developers' defects had the highest fix rate and specialized testers' defects had the lowest fix rate. Additionally, we found that people with a personal stake in the product (e.g., sales and consulting personnel) tend to place more importance on their defects than defect reporters in general, but it does not seem to improve their fix ratios. Prior work (Guo et al. 2010) has also shown that organizational distance affects fix ratios (i.e., the fix ratio is low when the distance between the defect reporter and the assigned developer is large). Future work should study the effect of role and organizational distance together in relation to defect fix ratios.

Defect database analysis is susceptible to organizational politics, i.e., competition of resources and overvaluing defects of one's own product or projects. As a possible solution for this problem, we suggest using group decision on defect importance—an approach used in case B, a company that did not suffer from this problem. Furthermore, we experienced that fields in the defect database are used sporadically, which supports the findings in existing research. We conclude that, instead of trying to collect more defect information through manual means, in the future one should strive for automated defect data collection as much as possible.

Acknowledgments The authors would like to thank Casper Lassenius, Timo Lehtinen, Tuomas Niinimäki, Jari Vanhanen, and the anonymous reviewers for their helpful comments. We acknowledge the efforts of the case companies who participated in our research and funding by TEKES and SoSE. A significant portion of this paper was written in Simula Research Laboratory, which provided an enthusiastic atmosphere for the corresponding author.

References

Ahonen, J. J., Junttila, T., & Sakkinen, M. (2004). Impacts of the organizational model on testing: Three industrial cases. *Empirical Software Engineering, 9*(4), 275–296.

Andersson, C., & Runeson, P. (2002). Verification and validation in industry–a qualitative survey on the state of practice. In *Proceedings of the 2002 international symposium on empirical software engineering* (pp. 37–47). Washington: IEEE Computer Society.

Andersson, C., & Runeson, P. (2007a). Investigating test teams' defect detection in function test. In *Proceedings of the first international symposium on empirical software engineering and measurement* (*ESEM*) (pp. 458–460).

Andersson, C., & Runeson, P. (2007b). A replicated quantitative analysis of fault distributions in complex software systems. *IEEE Transactions on Software Engineering, 33*(5), 273–286.

Aranda, J., & Venolia, G. (2009). The secret life of bugs: Going past the errors and omissions in software repositories. In: *Proceedings of the 2009 IEEE 31st international conference on software engineering* (pp. 298–308). Washington: IEEE Computer Society.

Arisholm, E., Gallis, H., Dyba, T., & Sjoberg, D. (2007). Evaluating pair programming with respect to system complexity and programmer expertise. *IEEE Transactions on Software Engineering, 33*(2), 65–86.

Arisholm, E., & Sjoberg, D. I. K. (2004). Evaluating the effect of a delegated versus centralized control style on the maintainability of object-oriented software. *IEEE Transactions on Software Engineering, 30*(8), 521–534.

Baddoo, N., & Hall, T. (2002). Practitioner roles in software process improvement: An analysis using grid technique. *Software Process Improvement and Practice, 7*(1), 17–31.

Beck, Kent. (2000). *Extreme programming explained*. Canada: Addison-Wesley.

Beer, A., & Ramler, R. (2008). The role of experience in software testing practice. In *Proceedings of the 2008 34th Euromicro conference software engineering and advanced applications* (pp. 258–265). IEEE Computer Society.

Benbasat, I., Goldstein, D. K., & Mead, M. (1987). The case research strategy in studies of information systems. *MIS Quarterly, 11*(3), 369–386.

Berner, S., Weber, R., & Keller, R. K. (2005). Observations and lessons learned from automated testing. In *Proceedings of the 27th international conference on Software engineering* (pp. 571–579). ACM.

Bertolino, A. (2007). Software testing research: Achievements, challenges, dreams. In *Proceedings of the 2007 future of software engineering* (pp. 85–103). IEEE Computer Society.

Cusumano, M. A., & Selby, R. W. (1995). *Microsoft secrets*. USA: The Free Press.

Engström, E., Skoglund, M., & Runeson, P. (2008). Empirical evaluations of regression test selection techniques: A systematic review. In *Proceedings of the second ACM-IEEE international symposium on empirical software engineering and measurement* (pp. 22–31). ACM.

Glass, R. L., Collard, R., Bertolino, A., Bach, J., & Kaner, C. (2006). Software testing and industry needs. Software IEEE *23*(4): 55–57.

Gokhale, S. S., & Mullen, R. E. (2010). A multiplicative model of software defect repair times. *Empirical Software Engineering, 15*(3), 1–24.

Guo, P. J., Zimmermann, T., Nagappan, N., & Murphy, B. (2010). Characterizing and predicting which bugs get fixed: An empirical study of microsoft windows. In *Proceedings of the 32nd international conference on software engineering* (pp. 495–504).

Iivonen, J., Mäntylä, M. V., & Itkonen, J. (2010). Characteristics of high performing testers: A case study. In *Proceedings of the 2010 ACM-IEEE international symposium on empirical software engineering and measurement* (p. 1). ACM.

Itkonen, J., Mäntylä, M. V., & Lassenius, C. (2009). How do testers do it? an exploratory study on manual testing practices. In *proceedings of the 2009 3rd international symposium on empirical software engineering and measurement* (pp. 494–497). IEEE Computer Society.

Itkonen, J., & Rautiainen, K. (2005). Exploratory testing: A multiple case study. In: *Proceedings of the international symposium on empirical software engineering* (pp. 84–93).

Jalote, P., Munshi, R., & Probsting, T. (2007). The when-who-how analysis of defects for improving the quality control process. *Journal of Systems and Software, 80*(4), 584–589.

Jönsson, P., & Wohlin, C. (2005). Understanding the importance of roles in architecture-related process improvement-A case study. In *Proceedings of international conference on product focused software process improvement* (pp. 343–357). Berlin: Springer.

Juristo, N., Moreno, A., Vegas, S., & Shull, F. (2009). A look at 25 years of data. *Software IEEE, 26*(1), 15–17.

Kettunen, V., Kasurinen, J., Taipale, O., & Smolander, K. (2010). A study on agility and testing processes in software organizations. In *Proceedings of the 19th international symposium on software testing and analysis* (pp. 231–240). ACM.

Martin, D., Rooksby, J., Rouncefield, M., & Sommerville, I. (2007). 'Good'organisational reasons for 'bad' software testing: An ethnographic study of testing in a small software company. In *Proceedings of the 29th international conference on software engineering* (pp. 602–611). IEEE Computer Society.

Myers, G. J. (1979). *The art of software testing*. New York: Wiley.

Patton, M. Q. (1990). *Qualitative evaluation and research methods*. Newbury Park: Sage Publishers.

Rooksby, J., Rouncefield, M., & Sommerville, I. (2009). Testing in the wild: The social and organisational dimensions of real world practice. *Computer Supported Cooperative Work (CSCW), 18*(5), 559–580.

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering, 14*(2), 131–164.

Strauss, A. L., & Corbin, J. (1990). *Basics of qualitative research: Grounded theory procedures and techniques*. Newbury Park: Sage Publishers.

Taipale, O., Karhu, K., & Smolander, K. (2007). Observing software testing practice from the viewpoint of organizations and knowledge management. In *Proceedings of the 1st international symposium on empirical software engineering and measurement* (*ESEM*) (pp. 21–30).

Vanhanen, J., Mäntylä, M. V., & Itkonen, J. (2009). Lightweight elicitation and analysis of software product quality goals—A multiple industrial case study. In *Proceedings of the third international workshop on software product management* (*IWSPM*) (pp. 42–52). IEEE Computer Society.

Zimmermann, T., Nagappan, N., Gall, H., Giger, E., & Murphy, B. (2009). Cross-project defect prediction: A large scale experiment on data vs. domain vs. process. In *ESEC/FSE '09: Proceedings of the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering* (pp. 91–100). ACM.

Author Biographies

Mika V. Mäntylä is currently a post-doctoral researcher at Lund University, Sweden. During the study, he was a project manager at the Department of Computer Science and Engineering, Aalto University, Finland. He received a D. Sc. degree in 2009 in software engineering from Helsinki University of Technology. His research interests include empirical software engineering, software testing, software quality assurance, defect databases and software evolution.

Juha Itkonen is a researcher and a doctoral student at the Department of Computer Science and Engineering, Aalto University, Finland. He received a M.Sc. degree in 2001 in software engineering from Helsinki University of Technology. His research interests include experience based and exploratory software testing, empirical software engineering, and quality assurance in agile context.

Joonas Iivonen is currently a senior systems analyst at National Institute for Health and Welfare, Finland. He received a M. Sc. Degree in 2009 from Helsinki University of Technology. During the study, he was working as a research associate for Helsinki University of Technology. His professional interests include software quality, software development methods, systems integration and data management.