Sending Out a Software Operation Summary: Leveraging Software Operation Knowledge for Prioritization of Maintenance Tasks

Henk van der Schuur, Slinger Jansen, Sjaak Brinkkemper
Department of Information and Computing Sciences
Utrecht University
Utrecht, The Netherlands
{h.schuur, s.jansen, s.brinkkemper}@cs.uu.nl

***Abstract*—Knowledge of in-the-field software operation is acquired by many software-producing organizations nowadays. While vendors are effective in acquiring large amounts of valuable software operation information, many are lacking methods to improve their software processes with such information. In this paper, we attempt to improve the software maintenance process by proposing a software operation summary: an overview of a vendor's recent in-the-field software operation, designed to support software processes by providing software operation knowledge. Particularly, we strive to improve prioritization of software maintenance tasks by fostering the reach of consensus between involved employees on such prioritization. Through an extensive survey among product software vendors in the Netherlands, we confirm the need for a software operation summary, and identify which crash report data are considered relevant as a basis for the summary when it is used for prioritization of software maintenance tasks. By means of a case study at a European software vendor, a software operation summary composed using crash report data is empirically evaluated. Results confirm the lack of consensus experienced between engineers and managers, and illustrate the value of the summary in fostering reach of consensus between employee roles, particularly in preparation of sprint planning and bug fixing activities.**

I. Introduction

One of the most time-consuming and challenging tasks in software maintenance is to reach consensus on the prioritization of software maintenance tasks. Many factors are involved in the process of software maintenance task prioritization, such as the number and names of customers that have reported a particular software failure, the severity and frequency of the failure as well as a software vendor's roadmap and available resources [1]. All too often, prioritization discussions and decisions of smaller software vendors are led by strong emotions of employees involved in the prioritization process, rather than by (f)actual knowledge. Employees may strongly focus on one particular aspect of software maintenance, and for example base their prioritization preferences on the interests and wishes of a large customer or on their image of the software code quality. Many software vendors are lacking in methods to improve such processes with objective data and knowledge, for example knowledge of in-the-field operation (and failures) of their software (i.e., software operation knowledge or SOK) [2]. As a consequence, reaching consensus on software maintenance task prioritization frequently is a time-consuming process, resulting in flawed prioritization decisions, unsatisfied customers and significant technical debt. The ISO standard on the software maintenance process [3] defines software maintenance as comprised of six activities, being *Process Implementation; Problem and Modification Analysis; Modification Implementation; Maintenance Review/Acceptance; Migration; Retirement*. In the context of this standard, software maintenance task prioritization activities as well as reaching consensus on such prioritization are covered by the *Process Implementation* activity.

The main contribution of this paper is the Software Operation Summary (SOS) concept: an overview of recent in-the-field software operation, which can be used for improvement of software processes. We improve software maintenance *process implementations* by fostering the reach of consensus in software maintenance task prioritization through this software operation summary concept.

The main research question is therefore '*Can prioritization of software maintenance tasks be improved through the concept of a software operation summary?*', which is answered through an extensive software maintenance survey held among product software vendors in the Netherlands. Based on survey results, we (1) confirm the lack of consensus on software maintenance prioritization experienced between engineering, management, and customer support, (2) establish the need for an SOS by these employee roles, and (3) identify which crash report data are considered relevant as a basis for an SOS that fosters reaching consensus on prioritization of software maintenance tasks [3], [4]. To qualitatively support survey results, we conducted a case study at a European software vendor, through which we composed an SOS based on crash report data and empirically evaluated its soundness and validity.

This paper is organized as follows. Section II further details the SOS concept. Section III describes our research approach; section IV details survey structure and contents. In section V, survey results are analyzed, while in section VI the case study results are presented. Next, we discuss research limitations (section VII) and place our work in context (section VIII). Finally, conclusions and future work are presented in section IX.

II. Software Operation Summary

We define a software operation summary as *an overview providing knowledge of recent in-the-field software operation, based on acquired software operation information*. The operation information needed for composition of the summary can be acquired by vendors from their software operating in the field, in a generic manner [5]. Both the operation information on which an SOS is based, as well as the operation knowledge provided by the summary are related to the in-the-field performance, quality or usage of the software, or to the feedback of end-users of the software [2]. An abstract SOS is provided in figure 1.

The software operation history, which forms the main component of the SOS, can be complemented with operation meta data such as the operation history timespan and software operation objectives that have been defined. As we show later in this paper, frequent use of a software operation summary (particularly, the software operation knowledge it provides) may foster reaching consensus between employees on software process issues. For example, it helps achieving consensus on prioritization of software maintenance tasks (i.e., determining when which bugs should be fixed, by whom).

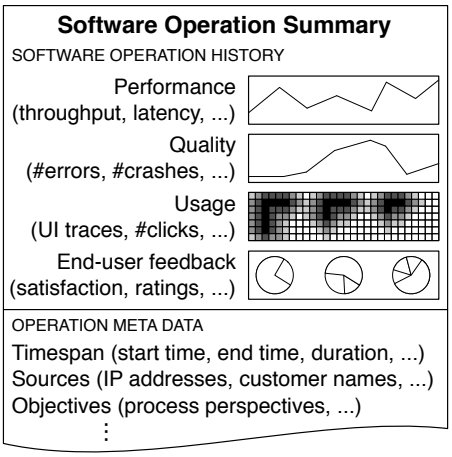

Fig. 1. Software Operation Summary concept

In industry, at least three types of employees are involved in software maintenance activities: customer supporters (filing bug reports based on feedback of customers experiencing in-the-field software failures), software engineers (developing and testing fixes for the reported failures) and development / product managers (guiding the maintenance process, being responsible for prioritizing maintenance tasks). Each of these employees may advocate a particular focus in prioritizing software maintenance tasks, and strive to improve a particular aspect within this focus. Customer supporters may be focused on increasing the satisfaction of end-users using the software. When prioritizing software maintenance tasks, customer supporters prioritize bugs that are experienced by the most influential and demanding customers. Software engineers, on the other hand, may be concerned with the quality of their software architecture and therefore prioritize bugs that are caused by poor or non-standard software design. Third, development / product managers may strive to increase the efficiency of the maintenance process itself, by maximizing the number of completed maintenance tasks and minimizing task duplication. By mapping those three employee foci to the SOK framework [2], three perspectives on software maintenance task prioritization can be observed. See table I.

	Engineering	**Management**	**Support**
Employee role	Software engineer	Development / Product manager	Customer supporter
Maintenance focus	Software architecture	Maintenance process	End-user
Improvement aspect	Quality	Efficiency	Satisfaction
SOK framework perspective	Development	Company	Customer

TABLE I
PERSPECTIVES ON SOFTWARE MAINTENANCE TASK PRIORITIZATION

In this paper, it is identified which crash report data are considered relevant as a basis for an SOS that fosters reaching consensus on prioritization of corrective and adaptive maintenance tasks [3], [4]. Crash report data are successfully used by Microsoft [6], [7], Apple [8], Canonical [9], and Google [10] to improve their software architecture quality, end-users satisfaction and maintenance process efficiency [11]. Which data are considered relevant, however, is situational and depends on many factors, such as the type of product, the type of customers and the type of employee using the data.

III. Research Approach

We structured the research approach using the SOK integration template method [12]. The method is designed for integration of software operation information in software processes (e.g. software maintenance). The instantiated method details the main research activities and related concepts, and therewith describes how our study can be repeated [13]. It is presented as a process-deliverable diagram (PDD) [14] in figure 2. Significant research activities and concepts are described below and in table II, respectively.

Operation information selection The first activity is concerned with determining which SOFTWARE OPERATION INFORMATION in crash reports is considered relevant in supporting the SOFTWARE MAINTENANCE TASK PRIORITIZATION process through a SOFTWARE OPERATION SUMMARY.
First, we acquired empirical understanding of this process by analyzing SOFTWARE MAINTENANCE TASK PRIORITIZATION processes in industry (see section II, as well as [12]) and answering questions like 'How is the process implemented in the organization?' and 'What are process dependencies?' (*Analyze target process*). Next, we used industry experiences and literature study results to identify issues with prioritization of software maintenance tasks, on which we based INTEGRATION OBJECTIVES (*Identify task prioritization issues in industry and literature*). Third, we categorized crash report data originating from various crash reporters (*Categorize crash report data*; see table IV) to identify SOFTWARE OPERATION INFORMATION demands of vendors

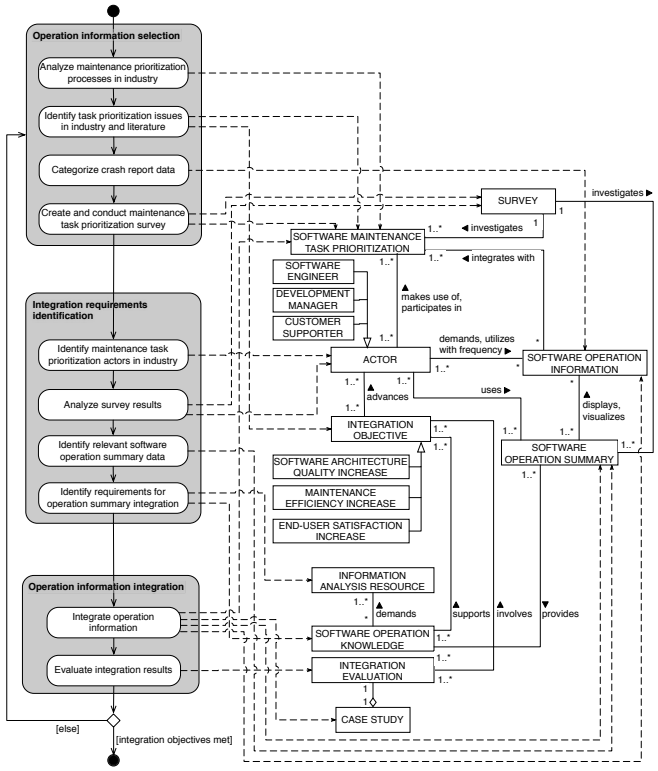

Fig. 2. Research approach based on SOK integration template method

acquiring crash reports. Based on the process analysis results and identified information demands, we have designed and conducted a SOFTWARE MAINTENANCE TASK PRIORITIZATION SURVEY (*Create and conduct maintenance task prioritization survey*; see section IV).

Integration requirements identification concerns identifying requirements for successful integration of the SOFTWARE OPERATION SUMMARY with SOFTWARE MAINTENANCE TASK PRIORITIZATION processes. First, we identified ACTORS involved with SOFTWARE MAINTENANCE TASK PRIORITIZATION by visiting software vendor sites (*Identify maintenance task prioritization actors in industry*; see table I). Next, we analyzed SURVEY results, for example to estimate how often SOFTWARE OPERATION INFORMATION would be used by the identified ACTORs (*Analyze survey results*; see section V). Based on SURVEY results and prior sub activities, relevant crash report data on which a SOFTWARE OPERATION SUMMARY can be based, were identified (*Identify relevant software operation summary data*; see section V-D). Finally, requirements for successful integration of the summary in SOFTWARE MAINTENANCE TASK PRIORITIZATION processes were identified (*Identify requirements for operation summary integration*; see section VI). For example, a requirement could be that valuable SOFTWARE OPERATION KNOWLEDGE is gained after integration of acquired operation information, demanding INFORMATION ANALYSIS RESOURCEs).

Operation information integration concerns integration

Concept name	Concept description
ACTOR	A person who demands and utilizes SOFTWARE OPERATION INFORMATION with a certain frequency, potentially visualized by a SOFTWARE OPERATION SUMMARY, and participates in SOFTWARE MAINTENANCE TASK PRIORITIZATION processes
CUSTOMER SUPPORTER	An ACTOR providing (technical) assistance with software products or services
DEVELOPMENT MANAGER	An ACTOR managing SOFTWARE ENGINEERs of a software development department
END-USER SATISFACTION INCREASE	One of the INTEGRATION OBJECTIVEs: an increase of the extent to which end-users believe the software available to them meets their requirements
INFORMATION ANALYSIS RESOURCE	A physical or virtual entity of limited availability needed to analyze and interpret SOFTWARE OPERATION INFORMATION and gain SOFTWARE OPERATION KNOWLEDGE
INTEGRATION EVALUATION	A systematic determination of merit, worth, and significance of the performed integration of SOFTWARE OPERATION INFORMATION using criteria against the set of defined INTEGRATION OBJECTIVEs involved
INTEGRATION OBJECTIVE	Goal of integration of SOFTWARE OPERATION INFORMATION with SOFTWARE MAINTENANCE TASK PRIORITIZATION involving SOFTWARE OPERATION KNOWLEDGE; generalization of the SOFTWARE ARCHITECTURE QUALITY INCREASE, MAINTENANCE EFFICIENCY INCREASE and END-USER SATISFACTION INCREASE INTEGRATION OBJECTIVEs
MAINTENANCE EFFICIENCY INCREASE	One of the INTEGRATION OBJECTIVEs: an increase of the extent to which wasted time and effort in the SOFTWARE MAINTENANCE TASK PRIORITIZATION process are avoided
SOFTWARE DEVELOPER	An ACTOR concerned with facets of the software development process, primarily (but wider than) design and coding.
SOFTWARE MAINTENANCE TASK PRIORITIZATION	The process of assigning priorities to software maintenance tasks by ACTORs, which is part of the *Process Implementation* activity of the ISO standard on the software maintenance process [3]
SOFTWARE OPERATION INFORMATION	Information resulting from data mining and abstraction of software operation data acquired from software operating in the field, possibly presented on a SOFTWARE OPERATION SUMMARY
SOFTWARE OPERATION KNOWLEDGE	Knowledge of in-the-field performance, quality and usage of software, and knowledge of in-the-field end-user software experience feedback [2]
SOFTWARE OPERATION SUMMARY	An overview of recent in-the-field software operation. Based on recent SOFTWARE OPERATION INFORMATION, the summary provides SOFTWARE OPERATION KNOWLEDGE
SOFTWARE ARCHITECTURE QUALITY INCREASE	One of the INTEGRATION OBJECTIVEs: an increase of the extent to which software is designed well, and how well the software conforms to that design

TABLE II
NAMES AND DESCRIPTIONS OF SIGNIFICANT CONCEPTS

of the SOFTWARE OPERATION SUMMARY with SOFTWARE MAINTENANCE TASK PRIORITIZATION processes. We conducted a CASE STUDY at a European software vendor to compose and evaluate a SOFTWARE OPERATION SUMMARY. First, the SOFTWARE MAINTENANCE TASK PRIORITIZATION process was altered to allow integration of relevant SOFTWARE OPERATION INFORMATION selected in 'Operation information selection' (*Integrate software operation information*; see section VI). Next, as part of the CASE STUDY, integration results were evaluated (*Evaluate integration results*; see section VI). If, based on a subsequent INTEGRATION EVALUATION, can be concluded that INTEGRATION OBJECTIVEs are met, the result of this activity is a SOFTWARE MAINTENANCE TASK PRIORITIZATION process that is effectively supported by acquired operation information. Otherwise, the method is reinitiated with the 'Operation information selection' activity.

This paper's main research question ('*Can prioritization of software maintenance tasks be improved through the concept of a software operation summary?*') is answered based on evaluation of the following six propositions:

P1 A software operation summary is expected to integrate with current software maintenance practices.
P2 A software operation summary is expected to increase knowledge of software architecture quality.
P3 A software operation summary is expected to increase knowledge of end-user satisfaction.
P4 A software operation summary is expected to increase knowledge of maintenance process efficiency.
P5 A software operation summary is expected to foster achieving consensus on software maintenance task prioritization.
P6 A software operation summary is expected to reduce the time needed for software maintenance task prioritization.

Sections	Questions	Propositions
General	1. How many people are employed at your company? [Less than 5; 5 to 10; 10 to 20; 20 to 50; 50 to 100; 100 to 200; More than 200]	N/A
	2. How many end-users are using software that is produced by the company you are employed by? [Less than 50; 50 to 200; 200 to 500; 500 to 2,000; 2,000 to 10,000; 10,000 to 100,000; More than 100,000]	N/A
	3. How many crash reports are received weekly by the company you are employed by? [Less than 50; 50 to 200; 200 to 500; 500 to 2,000; 2,000 to 5,000; 5,000 to 10,000; N/A]	N/A
	4. What is your role within the company you are employed by? [Engineering (software developer, software engineer, software architect, etc.); Management (Development manager, product manager, etc.); Support (customer supporter, help desk employee, etc.)]	N/A
	5. How many years of experience do you have in the field of information technology? [0..75]	N/A
Current situation	6. How much time do you and your colleagues spend weekly on prioritization of software maintenance tasks? [Less than 1 hour; 1 to 5 hours; 5 to 10 hours; More than 10 hours]	P1
	7. How frequent do you experience a lack of consensus on prioritizing software maintenance tasks with each of the following colleagues? [Never (1); Rarely (2); Monthly (3); Weekly (4); Daily (5); N/A] • Engineering • Management • Support	P1
	8. To which extent does knowledge of your software architecture play a role in prioritizing software maintenance tasks within your organization? [1 (Very minor role); 2; 3; 4; 5 (Very major role); N/A]	P1
	9. To which extent does knowledge of end-user satisfaction play a role in prioritizing software maintenance tasks within your organization? [1 (Very minor role); 2; 3; 4; 5 (Very major role); N/A]	P1
	10. To which extent does knowledge of the efficiency of the maintenance process within your organization play a role in prioritizing software maintenance tasks within your organization? [1 (Very minor role); 2; 3; 4; 5 (Very major role); N/A]	P1
Expectations	11. How often could you, in your current role, well use a software operation summary? [Never (1); Rarely (2); Monthly (3); Weekly (4); Daily (5); N/A]	P1
	12. With which of your activities is a software operation summary of most use to you? [...]	P1
	13. To which extent do you expect that a software operation summary increases your knowledge of each of the following: [Certainly not (1); probably not (2); Possibly (3); Probably (4); Certainly (5); N/A] • Software architecture quality • End-user satisfaction • Maintenance process efficiency	P2 P3 P4
	14. To which extent do you expect to save time on prioritization of software maintenance tasks with a software maintenance summary? [Certainly not (1); probably not (2); Possibly (3); Probably (4); Certainly (5); N/A]	P6
	15. To which extent do you expect information contained in a software operation summary to foster the reach of consensus on prioritization of software maintenance tasks, with each of the following colleagues? [Certainly not (1); probably not (2); Possibly (3); Probably (4); Certainly (5); N/A] • Engineering • Management • Support	P5
Crash report data	16. Indicate which data types in your opinion contribute most to successful execution of the following activities: [List of data types listed in table IV] • Determining which bug will result in the largest increase of software architecture quality, once fixed • Determining which bug will result in the largest increase of end-user satisfaction, once fixed • Determining which bug will result in the largest progress in the software maintenance process, once fixed	P2 P3 P4

TABLE III
SURVEY QUESTIONS AND PROPOSITIONS

The propositions are evaluated based on results of the software maintenance task prioritization survey we conducted.

IV. MAINTENANCE TASK PRIORITIZATION SURVEY

As part of the *Operation information selection* activity described in section III, a software maintenance task prioritization survey was designed and conducted to evaluate the six propositions and to establish which operation information from crash reports is considered relevant by the three employee roles. The survey was reviewed and tested by peer researchers and target respondents before publication. To acquire survey respondents, the survey was announced to our educational and professional networks, for example via an expert focus group consisting of chief technology officers, product managers and senior team leaders from industry. Also, the survey was noticed and advertised by press in the Netherlands [15]. The survey was composed of 16 questions divided over four sections (see table III):

General The survey started with five questions to identify the respondent, as well as the software vendor the respondent is employed by. The first three questions were asked to get insight in the size of the company the respondent is employed at, in terms of number of employees, end-users and received crash reports. The answer sets of these questions (as denoted between square brackets in table III) were deducted from the definition for small and medium-sized enterprises (SMEs) of the European Commission [16]. The answer options of question 3 contained a 'Not Applicable' option for vendors not receiving any crash reports. Question 4 was asked to identify the role of the respondent within its employing company and could be answered with three options that correspond to the roles in table I. Finally, question 5 was an open question querying the respondent's experience in information technology. This section's questions correspond to none of the propositions.

Current situation This section is composed of five questions that were asked to establish the situation at the respondent's organization regarding prioritization of software maintenance tasks, and identify potential improvement areas. The first two questions respectively queried the amount of time that is spent on software maintenance prioritization tasks by the respondent, as well as the frequency with which the respondent experiences a lack of consensus with colleagues on prioritizing software maintenance tasks. Questions 8, 9 and 10 each refer to a particular prioritization focus and improvement aspect in table I and were asked to confirm the importance of the maintenance foci within the respondent's company. Each of these questions correspond to an identical set of answer options that is equivalent to a five-point Likert item (see table III). All questions in the *Current situation* section were asked to evaluate proposition **P1**.

Expectations As an introduction to the questions in this section, the software operation summary concept was introduced to respondents before asking the questions by providing a basic definition of the concept, as well as a description of three of its goals (providing knowledge of (1) software architecture quality, (2) end-user satisfaction, (3) maintenance process efficiency). Five questions related to expected or potential use of a software operation summary

Vendor name		Microsoft [6], [7]	Apple [8]	Canonical [9]	Google [10]	ERPComp	CADComp
Crash reporter form		Operating system service			External library	Product software feature	
Category	**Unified data type**	**Data type instance**					
Software	Application name	Identifier	General AppName	Package	–	–	Cause
	Build version	Build Info	Version Information	StackTrace, Package	MODULE name	–	BuildVersion
	Code file and line number causing the crash	Backtrace	–	StackTrace	FILE name	–	CauseLocation
	Code file and line number presenting the crash (to end-user)	Backtrace	–	–	FILE name	–	–
	Database	–	–	–	–	Database	–
	Edition	–	–	–	–	–	Edition
	Error code	Exception Codes	Exception Code	–	–	–	–
	Error message	–	–	–	–	Exception details	–
	Error type	Exception Type	–	–	–	Exception Type	ExceptionType
	Last user action (click, command, etc.)	Backtrace	–	StackTrace	FUNC name	Action name	–
	Last operation context (page, screen, etc.)	–	–	–	–	Page name	–
	Localization	–	–	–	–	–	Localization
	Memory address	Exception Codes	Exception Address	–	FUNC address	–	MemoryLocation
	Module causing the crash	Backtrace	–	StackTrace	–		CauseModule
	Module presenting the crash (to end-user)	Backtrace	–	–	–	Page name	–
	Operation environment (operating system, browser)	OS Version	System Information	DistroRelease, ProcEnviron	MODULE operatingsystem	OS, Browser, page data	–
	Process name	Process Identifier	–	–	–	–	–
	Processor architecture	Code Type	System Information	–	MODULE architecture	–	ProcessorType
	Session type	–	–	–	–	Session data	–
	Thread name	Crashed Thread	–	–	–	–	–
	Version	Version	Version Information	Package	–	–	–
End-user	Comments accompanying report	–	–	–	–	–	EnduserComments
	Company name	–	–	–	–	Customer	CompanyName
	Country	–	–	–	–	–	SourceCountry
	IP address	IP address	–	–	–	IP address	IPAddress
	Language	–	–	–	–	–	Language
	License number	–	–	–	–	–	LicenseNumber
	User name	–	–	–	–	User	UserName

TABLE IV
CATEGORIZATION OF CRASH REPORT DATA TYPES

were asked in this section to establish the need for, and potential of such a summary based on crash report data. Particularly, questions 11 and 12 were asked to establish expected integration of a software operation summary with the respondent's current activities and requirements. We chose to let question 12 be an open question to prevent bias of respondents, which, for example, could be induced by providing a predefined set of activities. Questions 13, 14 and 15 query the respondent's expectations concerning the effects of using a software operation summary, respectively in terms of knowledge increase, time saving and consensus reach. The answer options of these three questions represent a five-point Likert item (see table III). Questions 11 and 12 were asked to evaluate proposition **P1**, question 13 to evaluate propositions **P2**, **P3** and **P4** and questions 14 and 15 were asked to evaluate propositions **P5** and **P6**, respectively.

Crash report data The last section of the survey is concerned with what crash report data are considered relevant input for prioritization of corrective and adaptive software maintenance tasks. A list of answer options in the form of unified crash report data types was assembled based on crash report data types of existing, widely-used and widely-known crash reporting techniques. See table IV.
Crash reporting techniques from Microsoft [6], [7], Apple [8], Canonical [9] and Google [10], as well as those of two European software vendors called ERPComp and CADComp (actual vendor names have been anonymized for confidentiality reasons), were analyzed and compared to assemble the answer list with data types that was presented to respondents as an introduction to this section. ERPComp produces an online ERP solution that is used by about 17,000 customers; CADComp produces an industrial drawing application targeted on the Microsoft Windows platform and is used by more than 4,000 customers in five countries. Both vendors are headquartered in the Netherlands and were visited on-site.
To ease understanding and answering of this question for respondents, data types were categorized into two categories: software and end-user. One additional answer option 'Other' was added to the list of 28 crash report data type answer options, to allow respondents to add additional data types matching their crash report data structure. Question 16 was asked to evaluate propositions **P2**, **P3** and **P4**.

As listed in table III, questions 7, 13, 15 and 16 had to be answered for three question components: questions 7 and 15 had to be answered for each of the software maintenance task prioritization perspectives (Engineering, Management and Support; see table I) and question 13 and 16 had to be answered for each of the prioritization foci and improvement aspects of these perspectives. These questions were designed as such to expose relations between employee roles, maintenance foci and improvement aspects in reaching consensus on prioritization of software maintenance tasks.

V. ANALYSIS OF SURVEY RESULTS

In total, 136 respondents from about 72 different software-producing companies[1] participated in our survey ($n = 136$). All of the respondents completed section *General*, 111 (81.6%) completed *Current situation*, 97 (71.3%) completed *Expectations* and 79 (58.1%) completed the final section of the survey, *Crash report data*. Those percentages should be

[1]Survey respondents participated with 72 unique IP addresses (addresses that only differed on third or fourth octet were considered identical).

taken into account in further analysis of survey results, which is provided in the following sections.

A. General

Most respondents are employed by a vendor that employs more than 200 people (49.3%) or 20 to 50 people (15.4%). Almost three-quarter of the respondents (72.1%) is employed by a vendor that serves more than 100,000 end-users (35.3%), 10,000 to 100,000 end-users (18.4%) or 2,000 to 10,000 end-users (18.4%). Most vendors (66.9%) receive less than 50 crash reports (52.9%) or between 50 and 200 (14.0%) crash reports per week (19.1% of the respondents answered 'N/A' to question 3). Respondents were evenly distributed over employee roles: 28.7% identified its role as part of Engineering (12.4 years of experience on average, $\sigma = 7.6$), 36.0% as part of Management (15 years of experience on average, $\sigma = 7.2$ years), and 35.3% as part of Support (9.6 years of experience on average, $\sigma = 8.9$ years). In total, respondents have 1676 years of experience (12.3 years on average, $\sigma = 8.3$ years).

B. Current Situation

Of the 111 respondents that have answered the first question of this section, most (55.9%) spend 1 to 5 hours weekly on prioritization of software maintenance tasks. 24.3% indicated to spend less than 1 hour on these tasks weekly, while 10.8% indicated to spend more than 10 hours weekly on prioritization of software maintenance tasks.

Figure 3a visualizes the lack of consensus experienced by the responding software engineers, customer supporters and managers, as inquired through question 7. Lack of consensus is most frequently experienced by (1) support with engineering (3.17 on average: more often than monthly), (2) engineering with management (3.00 on average: monthly) and (3) management with engineering (2.93 on average: less often than monthly). Also, engineering and management experience lack of consensus with respectively management and engineering most frequently, compared to other roles. We believe the cause of this outcome is related to the nature of the challenges that are faced by the two roles (i.e., technical versus managerial). Except for support, employees least frequently experience lack of consensus with employees having the same role.

Results of questions 8 to 10 (role of maintenance foci and corresponding improvement aspects) are displayed in figure 4. It is remarkable that for all employee roles, end-user satisfaction plays the largest role in prioritizing software maintenance tasks, software architecture quality the second-largest, and maintenance process efficiency relatively the smallest role. Comparing employee roles per maintenance focus, it should be noted that software architecture quality plays the lowest role for engineering. This may be caused by the confidence software engineers have in their work: of the three employee roles, engineers have the most knowledge of, and may be the most confident about the quality of their software architecture. Maintenance process efficiency plays the highest role for support (possibly because they wish as much in-the-field software failures reported by customers as possible to be fixed

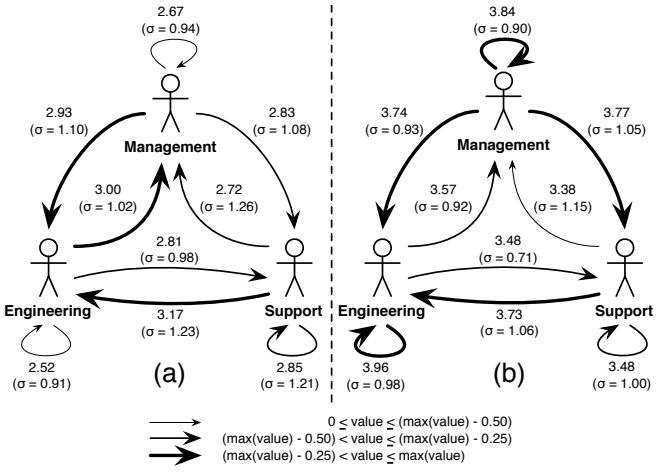

Fig. 3. (a): average frequency with which lack of consensus is experienced between the different employee roles (question 7); (b): average extent to which the employee roles expect SOS information to foster the reach of consensus (question 15). Thicker arrows indicate higher frequencies and expectations

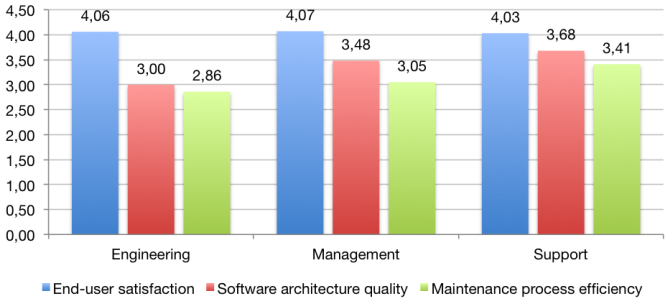

Fig. 4. Importance of the maintenance foci for the different employee roles (questions 8–10)

as soon as possible), and end-user satisfaction is approximately equally important for all employee roles.

Although the lack of consensus between the three employee roles can therefore not be justified by *absolute* differences in maintenance foci, it could be justified by *relative* differences. For example, the role software architecture quality plays in prioritization of software maintenance tasks increases as employee roles have more direct communication with customer's end-users. In software-producing organizations, customer supporters are typically directly confronted with bugs and crashes experienced by end-users than management. Although engineering departments typically provide third line support, communication with end-users is less direct and frequent than between management and end-users.

C. Expectations

All employee roles would like to use a software operation summary about every three weeks (average usage frequencies are all more often than monthly: engineering 3.46 ($\sigma = 0.76$), management 3.59 ($\sigma = 0.78$), support 3.39 ($\sigma = 0.90$)). To illustrate the potential applications of such a summary, we created a frequency list of all responses to question 12. Based on analysis of this list, a software operation summary

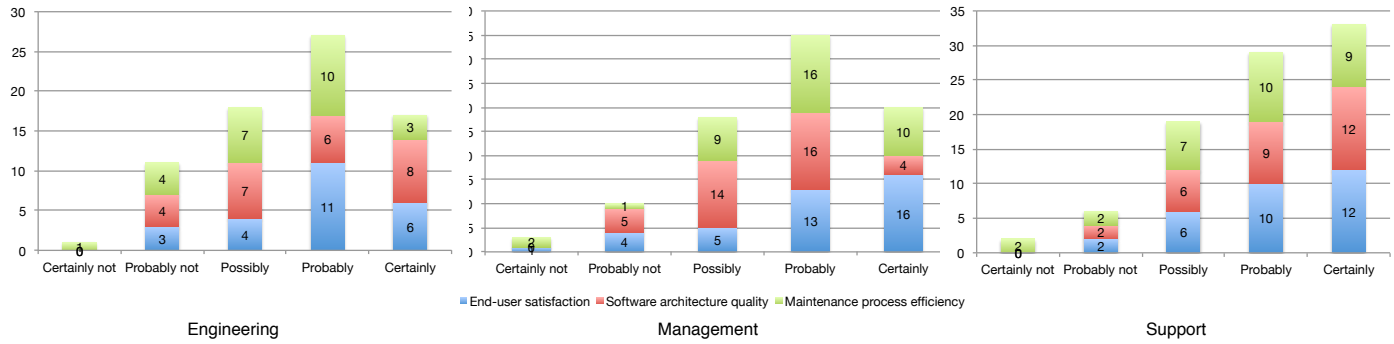

Fig. 5. Expectations per employee role regarding the extent to which a software operation summary would increase their knowledge (question 13)

is expected to be most of use during activities related to sprint and release planning, bug fixing and software development, as well as during evaluations (of, for example, software operation at a particular customer, or the most recent sprint).

Expectations of engineering, management and support employees regarding the extent to which a software operation summary could increase their knowledge of software architecture quality, end-user satisfaction and maintenance process efficiency (question 13) is visualized by figure 5. With an SOS, all employee roles expect to save time on software maintenance task prioritization to reasonable extent: the modus of the answers to question 14 was 'Possibly' for engineering (average: 3.5, $\sigma = 1.00$), and 'Probably' for management (average: 3.38, $\sigma = 0.92$) and support (average: 3.24, $\sigma = 0.97$). Figure 3b visualizes the extent to which the different employee roles expect information in a software operation summary to foster the reach of consensus (question 15). First of all, the parties that experience a lack of consensus the most (support with engineering, engineering with management, management with engineering), expect an SOS to foster reach of consensus with the colleagues with which they experience this lack of consensus. As opposed to support, engineering and management both expect an SOS to foster reach of consensus the most with colleagues that have the same role. Of all employee roles, management has the highest expectations in reaching consensus with other employee roles. In absolute terms, highest expectations are expressed by engineers, expecting an SOS to foster reach of consensus with other engineers with an average of 3.96 ($\sigma = 0.98$, modus = 'Certainly'). Finally, most employees consider an SOS fostering the reach consensus with colleagues on software maintenance task prioritization as probable (modus = 'Probably').

D. Crash Report Data

As stated in section IV, the last survey section of the survey is concerned with identification of which crash report data are considered relevant as a basis for an SOS that fosters reaching consensus on prioritization of software maintenance tasks. In view of this fact, we focus on the most-selected data types per maintenance focus *per employee role*. The more employees differ in data type selection, the more data types are involved in determining which bugs should be fixed to reach the largest increase of the three maintenance foci. See figure 6.

Regarding the software architecture quality focus, the most-selected data types are all related to software failure cause identification: data types 'Code file and line number causing the crash' and 'Module causing the crash' were most selected by engineering and management, while support selected data types 'Application name' and 'Thread name' the most. End-user satisfaction data types are more related to identification of the customer: 'Company name' and 'Application name' were data types most selected by engineering and management, while support selected data types 'Company name' and 'Comments accompanying report' the most. Finally, concerning maintenance process progress, some most-selected data types are related to software releases over time (i.e., 'Build version' and 'Version'). 'Build version' and 'Code file and line number presenting the crash' were data types most selected by management, 'Build version', 'Database' and 'Module cause crash' were most selected by engineering, and support selected 'Code file and line number presenting the crash', 'Code file and line number causing the crash' and 'Version' the most.

In total, four additional data types were suggested by respondents using the 'Other' field: (number of crash reports per) 'Performance peak, Slow request', 'Code file / class', 'GUI element' and 'Software configuration'. While the first suggestion is out of the scope of this paper (not being part of typical crash report data) and the second was already part of the presented data types (code file and line number, module), the latter two are taken into account for future research.

Focussing on the expectations for a software operation summary that is designed for fostering consensus on prioritization of software maintenance tasks, survey results can be summarized as follows:

- All employee roles would like to use such an SOS about every three weeks, particularly in preparation of (sprint) planning, (software) development and bug fixing
- Such an SOS is expected to particularly foster reach of consensus between managers and other employees, engineers themselves and between supporters and engineers
- Crash report data supporting identification of software failure causes, customers experiencing the failures, and software releases used by customers over time, form the basis of such an SOS.

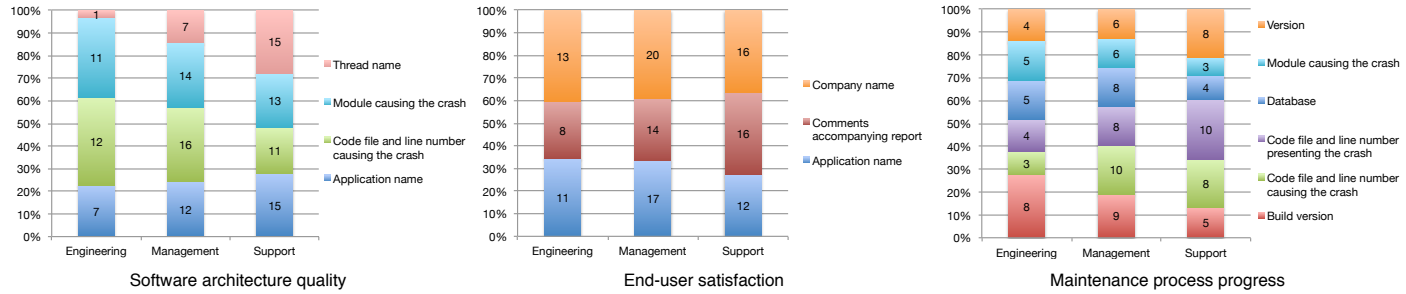

Fig. 6. Most-selected crash report data types per maintenance focus per employee role. The data types are expected to contribute the most to successful determination of which bugs will result in the largest increase of software architecture quality, end-user satisfaction and maintenance process progress, once fixed (question 16). The data types form the basis of an SOS that fosters consensus on prioritization of software maintenance tasks

VI. Sending Out an SOS: Case Study Results

To further establish the soundness and validity of the SOS concept, in addition to survey results we conducted a case study [17] of one month at CADComp, during which we composed an SOS based on crash report data. Having attended development meetings and plenary company meetings, we observed lack of consensus between (product) management, and both development and support employees on prioritization of software engineering work items: based on received crash reports, the former employees believed that in-the-field product quality was increasing and implementation of new features should have first priority, while both latter employee groups contrarily believed that product quality was decreasing and bug fixing should have first priority.

Based on our observations at CADComp, we created an SOS in the form of an ASP.NET MVC web application that was deployed at the vendor's intranet. Derived from CADComp crash report data, the SOS presents software operation information in the form of graphs that visualize recent software operation history. Figure 7 shows one of the graphs that visualizes recent software quality: crash reports sent by various in-the-field releases of the CADComp software are distributed over the code file names (partly hidden for confidentiality reasons) and line numbers that caused the particular crash.

Apart from which code is causing in-the-field software failures in which version of the software, this SOS graph provides knowledge about crash report acquisition service outages (A), how in-the-field software quality improves over time (B), which code forms structural weaknesses in the software (C), when the software is barely used by end-users (D) and when new major bugs are introduced (E). Furthermore, the SOS provides operation meta data such as operation environment statistics and customer names corresponding to the crash.

The soundness and validity of the SOS in the context of CADComp's work item prioritization were evaluated through seven unstructured interview sessions [17] with three product managers, two senior software engineers and two customer supporters. Although the product managers initially asked for help with interpreting the SOS graph depicted in figure 7 (they assumed the graph illustrated a linear increase of crash report submissions), all seven interviewees indicated that the graph increased awareness of in-the-field software operation

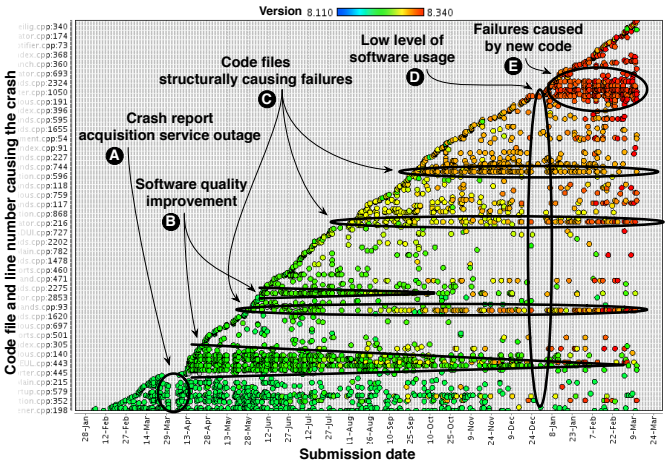

Fig. 7. A software quality graph of the SOS implemented at CADComp, showing crashes per 'code file and line number causing the crash' over time

throughout the organization, and expected the SOS application to contribute to the reach of consensus on work item prioritization (new features versus maintenance tasks). Also, software engineers suggested to only include crash reports in the SOS graph that form structural problems (since those problems cause the most crashes, and result in the highest increase in software architecture quality, once solved). Finally, customer supporters expected that on the long term, frequent usage of the software operation summary would result in an increase of customer satisfaction.

Based on results of both the CADComp case study as well as the maintenance task prioritization survey, the six propositions defined in section III are evaluated. Survey results show that most employees spend 1 to 5 hours weekly on software maintenance task prioritization, during which lack of consensus is frequently experienced, particularly with employees from engineering (monthly on average). All employee roles could well use a software operation summary about every three weeks, particularly during in preparation of (sprint) planning, (software) development, and bug fixing activities. Therefore, we consider proposition **P1** (*A software operation summary is expected to integrate with current software maintenance practices*) to be correct. The next three propositions (*A software operation summary is expected to increase knowledge of software architecture quality* (**P2**), *end-user satisfaction* (**P3**), *maintenance process efficiency* (**P4**)) are also confirmed

by survey results: employees from engineering, management and support mostly consider a software operation summary to increase knowledge of software architecture quality, end-user satisfaction and maintenance process efficiency as probable. Also, most respondents (28%) consider an SOS fostering the reach consensus with colleagues in terms of software maintenance task prioritization as probable. Proposition **P5** (*A software operation summary is expected to foster achieving consensus on software maintenance task prioritization*) is therefore considered to be correct. Proposition **P6** (*A software operation summary is expected to reduce the time needed for software maintenance task prioritization*) appears to be relatively difficult to evaluate: while on average, saving time on prioritization of software maintenance tasks through SOS usage is considered 'possible' by nearly all respondents, further research is needed to demonstrate reduction of the time needed for software maintenance task prioritization through SOS usage. We therefore consider the correctness of this proposition to be undetermined until further research on this subject has been performed. Finally, case study results support propositions **P1**, **P2**, **P3** and **P5**.

VII. Threats to Validity

The validity of the study results is threatened by several factors. First, survey-related aspects (as described by Kitchenham and Pfleeger [13]) threaten construct validity of the research. For example, although 136 respondents initiated the survey, 79 (58.1%) completed it. This may be grounded in the type, order and number of survey questions. Also, as a consequence of the fact that the survey was anonymous, we can not exactly determine the number of distinct software-producing organizations that have responded to the survey, as well as the distribution of employee roles per organization. Furthermore, while concepts (such as the SOS) were introduced to respondents before questions related to these concepts were asked, it might be that common understanding among respondents was not fully reached. Although statistical relevance of (differences in) survey results may be limited by these factors, we believe the results are indicative and representative for the product software industry in the Netherlands.

Second, internal validity of the study may be threatened by the fact that in this study, we focus on corrective and adaptive software maintenance tasks, while preventive and perfective tasks are considered outside the scope of the study considering the types of data generally provided by crash reports. Conclusions based on the survey results might appear to be biased to certain extent. We acknowledge, however, that different types of operation information can be used to meet different requirements in optimizing maintenance activities and processes (as expressed in section VI).

Third, external validity of our research may be threatened by two factors. First, survey results could be dominated by respondents of a small number of software vendors. Second, in this study we focus on corrective and adaptive maintenance tasks. Both factors may influence the extent to which the results are applicable to other (types of) vendors: survey results might to certain extent be typical for software vendors in the Netherlands, and therefore are limitedly generalizable to vendors outside these categories. Although we believe that the results of this study are representative for many vendors, further research is needed to mitigate these threats.

VIII. Related Work

Many research efforts cover the use of information fragments to improve people's practices, processes and workflows. However, to improve those through software operation knowledge is only considered by few. Kim et al. [18] propose a machine learning approach for predicting top software failures to prioritize maintenance efforts. Although their approach appears to be promising (75%–90% accuracy), it only considers the frequency of a crash as a prioritization factor: it does not consider software operation at particular customers, or involve the efficiency of the maintenance process, for example.

To determine the contents and investigate the quality of *bug* reports (as opposed to *crash* reports, which are researched in this paper), Zimmerman et al. conducted a survey among developers of Apache, Eclipse and Mozilla [19]. Their conclusions (e.g. well-written, complete reports are likely to get more attention than poorly written or incomplete ones) may also apply to the SOS concept. However, their study only involves software engineers and does not consider management or customer support employees.

As an attempt to answer questions asked by software developers during their daily work, Fritz and Murphy [20] have introduced an information fragment model that supports composition of information from multiple sources (among others, bug reports) and supports the presentation of composed information in various ways. The authors show that the model can support 78 questions developers want to ask about a development project. While we recognize the value of combining information from multiple sources for answering diverse questions asked by one type of people, in our study we use one source of information (*crash reports*) to answer one question (*how should software maintenance tasks be prioritized?*) asked by multiple types of employees (*engineering*, *management* and *customer support*). Involving and combining multiple sources of information could lead to lengthy discussions regarding reason of information involvement as well as weight of the information in the resulting combination, which could delay the reach of consensus between different employee roles.

IX. Conclusions and Future Work

Software-producing organizations increasingly recognize the relevance and potential of software operation information visualizations on operation dashboards, reports and other media. However, all too often in industry, vendors experience difficulties in selecting and presenting operation information in such a way that besides providing software operation knowledge, visualized software operation information actually contributes to improvement of software process execution. Software maintenance task prioritization, for instance, is often time-consuming because reaching consensus between involved

employees regarding this prioritization is tedious: involved employees have different roles, each with corresponding concerns, maintenance foci and improvement aspects.

In this paper, we propose the concept of software operation summary (SOS): an overview of recent in-the-field software operation based on acquired software operation information, which can be used to improve execution of software processes. For example, improving the process of software maintenance task prioritization by fostering the reach of consensus between employees on such prioritization.

We report on an extensive software maintenance task prioritization survey, held among 136 product / development managers, software developers and customer supporters. Survey results confirm the demand for an SOS that contributes to reach of consensus on software maintenance task prioritization, particularly in preparation of (sprint) planning, (software) development and bug fixing activities. As a basis for such an SOS, particularly crash report data supporting identification of (1) software failure causes, (2) customers experiencing the failures, and (3) software releases used by customers over time, are considered relevant. Furthermore, we report on a case study during which we compose an SOS based on crash report data, and empirically evaluate it at a European software vendor. Case study results show that an SOS increases awareness of in-the-field software operation throughout software-producing organizations, and indicate that it contributes to the reach of consensus on work item prioritization (e.g., new features versus maintenance tasks). Based on survey and case study results (through which the six propositions, defined to establish the soundness and validity of the SOS concept, are evaluated), we consider the main research question of this paper (*Can prioritization of software maintenance tasks be improved through the concept of a software operation summary?*) to be answered positively.

While this paper focuses on the use of a software operation summary in the context of software maintenance task prioritization, an SOS can be used to support numerous practices and processes. Using the SOK integration template method [12], software vendors can tailor the form and contents of a software operation summary to their needs and requirements, and therewith optimize integration and presentation of software operation information. Vendors should ensure that recent operation information is actually available at the time and with the frequency an SOS is used. Also, selected underlying data types should correspond with the needs of the involved employees. Vendors should be aware, however, that too many data types underlying their SOS may limit the extent to which reach of consensus between their employees is fostered.

Future research activities include extension and concretization of the software operation summary concept: concrete SOSes that are tailored to processes other than software maintenance, will be developed and empirically validated, to further show its viability in terms of measurable process efficiency increases. Finally, it will be researched which visualization techniques are most appropriate for presenting (operation) data on software operation summaries and other media.

ACKNOWLEDGMENTS

We would like to thank all participating software vendors and their employees for sharing their ideas and experiences. In particular, we thank Matthijs Steen for his suggestions.

REFERENCES

[1] L. Lehtola and M. Kauppinen, "Suitability of Requirements Prioritization Methods for Market-driven Software Product Development," *Softw. Process: Improvement and Practice*, vol. 11, no. 1, pp. 7–19, 2006.

[2] H. van der Schuur, S. Jansen, and S. Brinkkemper, "A Reference Framework for Utilization of Software Operation Knowledge," in *SEAA'10: Proceedings of the 36th EUROMICRO Conference on Software Engineering and Advanced Applications*. IEEE Computer Society, 2010, pp. 245–254.

[3] International Organization for Standardization, "ISO/IEC 14764:2006: Software Engineering — Software Life Cycle Processes — Maintenance," 2006.

[4] T. M. Pigoski, *Practical Software Maintenance: Best Practices for Managing Your Software Investment*. John Wiley & Sons, Inc., 1997.

[5] H. van der Schuur, S. Jansen, and S. Brinkkemper, "Reducing Maintenance Effort through Software Operation Knowledge: An Eclectic Empirical Evaluation," in *CSMR'11: Proceedings of the 15th European Conference on Software Maintenance and Reengineering*. IEEE Computer Society, 2011, pp. 201–210.

[6] "Microsoft Online Crash Analysis: Error Report Contents Information," 2005, http://oca.microsoft.com/en/dcp20.asp, verified 20/06/2011.

[7] "How to: Configure Microsoft Error Reporting," 2006, http://msdn.microsoft.com/en-us/library/bb219076(v=office.12).aspx, verified 20/06/2011.

[8] "Mac OS X Reference Library – TN2123: CrashReporter," 2010, http://developer.apple.com/library/mac/technotes/tn2004/tn2123.html, verified 20/06/2011.

[9] "Ubuntu Wiki – Apport," 2010, https://wiki.ubuntu.com/Apport, verified 20/06/2011.

[10] "google-breakpad – Crash reporting," 2010, http://code.google.com/p/google-breakpad/, verified 20/06/2011.

[11] K. Glerum, K. Kinshumann, S. Greenberg, G. Aul, V. Orgovan, G. Nichols, D. Grant, G. Loihle, and G. C. Hunt, "Debugging in the (Very) Large: Ten Years of Implementation and Experience," in *SOSP '09: Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles*. ACM, 2009, pp. 103–116.

[12] H. van der Schuur, S. Jansen and S. Brinkkemper, "If the SOK Fits, Wear It: Pragmatic Process Improvement through Software Operation Knowledge," in *PROFES'11: Proceedings of the 12th International Conference on Product Focused Software Development and Process Improvement*. Springer, 2011.

[13] B. A. Kitchenham and S. L. Pfleeger, "Principles of Survey Research Part 3: Constructing a Survey Instrument," vol. 27, pp. 20–24, 2002.

[14] I. van de Weerd and S. Brinkkemper, "Meta-Modeling for Situational Analysis and Design Methods," in *Handbook of Research on Modern Systems Analysis and Design Technologies and Applications*. Information Science Reference, 2008, pp. 38–58.

[15] "Universiteit onderzoekt prioriteren onopgeloste bugs (*University researches prioritization of unsolved bugs*)," 2011, http://www.automatiseringgids.nl/technologie/software/2011/4/onderzoek-naar-softwareonderhoud.aspx, verified 20/06/2011.

[16] *The new SME definition*, ser. Enterprise and Industry Publications. Office for Official Publications of the European Communities, 2005.

[17] R. K. Yin, *Case Study Research: Design and Methods (Applied Social Research Methods)*, fourth edition. ed. Sage Publications, 2009.

[18] D. Kim, X. Wang, S. Kim, A. Zeller, S. Cheung, and S. Park, "Which Crashes Should I Fix First?: Predicting Top Crashes at an Early Stage to Prioritize Debugging Efforts," *IEEE Transactions on Software Engineering*, 2011.

[19] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schrter, and C. Weiss, "What Makes a Good Bug Report?" *IEEE Transactions on Software Engineering*, vol. 36, no. 5, pp. 618–643, 2010.

[20] T. Fritz and G. C. Murphy, "Using Information Fragments to Answer the Questions Developers Ask," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering — Volume 1*, ser. ICSE'10. ACM, 2010, pp. 175–184.