If the SOK Fits, Wear It: Pragmatic Process Improvement through Software Operation Knowledge

Henk van der Schuur, Slinger Jansen, and 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 nowadays acquired by many software-producing organizations. Vendors are effective in acquiring large amounts of valuable software operation data to improve the quality of their software products. For many vendors, however, it remains unclear how their actual product software processes can be advanced through structural integration of such information. In this paper, we present a template method for integration of software operation information with product software processes, and present four lessons learned that are identified based on a canonical action research study of ten months, during which the method was instantiated at a European software vendor. Results show that the template method contributes to significant software quality increase, by pragmatic but measurable improvement of software processes, without adhering to strict requirements from cumbersome maturity models or process improvement frameworks.

1 Introduction

Nowadays, software vendors are continuously striving for refinement and improvement of their software processes to achieve and extend competitive advantage. Simultaneously, software vendors are experienced in acquiring in-the-field operation data from their software products and services. It has become common practice, for example, to monitor software operation and identify operation failures by means of acquired operation information [5, 12].

A wealth of software operation knowledge (SOK), gained from operation information analysis, can improve basically any software process in a product software company, such as software maintenance, software product management and customer support [11]. For many vendors, however, it remains unclear how software processes can be improved through integration of such information. As a consequence, acquired operation information is left untouched during execution of product software processes.

The research question we attempt to answer in this paper is '*How can product software processes effectively be improved with acquired information of in-the-field software operation?*'. To answer this question, we introduce a template

method for integration of software operation information with product software processes, and present four lessons learned that are identified based on a canonical action research study of ten months, during which the method was instantiated at a European software vendor. Several prescriptive 'one-size-fits-all' software process improvement (SPI) approaches, such as the Capability Maturity Model Integration (CMMI) and ISO/IEC 15504 (SPICE), are considered as too large, too extensive and too expensive to comprehend and implement effectively within small and medium-sized companies [1, 8, 10, 13]. As opposed to those approaches, the SOK integration template method presented is pragmatic and inductive, i.e., potential resulting improvements are based on the particular situation of an organization. The template method particularly describes *what* are typical operation information integration activities and concepts; a method instantiation describes *how* these activities should take place and how related concepts are involved, specific to a vendor's situation.

This paper continues with placing our work in context. Next, the research approach is detailed, after which the SOK integration template method is presented (section 4). Section 5 details in-the-field instantiation of the method, as well as an valuation of integration experiences, resulting lessons learned and research limitations. Finally, conclusions and future work are presented (section 6).

2 Related Work

Many research efforts cover the subject of software process improvement, but only few consider knowledge of in-the-field software operation as an instrument for improving product software processes.

For instance, Pettersson et al. [10] have proposed iFLAP, a process improvement framework that is to some extent similar to the SOK integration template method we presented: the framework is inductive in nature and draws on knowledge that is already residing in the organization to improve processes. However, as opposed to our method, the framework is specifically directed towards, and evaluated with, requirements engineering processes. Miler and Górski [9] report on a case study in which they apply a risk-driven software process improvement framework in a real-life software project. Case study results demonstrate that the proposed framework is able to reveal new, previously undetected risks that provide important input for process improvement. Although various undetected risks were identified, the framework requires a high level of process description detail to be applied effectively. Iversen et al. [7] proposed a framework for understanding risk areas and resolution strategies within software process improvement, as well as a corresponding risk management process. Although both the framework and the process are comprehensive and detailed, the framework was not evaluated through empirical studies and practical use. Moreover, it was left unclear how and to which extent vendors over time actually benefit from it. Also, many efforts are directed to demonstrating the effectiveness of software process improvement by means of CMM(I) [2, 4]. For example, Dangle et al. [4] analyze the role of process improvement in the context of small organizations

through an extensive case study in which CMM is applied. Based on this study, lessons learned are identified. Although one lesson is somewhat analogous to one of the lessons we have identified, it is left unclear to which extent the lessons learned of Dangle et al. are generalizable to vendors that have implemented a different maturity model, or no maturity model at all.

In this paper, we demonstrate pragmatic but measurable improvement of product software processes, and identify lessons learned that are generalizable to similar product software vendors.

3 Research Approach

To investigate how product software processes can be effectively improved with acquired information of in-the-field software operation, we designed a template method for integration of such information with product software processes, following a combination of design research and canonical action research principles [6, 3]. We conducted an extensive action research study at a European software vendor, during which the method was used to integrate software operation information acquired by the vendor with the vendor's product software processes, and therewith improve those processes. Based on study results, lessons learned are identified that may serve as a guiding substrate for similar vendors in integrating operation information with their product software processes.

3.1 Research Site

The action research study presented in this paper was carried out at CADComp[1], a European software vendor founded in 1990. The vendor develops an industrial design application, CADProd, which is targeted on the Microsoft Windows platform and is used daily by more than 4,000 customers in five countries. Since the start of its development in 1995, four major versions of the application have been released. Currently, CADComp employs about 100 people and is established in the Netherlands, Belgium and Romania. From December, 2009 to September, 2010, we were present at the vendor's main development site to integrate acquired SOK in the vendor's product software processes.

Before our study commenced, CADComp already acquired data of the in-the-field operation of its software product CADProd in the form of customized error reports. However, these data were not structurally analyzed, no software operation information was extracted from these data and no operation information was integrated with CADComp's existing product software processes.

3.2 Canonical Action Research

The canonical action research method described by Davison et al. [3] was used to structure the research activities. We attempted to satisfy each of their 'canonical action research principles':

[1] Note that for reasons of confidentiality, the names of the vendor and its software products have been anonymized.

1. Principle of the Researcher-Client Agreement The study was conducted at the European software vendor CADComp. Both the researchers and the vendor agreed on the action research approach; the vendor indicated that it is in need of an approach for improving its product software processes with the use of acquired operation data. We have agreed on a research plan that contains an overview of the shared research objectives as well as the data that was to be collected during the study (e.g. software architecture specifications, process descriptions, memos, semi-structured interviews, etc.)
2. Principle of the Cyclical Process Model The cyclical process model [3], originally proposed by Susman and Evered [14], served as a basis for our work at the vendor. Structuring research activities by means of this model ensures adequate action planning, action taking and evaluation, as well as specifying what other vendors and researchers could learn from our study.
3. Principle of Theory According to Davison et al. [3], action research theory takes the following form: in situation S with salient features $F_1, \ldots, F_n$, outcomes $O_1, \ldots, O_n$ are expected from actions $A_1, \ldots, A_n$. Our (grounded) theory, as reflected by the formulated research question, is consistent with this form: we expect vendors that acquire but not analyze or integrate operation information (S), to improve their product software processes (O) by means of implementing an instantiation of the SOK integration template method (A).
4. Principle of Change through Action As stated by principle 1 and 3, both the researcher and the vendor were motivated to change the situation at the vendor in terms of operation data and information use. Planned actions were designed and taken to achieve the defined objectives (see section 5).
5. Principle of Learning through Reflection Both the observations made as well as the actions taken were reported to, and evaluated with the vendor's employees and management. The researcher and vendor reflected outcomes of the study by means of semi-structured interview sessions (see section 5).

Adhering to these principles assisted us in establishing pure validity of our research approach, which contributes to realistic repetition of the study at similar software vendor sites.

4 SOK Integration

We have developed a template method to facilitate integration of acquired SOK into existing product software processes (*target processes*). The method is designed to guide vendors in (1) identification of relevant and valuable operation information, (2) analysis of target processes and their integration environment, (3) integration of selected information in, and transformation of, target processes and (4) presentation of integrated operation information.

When applied successfully, the method enables software vendors to increase the extent to which their practices, processes and tools are supported by SOK, which may result in directed software engineering, informed management and more intimate customer relationships. Figure 3 depicts the method as a Process-Deliverable Diagram (PDD) [15].

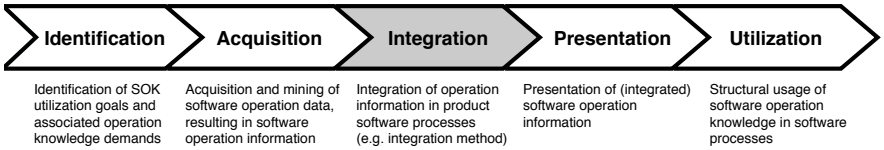

Fig. 1. Integration of operation information is part of the SOK life cycle [11]

In the context of the SOK framework [11], the SOK integration template method is an implementation of the SOK integration process, following identification [11] and acquisition [12] processes, and preceding presentation and utilization processes (see figure 1). In the SOK acquisition process, software operation data are acquired, mined and analyzed, resulting in software operation information that forms input of the method. After acquisition, operation information is presented on carriers determined during application of the method; software operation knowledge resulting from interpretation of such information is used with the frequency determined during method application.

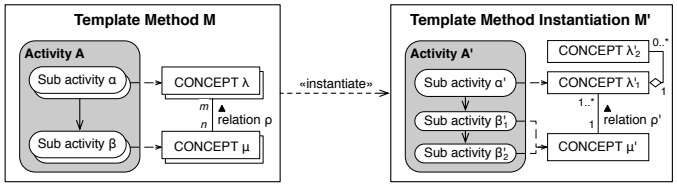

Fig. 2. Template method instantiation

4.1 Instantiation

The SOK integration template method can be used by software vendors that attempt to integrate operation information with product software processes. For each target process, a new method instance is created, each with corresponding object and activity instantiations. The template method prescribes *what* (rather than *how*) activities and concepts are to be implemented by software vendors. Therefore, the method is only composed of *open* activities and concepts [15], which should be instantiated with standard, open or closed equivalents. To anchor process improvement in the organization, vendors should instantiate both template activities and concepts consciously and diligently, taking into account daily practices. The method can be typically applied iteratively, continuously, and potentially in parallel with other method instantiations. The method's application duration and frequency depend on INTEGRATION RESOURCEs and environment. One or multiple people are responsible for successful execution of (sub) activities. We assume that both the SOFTWARE OPERATION INFORMATION as well as one or more TARGET PROCESSes are available and accessible, before instantiating the method. An instantiation example is visualized in figure 2.

4.2 Concepts

The nine template concepts the SOK integration template method refers to, depicted at the right side of the PDD (see figure 3), are detailed in table 1.

Concept Name	Concept Description
ACTOR	A human who demands and utilizes SOFTWARE OPERATION INFORMATION with a certain frequency, potentially visualized by a CARRIER, and participates in one or more TARGET PROCESSes
CARRIER	Medium that can convey and visualize SOFTWARE OPERATION INFORMATION
INTEGRATION RESOURCE	The (human) resources available for performing integration of SOFTWARE OPERATION INFORMATION
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 a TARGET PROCESS involving one or more INTEGRATION REQUIREMENTs
INTEGRATION REQUIREMENT	A SOFTWARE OPERATION INFORMATION integration necessity, demanding an amount of INTEGRATION RESOURCEs
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 CARRIER
TARGET PROCESS	A collection of related, structured activities, tasks, tools and ideas that produce a specific service or product for (a) customer(s), possibly dependent on other processes or activities, with which SOFTWARE OPERATION INFORMATION is integrated

Table 1. Concepts of the SOK integration template method

Each of the concepts referred to by the method, corresponds to at least one of the (sub) activities of which the template method is composed, which are detailed in the next section.

4.3 Activities

The method's activities and sub activities, depicted at the left side of the PDD (see figure 3) are detailed below.

1. Operation information selection In the first activity of the method, a selection of relevant operation information resulting from data mining and abstraction of software operation data is made. First, the TARGET PROCESS is analyzed (*Analyze target process*). Questions like 'How does the process actually work?', 'How is the process used?', and 'What are process dependencies?' are answered during this activity. *Analyze target process* may be performed from a specific perspective (e.g. human interaction, data dependencies, etc.), which results in a comprehensive view of the process. Based on this process analysis, INTEGRATION OBJECTIVES are determined (*Determine integration objectives*). In this activity, integration incentives and goals are identified, for example by defining the role and functioning of the TARGET PROCESS after integration. Secondly, software operation information demands of the vendor are identified (*Identify operation information demands*). Next, operation information that is considered relevant and valuable to integrate with the TARGET PROCESS, is selected (*Select relevant operation information*)[2], based on the process analysis results and identified operation information demands.

2. Integration requirements identification First, current and future actors involved with the TARGET PROCESS and acquisition, integration or presentation of operation information are identified (*Identify SOK actors*). Secondly, it is estimated how often SOK resulting from integration or presentation of operation information will be used by ACTORs within the TARGET PROCESS after integration (*Estimate SOK utilization frequency*). Thirdly, CARRIERs for presentation

[2] If operation information is missing, application of the method can be interrupted to first iterate through the identification and acquisition phases again (see figure 1), and therewith make sure that desired information is available and accessible.

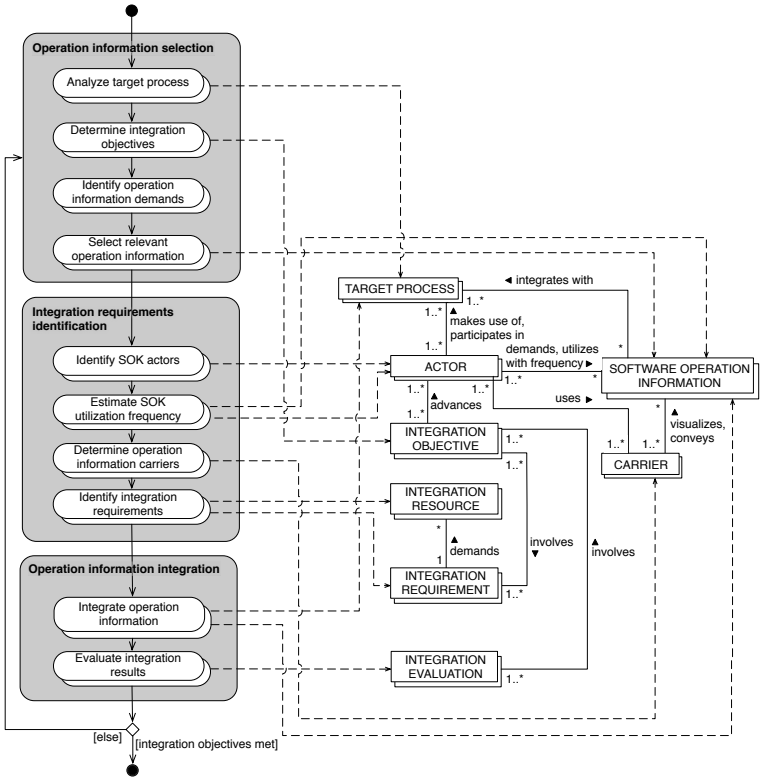

Fig. 3. SOK integration template method

of operation information integrated with the TARGET PROCESS are determined (*Determine operation information carriers*). Finally, based on the sub activities prior to *Identify integration requirements*, INTEGRATION RESOURCEs and INTEGRATION REQUIREMENTs for effective integration of acquired SOFTWARE OPERATION INFORMATION are determined. Resulting requirements serve as input for the subsequent 'Operation information integration' activity.

3. Operation information integration The TARGET PROCESS is altered to allow integration of relevant SOFTWARE OPERATION INFORMATION (*Integrate software operation information*) selected in 'Operation information selection'. Integration of selected operation information is dependent on, and constrained by, the available INTEGRATION RESOURCEs identified earlier (e.g. external data sources, time, people, knowledge, etc.) Operation information integration results are evaluated in the second activity (*Evaluate integration results*). If, based on the subsequent INTEGRATION EVALUATION, can be concluded that INTEGRATION OBJECTIVEs are met, the result of this activity (and therewith of the SOK integration template method) is a process that is effectively supported by acquired operation information. Otherwise, the method is reinitiated by starting the 'Operation information selection' activity.

In the following section, we demonstrate instantiation of the template method.

5 Three Pragmatic In-the-field Method Instantiations

CADComp uses its own software operation data mining and analysis tool called Denerr, to extract operation information from the acquired CADProd operation data. Denerr was deployed on CADComp's intranet, and is accessible to all CADComp employees. The tool presents acquired error reports, and provides comprehensive data filtering functionality allowing developers and maintainers to define constraints on each of the error report properties, for example to analyze a particular set of error reports. The SOK integration template method was used to integrate operation information extracted by the Denerr tool in three of CADComp's product software processes: (1) software maintenance, (2) software product management and (3) customer support. Each process was selected based on CADComps needs expressed by its CEO, and corresponds to a particular SOK framework perspective [11].

Tables 2 and 3 respectively list how the template activities and concepts of the integration method were instantiated for each process. As part of the integration process, adjustments were made to the Denerr tool and to CADComp product software processes. CADComp employees involved in the target processes were introduced to new Denerr functionality by means of e-mails describing the new functionality, meetings of both software development and customer support departments during which new functionality was presented, and short hands-on evaluation sessions during which new functionality was demonstrated to and evaluated by particular employees. The SOK integration template method activities were performed in parallel. All integration activities were performed by the researchers and were overseen by the CADComp CEO.

5.1 Observations

Six months after initiation of our study, weekly one-hour 'Denerr harvest meetings' were introduced to frequently analyze and delegate received error reports. The meetings were organized with two maintenance team leaders and one researcher, and were led by the international product manager who was responsible for all Denerr-related issues. We attended the first three meetings in preparation of the interviews. With around 8200 error reports received since the start of error report acquisition seven months earlier, aims of the first meeting were to (1) investigate which error reports could be considered old or irrelevant and put their status to 'Ignore', and therewith get a recent, realistic view of the status of all error reports, (2) delegate each aggregated report in the resulting top 10 of aggregated error reports to a software development team, and (3) investigate reports received in the last week to identify potential bugs introduced recently.

During the second and third meeting, respectively, the status of tasks identified during first harvest were discussed again with the team leaders, and reports with automatically assigned statuses (see table 3) were verified to check if the correct status was assigned. Based on our attendance of the meetings, three main observations were made. First, during analysis of the error reports, employees were surprised about the amount of reports being submitted, particularly from

Template activity	**Target process** (corresponding perspective)					
	Software maintenance (development)		**Software product management** (company)		**Customer support** (customer)	
	Time period	Activity instantiation	Time period	Activity instantiation	Time period	Activity instantiation
Analyze target process	12/09–01/10	Software maintenance is dependent on testing, customer support, customer training and sales feedback: error reports are just acquired and stored, progress regarding repairing software failures during software maintenance is not registered. CADProd developers are not used to involving end-user error reports in their maintenance work	01/10	The software product management process relies on operation information stored in customer support (CRM) software, as well as end-user feature requests and bug fix wishes reported by salesmen in planning features and bug fixes for upcoming releases. Error reports are not taken into account by product management in the context of these activities	01/10	Customer support is used to answer end-user questions, aid customers with software usage, identify software failures and sustain customer contact. Customer supporters have limited knowledge of the in-the-field operation of their software at customers, and are dependent on the customer's willingness to explain its situation and reason for calling
Determine integration objectives		Integration objectives (e.g. accelerate software maintenance) based on analysis of the imperfections of the current software maintenance process (e.g. process dependencies slowing down the maintenance process)		Integration objectives (e.g. direct software product management) were based on analysis of the imperfections of the current software product management process (e.g. limited use of operation information)		Integration objectives (e.g. increase customer intimacy) were based on analysis of the imperfections of the current customer support process (e.g. little a priori knowledge of customer's reason for calling)
Identify operation information demands		No significant demand elicitation was performed: software engineers requested for maintenance status information of error reports ad hoc, if at all		During evaluation of the Denerr tool, product managers and CEO requested for aggregated operation information and software operation trends		Through short talks with customer supporters, operation information supporting identification of particular customers was identified as information requested for
Select relevant operation information		Error report meta information was identified as relevant information, based on information demands of software maintainers and analysis of the current software maintenance process		Aggregated error report properties and software operation trends were identified as relevant operation information based on information demands of product managers and analysis of the current software product management process		Operation information which enables identification of customers was identified as relevant operation information based on information demands of customer supporters and analysis of the current customer support process
Identify SOK actors	01/10	Actors were selected from the employees that requested for operation information or were appointed by the CEO	02/10	Actors were selected from the employees that requested for operation information or were appointed by the CEO	01/10–02/10	Actors were selected from the employees that requested for operation information or were appointed by the CEO
Estimate SOK utilization frequency		By analyzing the frequency of current software maintenance process activities, it was estimated that SOK would be used at least once a week		By analyzing the frequency of current software product management process activities, it was estimated that SOK would be used at least every Scrum sprint		By analyzing the frequency of current customer support process activities, it was estimated that SOK would be used potentially with every customer support call
Determine operation information carriers		Based on (meta) operation information demands and feedback from CADComp employees, an error report list view with status information was added to the Denerr tool		Based on (meta) operation information demands and feedback from CADComp employees, e.g. an aggregated error report view was added to the Denerr tool		Based on (meta) operation information demands and feedback from CADComp employees, customer-specific error report views where added to the Denerr tool
Identify integration requirements		Creating and introducing error report labeling functionality were identified as integration requirements, based on results of previous activities as well as integration resources appointed by CADComp's CEO		Creating and introducing error report aggregation functionality were identified as integration requirements, based on results of previous activities as well as integration resources appointed by CADComp's CEO		Extending the error report format and Denerr functionality with customer-specific elements were identified as integration requirements, based on results of previous activities as well as integration resources appointed by CADComp's CEO
Integrate operation information	02-10–03/10	Denerr was extended to show error report status, assign responsible people and expand team leader's responsibilities	03/10–09/10	Various graphs have been created (e.g., error cause modules, error report submission time, end-user countries, etc.), as well as an aggregation view containing the most frequent occurring errors	02/10–06/10	IP-based location determination was implemented by means of IP geolocation library, customer-specific error report analysis view was created, integration with external tools was realized
Evaluate integration results		After deployment of new Denerr functionality, short evaluation talks were held with the involved software engineers		After deployment of new Denerr functionality, short evaluation talks were held with the involved product managers		After deployment of new Denerr functionality, short evaluation talks were held with the involved customer supporters

Table 2. Template activity instantiations

Template concept	Target process (corresponding perspective)		
	Software maintenance (development)	**Software product management** (company)	**Customer support** (customer)
ACTOR	2 software engineers, 4 team leaders, 3 product managers	3 product managers, CEO	3 customer supporters, 2 software engineers
CARRIER	Error report list view with status information	Statistics, graphs, aggregation error reports view	Customer-specific error report view, acquisition, mining and analysis of subjective operation information (e.g. end-user comments)
INTEGRATION RESOURCE	1 software engineer, 3 team leaders, 1 product manager	1 software engineer, 3 team leaders, 1 product manager	1 software engineer, 3 team leaders, 1 product manager
INTEGRATION EVALUATION	Employees suggested that error reports could (and should) be automatically be assigned a status, particularly when at least 95% of the equivalent error reports is assigned one and the same status, to streamline maintenance work and reduce repetitive administration tasks. Also, it was suggested to tighten integration of acquired operation knowledge with the software maintenance process by keeping track of error reports status change(r)s over time, to analyze past and delegate future maintenance tasks. Both ideas were implemented. Finally, it was suggested to integrate error report data with bug tracking software, to align error reports with identified bugs.	Initially, the graph- and aggregation views were always generated for all error reports. Later, it appeared that more specific queries were requested for generating these views, resulting in views generated for error reports with a particular build number, status or IP address. Therefore, both views were implemented as a search result view. Also, it appeared convenient to delegate errors to teams using 'error report bundle' URLs instead of URLs to individual reports: teams are provided with insight in the scope and 'in-the-field severity' of software failures. 'Bundle URLs' were created to support this form of communication.	A basic customer view was first accessible via the detail view of each individual error report. Supporters made very little use of this customer view. A global customer search was implemented to alleviate the task of associating customer support details with operation information and tighten integration of operation information in the customer support process. Also, it was suggested to extend CADComp's Salesforce 'customer prepsheet' with operation information of the particular customer, to provide salesmen and supporters with insight in the customer's recent in-the-field software operation
INTEGRATION OBJECTIVE	Increase error report status awareness; enable error report status updating and monitoring; informed software maintenance	Gain insight in trending error report characteristics; directed software product (release) management	Gain insight in software operation at particular customers; increase customer intimacy
INTEGRATION REQUIREMENT	Labeling error reports upon receive with 'New' status, visualize labels within Denerr tool, implement label update functionality	Adding aggregated error report view and trending graphs to Denerr tool	Extending error report format and Denerr mining, analysis and reporting functionality (include customer profile and comment data)
SOFTWARE OPERATION INFORMATION	Error report status labels	Aggregated operation information, software operation trends	Operation information which enables identification of customers, such as IP address, customer and user name, license number, etc.
TARGET PROCESS	Software maintenance	Software product management	Customer support

Table 3. Template concept instantiations

versions that were considered old versions by the employees. As a consequence, questions like '*When can we ignore error reports originating from a particular release build?*' and '*To which extent is it desirable to see the number of error reports received from a new software product significantly increase month over month?*' were discussed. Secondly, although employees understood the significance of the reports ('*Those error reports represent the unhandled exceptions that are experienced by our end-users*)', occasionally, insufficient data was available to gain a clear understanding of an end-user's software operation. As a consequence, end-user comments accompanying the error reports were frequently analyzed to identify the usage history and goals of end-users. Also, team leaders formulated more accurate operation information demands. Thirdly, we observed a demand for fine-grained report analysis. After aggregation and statistic views were implemented for all error reports, employees desired to show these views only for reports originating from release builds, internal builds and per (build) version. The aggregation view was used to quickly identify which bugs were not under investigation for the current sprint. Bug fix work items were created, and engineers were managed based on the aggregation view.

5.2 Experience evaluation

Twelve semi-structured interviews consisting of 34 questions divided over seven sections[3] (*Process Improvement*, *Integration Objectives*, *Integration Challenges*, *Return On Investment*, *Lessons Learned*, *Future* and *Final Remarks*), were performed after application of the SOK integration template method. Interviews

[3] Interview questions can be found at http://people.cs.uu.nl/schuurhw/sokintegration

were conducted with CADComp employees that are involved in a particular product software process (see table 4), to reflect on the SOK integration process and identify lessons learned. The method, tables 2 and 3 as well as observations of the attended Denerr harvest meetings served as input for the interviews. The interviews took 1.5 hour on average and were conducted over a period of 68 days. Interview results of the first four sections are summarized in table 5 (lessons learned are presented separately in section 5.3).

Interviewee type	Software maintenance	Software product management	Customer support	Years experience in IT (average)
Senior supporters			3	16.3
Senior software engineers	2			12
Team leaders	3			8.5
Product managers		3		16.5
CEO		1		25

Table 4. Interviewees per target process

Area	Software maintenance	Software product management	Customer support
Integration objectives	Quickly identify software failures most customers are experiencing frequently, faster improve software quality and performance, increase customer satisfaction	Gain more precise insight in software failures and weak spots in the software code base, efficiently increase software quality	Increase customer intimacy, increase efficiency of product software processes, get insight in software usage of customers that do not call for support
Process improvements	Software maintenance (e.g. bug prioritization): time was saved because software failures are faster reproducible, better insight in and awareness of in-the-field software operation and quality was gained	Software maintenance, software product management: processes have been accelerated because software failures are bundeled (clustered) and prioritized automatically and discussed on a weekly basis (instead of manual, subjective bundling)	Software maintenance, customer support: more detailed information of in-the-field software operation is available which helps to faster determine failure causes in collaboration with the software development department
Integration challenges	Procrastination of developers during identification and reparation of software failures based on software operation information, coping with large amounts of acquired operation information (identifying what information is most relevant in which situations and cope with diversity of information during analysis)	Assigning responsibilities to employees in involving operation information in product software processes, while ensuring a balance between (1) the liberty of an employee and its team, and (2) improving the performance and efficiency of a department as a whole	Based on large amounts of acquired operation information, correctly determine what are actual causes of software failures and what additional (operation environment) information is needed to do so if those causes can not be correctly determined
Integration side effects	Operation information can be used to convince development management in taking release planning decisions, exception handling mechanism of the software was extensively refactored to increase software quality	Information on software usage is also gained, e.g. insight in a customer's software update policy can be gained	Customers feel taken seriously, especially when they are contacted after providing information regarding their software operation
Return on Investment[a]	Software quality (robustness) increase of 25%, decrease of software maintenance time of 50%[b]	Unhandled exception occurrence decrease of 40%, customer satisfaction increase of 25%	Customer support time decrease of 50%, software quality increase of 25%
Main future challenge	Knowing what are the functional requirements of the main customer (end-user) types, and to ensure that relevant, reliable operation information is extracted while data acquisition increases	Realizing a customer-specific approach in terms of software licensing and customer support, and finding an optimal balance between steering processes through operation information, and sustaining a leading role in industry by implementing a software product vision	Making sure that operation information is used and prioritized as effective as possible, while data acquisition sources and resulting operation data amounts increase

[a] All percentages are averages of rough estimations made by interviewees

[b] Before operation information integration, software failures were frequently unreproducible and were never repaired

Table 5. Interview results

Although increase of software quality was considered a significant return on investment, a particular rival hypothesis regarding software quality increase was postulated often during the action research study and the reflective interviews. Various employees pondered over the actual cause of the decrease of received error reports: had the software quality actually been improved, or was there a random downward trend in software usage (or, more particularly, error report submission)? Error report submission history (see figure 4) indicates that software quality actually has been improved during period our study was conducted. While the number of submitted error reports increased about linearly

from February, 2009 until June, 2010, this trend was broken in September, 2010 (the drop during July and August could well caused by summer vacation). In this month, a new major release of CADProd was released and delivered to customers, causing a slight increase in number of CADProd users.

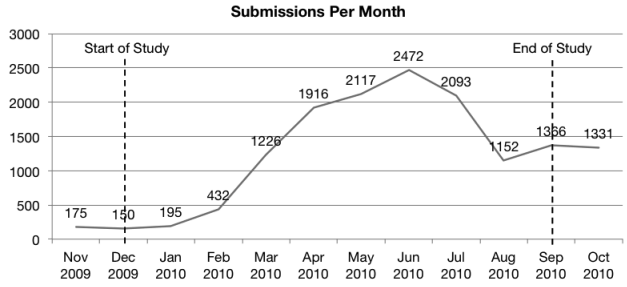

Fig. 4. CADProd error report submission during our study, showing a submission decrease of about 45%

5.3 Lessons Learned

The lessons learned listed below have been identified based on interview session results as well as observations during our presence at the vendor.

1. Integration Processes Should be Lean The effects of integrating operation information in product software processes should not be underestimated. Additional processes with corresponding responsibilities may be required to ensure effective and continuous integration of acquired operation information (for example, registering software maintenance tasks based this information and delegating those tasks to the right employee(s)). Vendors should ensure that additional (administrative) tasks caused by integration of operation information are handled in a pragmatic and lean way: additional administration may negate the time gain caused by integration of operation information.

2. Integration Responsibilities and Results Should Be Evangelized An internal manager that has affiliation and experience with development-, business- and customer-related processes should be made responsible for integration of acquired operation information with those processes, since acquired operation information will not integrate automatically: during our action research study at CADComp, we observed that making integration of operation information everyone's shared responsibility, is effectively equivalent to making no one responsible. Inter alia, such a manager should ensure that the potential and results of the 'SOK-supported' process both are communicated clearly and frequently: this increases awareness and acceptance of the new way of working, both among employees as well as at management level. Evangelism of SOK integration potential and results is key in integrating operation information.

3. SOK Integration Opens Up Black Boxes In line with expectations of CADComp employees, integration of acquired operation information improved software maintenance, software product management and customer support processes: the time needed to reproduce software failures was decreased, deeper in-

sight into (and awareness of) in-the-field software operation and end-user behavior was gained, and customer satisfaction was increased. Operation information is for prioritization of fixes for software failures that are actually experienced by end-users, which may decrease the time required for software maintenance and customer support. However, as became clear after integrating CADProd operation information with CADComp processes, unanticipated improvements may result from effective SOK integration. For example, since developers are made aware of in-the-field software operation quality, SOK integration may result in a more customer-central, pro-active development mentality ('*build what the customer will use, before the customer asked for it*'). Also, integration of operation information in product software processes may clarify and speed-up interdepartmental communication as well as communication between employees and management. Improvement areas that were unnoticed before, are highlighted as such after (and potentially as a side effect of) SOK integration.

4. Continuous Refinement of SOK Integration Objectives and Requirements Leads to Optimization of Integration Results Integration results are dependent on integration objectives and requirements. Since product software processes and activities change, as well as software operation environments and customer demands, integration objectives and requirements should be evaluated and refined continuously to correspond to both a vendor's product strategy as well as a vendor's customers needs. On the long term, software vendors should attain a balance between using operation information to steer their product software processes, and adhering to their product vision and strategy.

These lessons learned may serve as a guiding substrate for similar vendors in integrating information of in-the-field software operation.

5.4 Threats to Validity

The validity of the study results is threatened by several factors. First, construct validity of our study is threatened by the fact that the researchers conducting the study were involved in objects of study (e.g. product software processes implemented at CADComp): observations or conclusions could be biased. This threat is addressed by adhering to the principles for canonical action research elicited by Davison et al. [3] (see section 3.2). For example, the researchers and software vendor being studied agreed on a research plan describing the shared objectives and data that was to be collected during the study. Also, both the researcher and the vendor reflected upon the outcomes of the study by means of semi-structured interview sessions.

Secondly, primary threat to the internal validity of the study is the relation between the instantiation of the SOK integration template method at CADComp and the subsequent software quality increase perceived by CADComp interviewees. Although it is a challenge to isolate the particular influence of template method instantiation on improvement of CADComp's product software processes, we regard CADComp's error report submission history (as analyzed in section 5) as representative of the extent to which CADComp's software

maintenance, software product management and customer support processes are improved through instantiation of the SOK integration template method.

Thirdly, external validity is threatened by the fact that the SOK integration template method was instantiated at only one software vendor, during a limited period of time. While we acknowledge this threat, we regard the study as repeatable with the same results, presuming similar circumstances (e.g. similar operation information, processes, software vendors, etc.)

6 Conclusions and Future Work

All too often in industry, software vendors acquire large amounts of valuable software operation data, without effectively using these data in advancement of their processes. Operation information extracted from operation data is not structurally integrated with product software processes, leaving vendors in the dark regarding in-the-field software performance, quality and usage, as well as end-user feedback. Vendors are in need of an approach that supports them in accomplishing such integration.

We presented a template method that aids product software vendors in (1) identification of relevant and valuable operation information, (2) analysis of target processes and their integration environment, (3) integration of selected information in, and transformation of, target product software processes, and (4) presentation of integrated operation information. During an action research study of ten months performed at a European software vendor, the template method was instantiated to improve the vendor's product software processes through integration of acquired operation information.

Evaluation of the study shows that typical product software processes like software maintenance, software product management and customer support benefit from structural integration of operation information in terms of software quality, operation knowledge and customer intimacy. Based on this evaluation, four lessons learned are identified that may serve as a guiding substrate for similar vendors, in integrating information of in-the-field software operation with their product software processes. We regard the SOK integration template method and lessons learned as an adequate early answer to the main research question of this paper, '*How can product software processes effectively be improved with acquired information of in-the-field software operation?*'. We demonstrated how product software processes can be improved pragmatically but measurably, without adhering to strict requirements from cumbersome maturity models or process improvement frameworks.

Future work will include additional action research or case studies to instantiate and evaluate the SOK integration template method in industry, and therewith further demonstrate its soundness and utility. Further research is also needed to mature the identified lessons learned towards generic guidelines or principles for effective integration of software operation information.

Acknowledgements

We thank CADComp and its participating employees for cooperating and sharing experiences.

References

1. Conradi, R., Fuggetta, A.: Improving Software Process Improvement. IEEE Softw. 19(4), 92–99 (2002)
2. Dangle, K.C., Larsen, P., Shaw, M., Zelkowitz, M.V.: Software Process Improvement in Small Organizations: A Case Study. IEEE Software 22, 68–75 (2005)
3. Davison, R.M., Martinsons, M.G., Kock, N.: Principles of canonical action research. Information Systems Journal 14, 65–86 (January 2004)
4. Fitzgerald, B., O'Kane, T.: A Longitudinal Study of Software Process Improvement. IEEE Software 16(3), 37–45 (1999)
5. Glerum, K., Kinshumann, K., Greenberg, S., Aul, G., Orgovan, V., Nichols, G., Grant, D., Loihle, G., Hunt, G.C.: 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. pp. 103–116. ACM (2009)
6. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design Science in Information Systems Research. MIS Quarterly 28(1), 75–105 (2004)
7. Iversen, J.H., Mathiassen, L., Nielsen, P.A.: Managing Risk in Software Process Improvement: An Action Research Approach. MIS Quarterly 28(3) (2004)
8. Kuilboer, J.P., Ashrafi, N.: Software process and product improvement: an empirical assessment. Information and Software Technology 42(1), 27–34 (2000)
9. Miler, J., Górski, J.: Risk-driven Software Process Improvement - a Case Study. In: EuroSPI'04: 11th European Conference on Software Process Improvement. Springer (2004)
10. Pettersson, F., Ivarsson, M., Gorschek, T., Öhman, P.: A practitioner's guide to light weight software process assessment and improvement planning. Journal of Systems and Software 81(6), 972–995 (2008)
11. van der Schuur, H., Jansen, S., Brinkkemper, S.: A Reference Framework for Utilization of Software Operation Knowledge. In: SEAA'10: 36th EUROMICRO Conference on Software Engineering and Advanced Applications. pp. 245–254. IEEE Computer Society (2010)
12. van der Schuur, H., Jansen, S., Brinkkemper, S.: Reducing Maintenance Effort through Software Operation Knowledge: An Eclectic Empirical Evaluation (2011), accepted for publication in the proceedings of the 15th European Conference on Software Maintenance and Reengineering (CSMR 2011)
13. Smite, D., Gencel, C.: Why a CMMI Level 5 Company Fails to Meet the Deadlines? In: Product-Focused Software Process Improvement, Lecture Notes in Business Information Processing, vol. 32, pp. 87–95 (2009)
14. Susman, G.I., Evered, R.D.: An Assessment of the Scientific Merits of Action Research. Administrative Science Quarterly 23(4), 582–603 (1978)
15. van de Weerd, I., Brinkkemper, S.: Meta-Modeling for Situational Analysis and Design Methods. In: Handbook of Research on Modern Systems Analysis and Design Technologies and Applications, pp. 38–58. Information Science Reference (2008)