An Empirical Study of the Complex Relationships between Requirements Engineering Processes and Other Processes that Lead to Payoffs in Productivity, Quality, and Risk Management

Daniela Damian and James Chisan

Abstract—Requirements engineering is an important component of effective software engineering, yet more research is needed to demonstrate the benefits to development organizations. While the existing literature suggests that effective requirements engineering can lead to improved productivity, quality, and risk management, there is little evidence to support this. We present empirical evidence showing how requirements engineering practice relates to these claims. This evidence was collected over the course of a 30-month case study of a large software development project undergoing requirements process improvement. Our findings add to the scarce evidence on RE payoffs and, more importantly, represent an in-depth explanation of the role of requirements engineering processes in contributing to these benefits. In particular, the results of our case study show that an effective requirements process at the beginning of the project had positive outcomes throughout the project lifecycle, improving the efficacy of other project processes, ultimately leading to improvements in project negotiation, project planning, and managing feature creep, testing, defects, rework, and product quality. Finally, we consider the role collaboration had in producing the effects we observed and the implications of this work to both research and practice.

Index Terms—Requirements engineering, process improvement, process interactions, empirical investigation.

1 Introduction

Requirements engineering (RE)—the elicitation, definition, and management of requirements—is often cited as one of the most important, but difficult, phases of software development [4]. Attention to upfront requirements activities has been said to produce benefits such as preventing errors, improving quality, and reducing risk throughout software development projects [4], [28]. Studies conducted by the Standish Group [33] found a striking 74 percent project failure rate, while 28 percent of projects were cancelled completely. The study suggests that the top factors of failure are related to requirements problems, including lack of user input, lack of a clear statement of requirements, and incomplete and changing requirements.

However, this study is based on a survey of past projects and concrete evidence based on systematic studies of the role of good RE practice in software development is very limited [9]. This becomes problematic both for research on the fundamentals of software engineering and RE research exchange with practitioners. Panels of researchers and practitioners at major international conferences on requirements engineering have repeatedly considered the topic of RE research adoption (e.g., [1], [21]). One of the major issues they identified is the lack of concrete knowledge of what organizations can gain from applying state-of-the-art requirements approaches [21].

In this paper, we present research toward filling the gap between claims in the literature and requirements engineering practice. We conducted a 30-month *explanatory* case study at an organization that had revised its RE process (REP) and which provided us with the opportunity to assess the effects of improved RE over an entire project lifecycle. The evidence collected enabled us to explain how RE processes played a part in improving developer productivity, product quality, and project risk management. The case study was conducted in three separate stages. Earlier publications of our research report findings from the first two stages, in particular, the perceived benefits of the improved RE practice in the early [6] as well as the downstream [7] stages of development in the studied organization.

Here, we discuss evidence from the third stage of our three-stage case study. In this stage of our research, we sought to understand how the RE process can interact with other processes and how this interaction may have contributed to the effects we observed earlier in our study. We present the evidence from this work in the context of the entire research study and specifically build on the findings

• *The authors are with the Department of Computer Science, Engineering/ Computer Science Building (ECS), Room 504, University of Victoria, 3800 Finnerty Road, Victoria, BC, Canada V8P 5C2.*
E-mail: danielad@cs.uvic.ca, james@chisan.com.

Manuscript received 10 June 2005; revised 17 May 2006; accepted 1 June 2006; published online 9 Aug. 2006.
Recommended for acceptance by N. Maiden.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSE-0179-0605.

of the earlier research stages in explaining the role played by the improved RE practice in producing these benefits.

The contribution of the paper is twofold: 1) We report on the study of the impact of the RE process and other processes as practiced at a commercial software development organization and 2) we provide a detailed *explanation* of how improvements in RE practice can lead to improvements in productivity, quality, and risk management. In providing this explanation, we construct a map that shows the perceived benefits as reported in our earlier publications and the complex interaction between the RE process and other processes that contributed to the identified benefits. Specifically, we unveil a rich interaction between requirements engineering and other development processes such as project planning, testing, and development, ultimately leading to payoffs in productivity, quality, and risk management. We also describe how certain aspects of the REP improvement have contributed more significantly to this interaction. Finally, we consider the role of collaboration in producing the effects we observed and the implications of this work to both research and practice.

In Section 2, we review the limited evidence that exists about RE payoffs and, in particular, discuss expected benefits as a result of rigorous RE practice. Section 3 describes our overall research question and the methodology we used to develop an understanding of the relationship between improved RE processes and improvements in software development within the organization. Section 4 provides background information on the organization where we conducted the research, its revised REP, and the evidence we collected in our earlier stages of research and which indicated that improvements in REP were related to gains in productivity, product quality, and risk management. Section 5 explicitly summarizes this evidence. Section 6 describes the more in-depth investigation of the interaction between the RE process and other software processes in the organization. Section 7 discusses in detail how this interaction represents a complex relationship between the improved REP and improvements summarized in Section 5. We finally look at this research's implications for research and practice in Section 8. Section 9 discusses the threats to the validity in our research and issues of generalizability of these findings to other software organizations.

2 Related Work

In spite of the claims in the literature that requirements engineering is positively associated with benefits to project management, downstream development, and software quality, e.g., [1], [4], empirical evidence of such payoffs in practice is very limited. Notable exceptions include data released by NASA suggesting that time spent on RE activities negatively correlates with project cost and schedule overruns ([9, p. 45]). However, this study provides only a high-level statistical comparison, without exploring how REP may have contributed to these effects during the development life cycle.

We reviewed the literature on requirements engineering and found a number of reports on RE process improvement [5], [10], [16], [18], [19], [20], [29], [35]. These reports provide details of the improvement process as well as lessons learned from the improvement initiatives. Kauppinen et al. [18], [19], [20] report from a study of four Finish organizations that had introduced requirements engineering to their product development and discuss the organizations' successes factors and challenges. One of their main insights is that REP improvement requires culture change in the organization and not merely a change in process or supporting technology. Kauppinen et al. also report that REP implementation is a demanding undertaking whose success depends on human factors such as motivation, commitment, and enthusiasm. Similarly, Claus et al. [5] report on REP improvement at a large German organization and identify that a key to promoting process acceptance is the involvement of all project members, including management. A more detailed report on REP improvement at Ericsson Eurolab, by Jacobs [16], describes an REP improvement that emphasized structuring requirements and breaking each requirement into subrequirements. This resulted in a major change in the organization's attitude toward requirements engineering. The authors emphasize not only the change in the designers' behavior but also an interaction with other processes in the organization that could have contributed to this effect. Specifically, as a result of better understanding the customers' requests, the process of communication with customers was improved, as well as the processes of software inspections being made more formal and becoming more effective in identifying faults in the system.

The conclusion we draw from these reports is that requirements engineering activities are bound tightly with other system engineering activities and that a complex interaction between REP and other processes in the organization may exist. As a result, it becomes difficult to measure the REP improvement benefits in isolation, particularly before a project has been completed. It is difficult to control and understand the effects of REP over the course of a nontrivial development project, making the empirical study of RE very difficult. This may also explain why organizations that implement process improvements rarely measure the costs and benefits of RE activities—compounding the scarcity of comparative data [2], [9].

Social factors also play an important role in RE [17]. Project and requirements knowledge is being created and manipulated through collaboration processes rooted in social and organizational practices that influence how information about requirements is communicated and managed throughout the development life-cycle. Consequently, introducing new or revised RE methods can rest on promoting cultural change by strategically managing concerns process adoption [20], [21]. RE practice and its success thus *has to* be understood not only from a systems development perspective, but also from a social and organizational context. Quantitative assessment of RE improvements needs to be complemented by qualitative and subjective assessment based on levels of satisfaction of developers and managers, as well as clients involved in the RE practice [5], [20].

A broader examination of the literature on process improvement suggests that implementing REP improvement can lead to specific beneficial outcomes. These

outcomes are often described by process improvement models such as the CMM [31], ISO/IEC 15504 [14], and the process capability model developed by Sommerville and Sawyer [32]. These models all suggest that rigorous requirements engineering processes lead to increases in productivity, enhanced quality, and improved risk management. Unfortunately, they provide few details on how these benefits are realized or how they can be measured.

For insight on how productivity, quality, and risk management can be operationalized in the context of REP improvement, we considered the following work:

Developer *productivity* is cited often, usually as it pertains to increases in efficiency, such as preventing the need for implementation rework or lowering the cost of development. Lauesen and Vinter [22] consider RE performance strictly in terms of hours saved when comparing particular RE techniques, such as user scenarios and performance specification. In contrast, Wohlwend and Rosenbaum [36] provide a much simpler test, reporting that RE can be considered successful when software delivery is ontime.

The literature also suggests that enhancing *quality* through REP improvements leads to software that is more likely to be technically correct and satisfactory to customers. Herbsleb and Goldenson [12] report that mature organizations, according to the CMM, exhibited significant improvements in product quality and customer satisfaction. Similarly, after software improvement initiatives at Schlumberger, engineering teams that had formally been plagued with delivering incomplete functionality began to ship software that was "correct" [36].

Although productivity and quality are important factors in the development of software, Brodman and Johnson [3] found that many companies seek to implement software process improvements primarily to reduce their exposure to risk, rather than to optimize their software production. The companies they surveyed expressed a clear interest in the accurate assessment of costs and scheduling and in decreasing the variability of project success and performance. Although costs and scheduling can be considered productivity concerns, forecasting during the initial stages of a project is clearly a matter of risk management [13] and is tightly related to activities of requirements management. As Paulk [27] notes, both the CMM and ISO share the same goal, to "consistently improve project performance," implying that both models reduce variability across projects.

Finally, there is little doubt that these effects, in turn, have an impact on the people who realize them, exemplifying the importance of perception among engineers. For example, Wohlwend and Rosenbaum [36] note that, after successful software improvements, developer morale improved significantly. Others [3] suggest that companies have also found that less overtime leads to improved confidence, less turnover, and increased intraorganizational cooperation. Hall et al. [11] specifically link improved REPs to better staff retention rates, while Humphrey et al. [13] found that "Pride [from continuous improvement] feeds on itself ... leading to success." However, the details of the relationship between RE and the benefits ascribed to it are still an outstanding research question that challenges our more complete understanding of the role of RE in software development. Software engineering researchers and practitioners are in need of systematic empirical studies on the benefits of rigorous RE practice throughout the entire software life-cycle and they require detailed accounts of how these benefits were realized as a result of improvements in RE. The research presented in this paper seeks to answer this question by providing a detailed account of how REP affected other development processes and how the entanglement of processes helped produce the perceived benefits in productivity, quality, and risk management.

3 Research Questions and Research Methodology

The evidence discussed in this paper is from our case study designed to examine the effects of RE at the Australian Center for Unisys Software (ACUS), an organization that improved its software processes with a focus on redefining their requirements engineering process. The goal of our research was to study a project through its complete development cycle immediately following their software process improvement initiatives. In particular, we designed an *explanatory* case study [37] in which existing theories about the effects of "good" RE in software development, namely, improvements in productivity, quality, and risk management, guided our investigation of the role of RE in software practice. To address the following research question:

> *How do improvements in requirements engineering processes relate to improvements in productivity, quality, and risk management?*

we intended to observe how RE affected software development at ACUS and explain *how* these effects were realized.

The case study was conducted over a 30-month period (August 2001-February 2004) and, due to the extended and complex nature of the case study, our investigation was conducted in three separate stages. Although we only provide detailed findings from the third stage in this paper, we present here the detailed research questions that guided each stage, the kind of data collected, and when, within the software project lifecycle, it was collected. The detailed explanation we construct in order to answer our overall research question draws upon data from all three case study stages. Therefore, providing thorough background information about our research methodology is important in understanding the contributions of this paper.

Although similar data collection methods were used throughout the study, each stage had unique characteristics that resulted in slight variations in methods. The methods used in all stages included questionnaire, interviews, and document inspection. The requirements process documentation was studied, as well as other (current and historical) project information. This being the first time the requirements process was rigorously defined at ACUS, historical data was very limited and, instead, we relied on the extensive professional experience of ACUS engineers to provide comparison to previous practice. ACUS is a company with low employee turnover, so most of these participants were very familiar with the history of the product and, most importantly, of the requirements

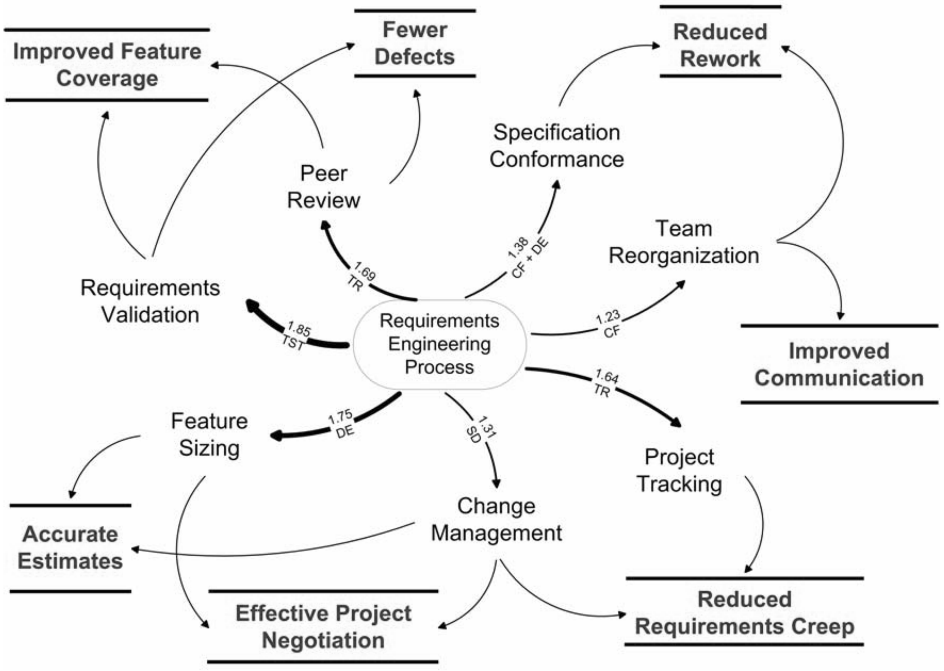

Fig. 1. A rich interaction between the REP and other development processes contributed to gains in productivity (communication, rework), quality (defects), and risk management (estimations, feature coverage, negotiation, requirements creep). Observed payoffs are shown on the periphery.

management process during the last 15 years. The majority of the evidence was collected anonymously to protect respondents from the scrutiny of the organization. In particular, the names of participants were unknown to ACUS senior management to allay any concerns participants may have had about providing honest responses. Further details about data collection and analysis methods for each stage are provided in Sections 5 and 6.

The first research stage was designed to answer the question:

1. How do improvements in the RE practice impact the early stages of development?

Early in the lifecycle, we expected immediate benefits to include enhanced estimation ability, project scope negotiation, and increased problem understanding. Data collection and analysis took place in the first eight project months before design began.

The second stage was designed to answer the questions:

2. How do improvements in the RE practice impact the downstream development stages?
3. Which components of the RE process were more significant in contributing to this impact?

At this stage, we expected benefits to include less implementation rework, increased product quality, and reduced exposure to risk. This stage collected and analyzed data near the end of the project when development was nearly complete.

Recognizing the complexity of interactions between the requirements engineering and other development processes at ACUS, we realized that it would be rather difficult to attribute these effects solely to the REP. While a systematic investigation of direct impact of confounding factors was outside the scope of this study, this stage was designed to answer the question:

4. How could the interaction between REP and other processes have contributed to these results?

This final investigation was pursued after product deployment to customers.

The results of our entire case study are summarized in an interaction map, of which graphical representation is presented in Fig. 1. By combining evidence from all stages in the research, an answer to our main research question emerged. We identified a complex but positive relationship between improvements in the RE process and other processes at ACUS and benefits in the early stages of development, in downstream development, and in post-deployment software quality.

We proceed by discussing our research setting: the company and its RE process improvement program. We then briefly introduce, in Section 5, the findings from the first two stages in our research, namely, the perceived payoffs of the RE practice in early and downstream development at ACUS (described in detail in [6], [7]). In this paper, however, the emphasis is placed on providing a comprehensive explanation of the role that improvements in REP played in creating these payoffs. Section 6 details the third stage of our research that investigated the interaction between the REP and other processes that resulted in the observed RE payoffs. The evidence from this stage of our case study is presented for the first time here. It represents an essential component in constructing an explanation of the role of requirements engineering in producing these payoffs. Besides this evidence, the major contribution of this paper is this explanation (together with the graphical

representation in Fig. 1), which draws upon the analysis of evidence from all three stages of our research.

4 Research Context: The Company and Its RE Process Improvement Program

In this section, we describe the research setting, the company, and its program for improving its requirements engineering processes.

4.1 The Company, Challenges in Requirements Practice, and a Revised RE Process

This research was conducted at ACUS. ACUS is a 130 employee software development center within a multisite organization with product management and marketing divisions in the US (hereafter referred to as their marketing unit) and customers worldwide. ACUS has developed a successful software product line with a 20-year history. It is a large multiplatform product, approximately 4 million lines of code, written in various languages, including Java, C++, Cobol, and Smalltalk. This software is an application development platform supporting the development of enterprise-scale, transaction intensive applications. It is typically customized by customers or third party developers before being usable by end users. The product has a core architecture and each new release adds features that address important customer needs.

For ACUS, new features are normally requested by a US-based marketing unit which acts as a surrogate customer, expressing both market needs and strategic corporate concerns. Although there is no direct access to users during development, ACUS does provide support and receives feedback from customers after deployment.

In August 2001, ACUS engaged in a software process improvement initiative to reach CMM Level 2. RE was identified as the primary Key Process Area in need of improvement. Prior to this initiative, ACUS effectively had no well-defined RE process. Despite having major stake holders spread across several continents (e.g., North America, Australia, Europe), the product development group had limited experience with formal requirements management processes. As a result, ACUS faced significant challenges. Projects at ACUS typically suffered from significant requirements creep, schedule overruns, and cost overruns. ACUS management had difficulty understanding the requested features and providing reasonably accurate development estimates. Managers at ACUS cited ineffective negotiations between ACUS and their marketing unit. Ineffective negotiations made it difficult to align development capacities and marketing needs. In particular, their requirements practice suffered in the following ways:

- Requirements were provided to ACUS in the form of one or two line "system features."
- Once communicated, there were instances where features were not fully defined or documented, providing insufficient information to developers.
- Any collective understanding about features was largely disseminated by word of mouth.
- Individual designs depended heavily on the designers' interpretation of those feature requests.
- Inadequate requirements management and control compounded these problems by enabling the US marketing unit to demand new features late into the development cycle.

With strong support from the management, the organization developed an RE process that sought to specifically address these challenges. The revised process defined a distinct, discrete phase of the project during which requirements would be elicited, analyzed, and negotiated. When this requirements phase had ended, requirements were to have been agreed on, committed to, and then baselined during project planning. Subsequent requirements changes would only be allowed after being approved in a formal requirements management process. The new REP consisted of the following components:

1. *Feature Decomposition* into technical requirements. Requested features were analyzed in the context of the existing product architecture and derived into detailed technical requirements. Each analysis was led by a designated developer involved with the design of similar features in the past and moderated by the project manager in a Group Analysis Session (as in 3 below).
2. *Requirements Traceability*. Traceability links from (technical) requirements to requirements rationale, design documents, and test scenarios were created in the Requirements Document by the project manager. These links were identified and discussed and agreed upon in Group Analysis Sessions (as in 3).
3. *Group Analysis Sessions* were organized on a weekly basis to carry out feature decomposition and typically attended by 6-10 engineers. They played a central role in the revised REP, as feature decomposition, as well as test scenario and traceability links, were all carried out and/or identified during these meetings.
4. *Cross Functional Teams* were formed to participate in the Group Analysis Sessions.[1] These teams were composed of personnel from different functional departments (i.e., design, code, test, and product information).
5. *Structured Requirements*. The requirements documentation included information on the requested features and their associated detailed technical requirements (as derived during the group analysis sessions). Each requirement specified its rationale, test scenario(s), and traceability links to rationale and test scenario(s). These documents were drafted and maintained in Rational's RequisitePro by the Project Manager and accessible by all engineers.
6. *Testing According to Requirements*. Test scenarios were defined for each detailed technical requirement so as to accommodate testing against requirements during the system test. They were defined during the Group Analysis Sessions with the instrumental contribution of the testing personnel.

Additionally, the revised practice placed more emphasis on the process of requirements change management as part

1. And were responsible for subsequent implementation.

TABLE 1
Summary of Observed REP Payoffs during Development

RE Payoff	Operationalization at ACUS	Trend	Evidence of REP impact
Increased Productivity	Problem Understanding	↑	100% positive response
	Communication		
	Superfluous communication	↓	85% positive response
	Effective Communication	↑	57% positive response
	Informed decisions by developers	↑	91% positive response
	Rework	↓	65% positive response
Increased Quality[2]	Support Requests	↓	45% fewer
	Post-Deployment Defects	↓	55% fewer
Improved Risk Management	Estimations	50% improvement	100% positive response
	Feature Coverage	↑	
	Requirements Creep	↓	90% positive response
	Project Negotiations	↑	100% positive (manager) response

2. This data became available six months after release, during the third stage in our case study.

of the broader change management processes at ACUS. Each "change request" followed a formal process that required a documented description and analysis, review, and approval by a change control board.

5 Payoffs of the RE Practice at ACUS

Our pursuit of the first three research questions in the first two stages of the case study, namely:

1. How does the RE practice impact the early stages of development?
2. How does RE practice impact the downstream development stages?
3. Which components of the RE process were more significant in contributing to this impact?

provided us with evidence of perceived RE payoffs at ACUS. A report of this evidence has been published before [6], [7] and Table 1 summarizes conclusions from this data: It outlines the perceived payoffs, their operationalization at ACUS (i.e., the factors examined to measure these payoffs, as suggested in the literature surveyed in Section 2), and data from the study participants, implying REP's role in producing these effects. In summary, having a rigorous REP was perceived as having provided a strong foundation for improvements in developer productivity, product quality, and risk management.

This evidence represents important background information for our investigation into how these payoffs were realized at ACUS, which is the main focus of this paper. Our intention is to provide here the detail that is necessary in constructing the map of interaction between REP and other processes at ACUS in producing these payoffs, as shown in Fig. 1.

5.1 Data Collection and Analysis Methods

During these first two stages in the case study, on-site observations of requirements analysis and negotiation sessions were possible as one researcher was on site for 12 months of the project. To answer the first research question, 34 members of the software engineering, management, and product information departments were invited to participate in the study. A questionnaire elicited input about the perceived effectiveness of the REP and, in particular, about the immediate benefits of the revised REP before the design and coding stages.

Pursuing second and third research questions in the second stage, 31 project members from the same functional departments were invited. Interviews and a questionnaire were designed to determine perceptions of how the REP impacted downstream development stages. In particular, how engineers used requirements and which components of the REP were perceived as having impacted their work. Additionally, change requests documents, project development estimation data, and entries within the requirements management tool (Rational Requisite Pro) were analyzed during this stage. These questionnaires and interview questions are available at: http://vigilant.segal.uvic.ca/acus.

5.2 Evidence of RE Payoffs in Early and Downstream Stages of Development

An analysis of data from the first two stages of investigation indicates perceived improvements in productivity, quality, and risk management directly attributable the revised REP. We observed both tangible and intangible effects of the improved REP at ACUS. First, developers perceived that the "soft" aspects of their work were improved, such as their ability to understand the customers' requests, developers' ability to make more informed decisions, and the quality of their communication. Second, the findings also indicated significant perceived improvements in project estimations and forecasting. These effects were perceived as the result of the revised REP as respondents' opinions indicate, as summarized in the last column of Table 1. Finally, perhaps the most compelling result was in product quality: The data indicates a significant reduction in support requests and postdeployment defects compared to previous releases, suggesting that users encountered far

fewer difficulties with the product that had been produced under the revised REP.

5.2.1 Increased Developer Productivity

A variety of observed effects indicated that productivity significantly increased after the revised REP had been implemented in the organization. Having a rigorous REP was perceived as having improved problem understanding, leading to informed decisions, reduced rework, and greatly improving the quality of communication among engineers in the organization.

Problem understanding, to some extent, forms the basis by which all other aspects were improved. Engineers, including designers, programmers, testers, and documenters, all indicated that the REP was vital for them because requirements revealed details, dependencies, and complexities of features early on (positive responses: 100 percent). When respondents were asked how they utilized detail revealed to them during the requirements, 91 percent of engineers indicated that it allowed them to make informed decisions. Engineers make micro-decisions throughout their work that reflect their understanding of customer needs. These decisions have a long-lasting impact on a myriad of manifestations from design decisions to effort estimations. In part, this may explain why rework during the project, according to 65 percent of respondents, had decreased significantly compared to similar past projects.

Furthermore, evidence indicates that project communication patterns changed in significant ways after the improvement. Engineers felt that open, effective dialog which had been engendered during group analysis sessions early in the project continued to foster improved communication across functional groups within the organization (positive responses: 57 percent). Superfluous communication that had formerly been necessary when seeking clarifications, reiterating information, and coordinating among developers decreased significantly, saving developer time. Requirements acted as a common ground during the project lifecycle, enabling engineers to use requirements artifacts to refamiliarize themselves with feature details (positive responses: 85 percent). In short, rather than engage in costly face-to-face synchronization with peers, engineers were able to more efficiently rely on information contained in the requirements to accomplish this task.

5.2.2 Improved Product Quality

Evidence that became available during stage three of the investigation suggests that product quality and end-user experience has been greatly improved compared to previous projects. Although developer sentiment had hinted that quality was higher, concrete measurements were sought. The organization carefully tracked user-reported deficiencies and product defects after release, both of which show a marked decline compared to comparable past releases.

User-reported deficiencies (URD) typically include any instance where users have reported having difficulty using the product, either due to legitimate defect, missing functionality, or user error. Company statistics indicate the there have been 45 percent fewer URDs compared to the past two releases, suggesting that users encountered fewer far difficulties using the product that had been produced under the revised REP. Similarly, defects reported during the same period show a decrease of 55 percent compared to defect rates of the same past projects. As reported by team leaders involved with previous product releases, this release was similar to the previous two in terms of the number and complexity of the requested features.

5.2.3 More Effective Risk Management

Managers were also able to leverage requirements to greatly enhance their ability to manage risk. Although the most striking improvements were made in project estimations and forecasting, managers indicated that they were able to more effectively negotiate project scope, curtail requirements creep, and confidently assure feature implementation.

An analysis of project estimations made before and after the requirements analysis phase indicated significant improvements in the accuracy of effort estimations. The estimations were performed by using an expert-judgment bottom-up feature estimation method by experienced team leaders in consultation with the individual developers responsible for the implementation of particular features. A comparison of the actual effort required and effort estimations at the beginning of the project before and after requirements analysis shows a 50 percent reduction in the total estimation error. Respondents reported unanimously that this improvement was due to more thorough understanding of the features as a result of the detail developed in the software requirements.

The revised REP was directly linked to improvements in feature coverage. The new practice of testing against requirements had provided evidence to managers and SQA personnel that requirements were indeed implemented and delivered. The only features that were fully or partially dropped were agreed upon as part of the change management process. Worth noting is the unanimous positive response from the managers with regard to the direct relationship between the revised REP and project negotiations.

5.3 The Role of Collaboration

During these early stages, we also asked engineers to consider which particular improvements of the REP affected their software development activities (i.e., design, implementation, documentation, and testing). The relative responses are shown in Fig. 2, indicating that not all REP improvements were perceived as equally important to development at ACUS.[3] To our surprise, a strong impact by *cross-functional teams* and *group analysis sessions* was perceived. Collaboration is inherent in both of these components, leading us to consider the importance it plays in achieving the payoffs we observed.

5.4 The Need for Further Investigation

From the analysis of the qualitative data collected during stage one and two (and some quantitative data on estimations and user-reported defects), it quickly became apparent that the relationship between REP and its payoffs was more complex than expected. Evidence of *how* the REP

3. Note that this inquiry did not refer to all REP components as in Box 3.

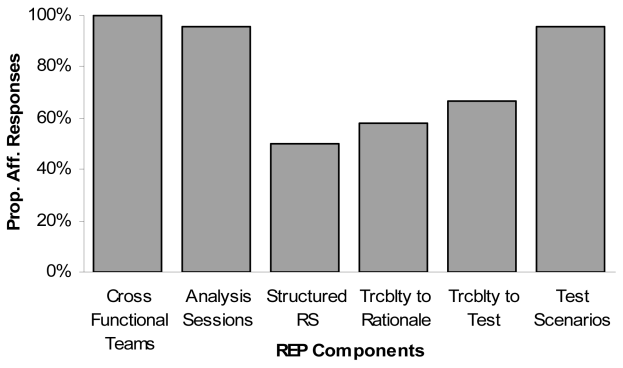

Fig. 2. Perceived importance of REP in relation to stages of the development practice.

led to these payoffs was required and motivated the third stage in our research. Comments by respondents indicated the REP may have additionally interacted with other development processes at ACUS in leading to these payoffs. Table 2 contains a sample of comments made by respondents which suggest that some factors in Table 1 (e.g., "estimations" and "rework" included on black background) were affected by the revised REP as well as other development processes such as "cross-functional teams," "sizing," and "peer-review." The comments are arranged in Table 2 according to the "payoffs" (white on black). The processes listed in the left-hand column indicate which affected-process was named or implied by the comment. For example, "estimation" as one item in our operationalization of risk management has been positively affected by the process of "feature sizing," which in turn was aided by the revised REP: in one developer's words, by "less guesswork, more mechanical and mathematical."

This led us to believe that the payoffs we had observed may have been caused, in part, by REP's interaction with other development processes. The remainder of the paper describes in detail our third research stage. We consider the effect of the REP on other development processes at ACUS, as well as identify which particular components of the REP contributed most significantly to this interaction.

TABLE 2
Sample Comments about an Interaction between RE Process and Other Processes in Leading to the Observed Benefits

Communication	
Cross-func. Teams	- "reducing meeting times and contributing to good communication to other teams" - "provided consistency across functional groups"
Rework	
Conform to Spec	- "less functionality missed due to constant reinforcement of requirements specifications"
Cross-func. Teams	- "involvement across teams had positive effects"
Defects	
Peer Review	- "[fewer defects due to] improved peer review" and "the review peer processes"
Estimates	
Sizing	- "less guess work, more mechanical and mathematical" - "[understanding]* of interdependencies" - "[spend time on RE] to get finer detail … making it possible to have better estimates earlier on"
Change Mgmt	- "reduced the number of changes sneaking into the product"
Feature Coverage	
Peer reviews	- "[successful because] reviews were done with designers and test case authors to validate the 'how' of each test case" - "test specs came under scrutiny, the resulting issues were sometimes abundant"
Testing Against Requirements	- "[test scenarios] provided a good base to start writing test cases" - "easier verification of what you have built" - "targeting the requirements and testing them too caused bugs to be detected earlier"
Requirements Creep	
Project Tracking	- "able to identify schedule risks" - "easier to forecast resource crunches"
Change Mgmt	- "firmly controlled and visible" - "better requirements evolution due to change control"
Project Scope Negotiation	
Change Mgmt	- "our ability to negotiate was enhanced … by the level of requirements elicitation with US marketing unit and … structured documentation of requirements"

* Square brackets denote necessary addendum to succinctly capture the context of the comment being made.

6 A More In-Depth Analysis of the Interaction between REP and Other Processes

6.1 Data Collection and Analysis Methods

A total of 20 managers, team-leaders, and senior engineers from the software engineering, product management, and product information departments were invited to participate in the third stage of our case study. Their positions as managers or team-leads made them uniquely qualified to assess the subtle process interactions which we sought to understand. From these 20 invitations, 15 participated in the questionnaire. These participants were familiar with the history of the product and, most importantly, of the requirements process in place during the last 15 years.

To collect participant opinion about the effects of REP on other processes, a Web-based questionnaire was used. Although a sample is included in Appendix A, the full version of the Web form can be found at http://vigilant.segal.uvic.ca/acus/. The questionnaire was designed to determine the impact (positive or negative) of the revised REP on other major development process areas at ACUS and to determine which component of the REP most contributed to that impact. The process areas and the REP components investigated are listed in Box 2 and 3, respectively (shown in Fig. 3).

The questionnaire asked respondents to rate, on a scale from -3 to 3, the impact of the REP on each of the five major processes and their constituent subprocesses. Respondents were instructed that a positive response indicated a positive impact, while a negative response would indicate hindrance of the process by the REP. For example, for project planning, a respondent might indicate +2, suggesting that the REP had a moderately positive impact on project planning in general.

Box 2. Development Process Areas and their constituent sub-processes[4]

Project Planning
- Feature sizing
- Risk assessment
- Scheduling
- Resource planning
- Change management
- Responsibility allocation
- Requirements tracing

Software Quality Assurance
- Project tracking
- SQA team
- Meetings and reports
- Deviations

Software Configuration Management
- Baselining
- Code change management
- Change metrics
- Role adherence
- Software change control board
- SCM tools

Development
- Team reorganization
- Split leads
- Planning
- Specification conformance
- Code inspection

Testing
- Automation
- Requirements validation[5]
- Peer-review
- Repeated test

Box 3. REP Components

(as described in Section 4.1)

(1) Feature Decomposition
(2) Requirements Traceability[6]
   - Traceability to Rationale
   - Traceability to Tests
(3) Group Analysis Sessions
(4) Cross Functional Teams
(5) Structured Requirements
(6) Testing According to Requirements

Fig. 3. Box 2 and Box 3. [4]A description of these processes and subprocesses is included in Appendix B. [5]Although requirements validation is typically an RE activity and refers to whether requirements express customers' needs, at ACUS, this term was used to refer to the use of test scenarios defined during the requirements analysis stage. [6]Traceability was investigated as two separate components indicated during questionaire 2, but more generally in questionaire 3.

Second, for each process and subprocess, respondents could also specify which component of the revised REP they felt was particularly responsible for the effect on that process. The REP components that were included in the questionnaire were a simplified set of the same REP components investigated earlier in the study (see Fig. 2). For example, respondents may have chosen *feature decomposition* to indicate that this particular REP component contributed to project planning more than any other. So, for each development process and subprocess, participants rated the impact and then chose which REP improvement most significantly contributed to the impact. To simplify the questionnaire, *rationale traceability* and *traceability to test* were combined into *requirements traceability*.

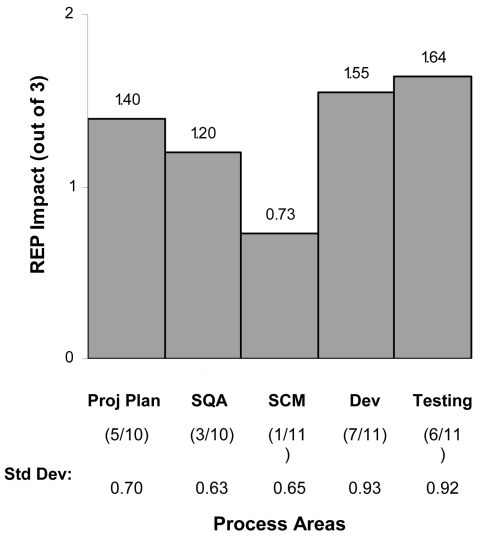

Fig. 4. REP impact on process areas (responses >=2 shown in brackets).

6.2 Findings: The REP Interactions

6.2.1 REP Impact on other Processes

Fig. 4 shows the average responses regarding REP's impact on each of the five major development processes (as outlined in Box 2: Proj Plan = Project Planning; SQA = Software Quality Assurance; SCM =Software Configuration Management; Dev = Development and Testing). As the questionnaire allowed impact responses in the range between -3 to 3 inclusively, 3.0 represents the maximum possible value. A negative impact indicated that the REP hindered the performance of the respective process or subprocess. The ratios shown in brackets report the number of responses which were greater than or equal to 2, relative to the total number of responses, to emphasize the ratio of responses on high REP impact. Although 15 practitioners participated in the questionnaire, each question received between 10 and 11 responses. Although there were only a few negative responses over the entire survey, we did not conduct follow-up interviews with respondents who provided outlying responses. Such interviews would have provided further insight into the rationale behind individual responses.

Similarly, Fig. 5 shows the average response regarding REP's perceived impact on the individual subprocesses grouped according to process area (standard deviations in brackets for each subprocess). For example, within the testing process area, the Requirements validation subprocess received, on average, the most positive responses: 1.85 out of a possible 3.0, which indicated that this process area was positively affected by the revised REP. The chart shows high REP impact in the areas of project planning, testing, and, to some degree, development.

A chi-square test has been performed to compare the number of "positive-effect" responses with the number of "no-effect" together with "negative-effect" responses. In other words, to see whether the number of responses indicating positive REP impact was statistically significant from the other responses. In calculating the test, the number

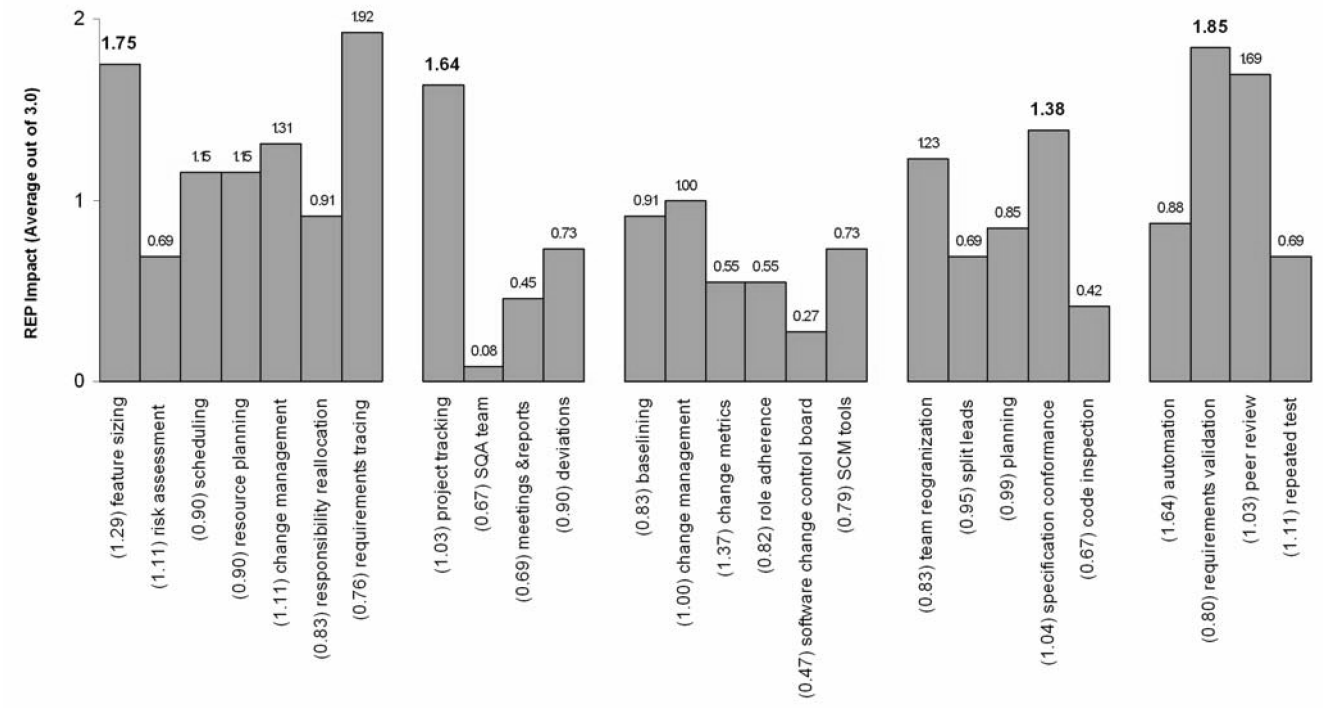

Fig. 5. REP impact on subprocesses according to process area (average of responses -3 and 3).

of positive-effect responses was set to equal the number of responses indicating there was a positive impact by the REP and compared to the number of all other responses (negative or zero). The processes for which the REP impact was significant ($p < 0.05$) are Project planning with its constituent subprocess Feature sizing, SQA with its constituent subprocess project tracking, Development with its constituent subprocesses Team reorganization and Specification conformance, and Testing with its constituent subprocesses Requirements Validation and Peer review. While Appendix C provides raw data showing all responses regarding REP impact, Table 5 in Appendix D shows all results of the statistical test.

6.2.2 A Closer Look at the REP Improvement and Impact

It is also worth considering data which illustrates which components of the improved REP most significantly contributed to the impact on these processes. In addition to assessing REP impact on particular subprocesses, respondents were asked to consider which components of the revised REP contributed to the effects they perceived. Respondents were informed that, by selecting an REP component, they were indicating that that component was predominantly responsible for effects on the process in question.

Then, due to the strong relationship between component identification and perceived impact, component responses were weighted accordingly. For example, if a respondent selected decomposition and indicated a value of 3 for REP impact, we would record a score of 3 in favor of *decomposition* for that particular subprocess. In the same way, for another respondent who makes the same component selection, but indicates a value of 1 as REP impact, we would record a score of 1 in favor of *decomposition* for that particular subprocess. All component responses, weighted in this manner, were added to produce a single score for each component in each process. Tables 6 and 7 in Appendix E show the number of responses for each process/subprocess, per REP component, raw, and weighted responses, respectively.

Table 3 summarizes the component responses for the four most affected subprocesses from the chi-square test and which we included in Fig. 1. Below each subprocess

TABLE 3
REP Impact Component Responses
for Highest-Scoring Subprocesses

		DE	TR	GAS	CF	SD	TST
Feature Sizing	Score	17		**6**			-1
	Resps	7		**3**			1
Project Tracking	Score		9			5	
	Resps		4			2	
Spec Conformance	Score	4	4			2	
	Resps	2	3			1	
Peer-Review	Score	3	11		2		3
	Resps	1	5		1		2
Team Reorg	Score			2	8	2	
	Resps			1	5	1	
Reqs Validation	Score		3				16
	Resps		2				8
Change Mgmt	Score	2	3			8	
	Resps	1	2			4	

DE = Decomposition, **TR** = Traceability **GAS** = Group Analysis Sessions, **CF** = Cross Functional Teams, **SD** = Structured Requirement Document and **TST** = Testing According to Requirements

heading, the REP impact score is indicated, together with the number of responses contributing to this score. It is apparent that participants believed that *feature decomposition* (DE) had the greatest impact on improving sizing. Likewise, *traceability* (TR) positively contributed to both tracking and peer-review. Under the development process area, conformance (implementing specification conformant code) was equally affected by both *decomposition* and *traceability*.

7 Putting It All Together: A Complex Relationship between RE Process and Expected Payoffs

Having completed our study, we are able to step back and consider the complex relationship between the RE process and its tangible benefits. It is now feasible to answer the question our study had originally sought to answer: *How do improvements in requirements engineering processes relate to improvements in productivity, quality, and risk management?*

By combining evidence presented in detail here and earlier evidence summarized in Section 5.2, an answer becomes apparent. The cumulative results illustrate how the REP both directly and indirectly, through the interaction between REP and other development processes, contributed to the observed payoffs. This relationship is illustrated in Fig. 1.

By interpreting the detailed evidence presented in this paper, we provide an explanation of how REP, through the interaction with other processes at ACUS, was perceived as having contributed to the payoffs observed earlier in our research. Specifically, data from Table 1, Table 2, Table 3, and Fig. 5 represent the evidence that connects the elements in Fig. 1. In the center of the diagram, the REP is connected to the subprocesses on which we found a statistically significant REP impact (as described in Box 2 in Fig. 3 and Section 6.2.1), which connect to the observed payoffs (as in Table 1), shown as terminal nodes around the periphery of the diagram. We did not include the subprocesses called "scheduling" and "resource planning" because they are a directly related to feature sizing which is already included. The numerical value on the arrows (also shown as their width) from the REP node indicates the strength of the REP's positive contribution to intermediate processes (as in Fig. 5). The REP components specified in brackets on the arrows indicate which REP improvement primarily contributed to that impact (as in Table 3). Arrows connecting subprocesses processes to the observed payoffs are justified by the qualitative evidence from Table 2.

The discussion of how REP was related to each of the observed payoffs is organized as follows: 1) risk management in terms of accurate estimations, improved feature coverage, effective project negotiations, and reduced requirements creep, 2) quality in terms of fewer defects, and 3) productivity in terms of improved project communication and reduced rework. In particular, each heading below indicates how the REP and its particular components interacted with the other development processes at ACUS in leading to the payoffs on the periphery of the diagram in Fig. 1.

Feature decomposition, sizing, and change management led to *more accurate estimates*. Accurate project estimations were primarily the result of accurate feature effort sizings conducted by engineers who could make extensive use of technical requirements. Interestingly, ACUS achieved this improvement without the use of elaborate estimations methods, such as function points as proposed in [15]. By bringing clarity and focus to its requirements, ACUS was able to leverage its engineers' extensive technical knowledge and experience of the product to produce far more accurate project estimations.

Individual technical requirements, estimated by the engineers assigned to analyze and later implement them, were aggregated to construct feature estimations and ultimately guide project planning and project estimations. The revised REP facilitated a thorough understanding of requirements to aid feature sizing: As one team-lead put it, "time spent during requirements analysis to get fine detail made it possible to have better estimates earlier on." Furthermore, the structured RE document could be used effectively during the change management process. Once schedule allocations were established, careful control of change requests prevented project corrections while ensuring the currency of the original estimations. As one manager said, estimations remained on-target via change management because "the process reduced the number of changes sneaking into the product."

Upfront test scenario definition, requirements validation, and peer-reviews led to *improved feature coverage*. Although testing leads to the prevention of defects, it also plays an important role in assuring that the product contains the required features. Many questionnaire responses and comments recorded during interviews suggest that testing was significantly improved due to the test-scenarios conceived during requirements analysis. For example, one tester indicated that the REP had helped by "creating a starting point for us as to what to test, there was more of a focus on the actual requirements and not just the feature." Likewise, another tester specifically identified the REP's role in improving their testing: "test scenarios provided a good base to start writing test cases with."

Another indicated that test scenarios were used to validate specific test cases: "We could review the test case to see whether it has met the requirements and what we should be testing." Although these comments indicate a direct impact by the requirements process, some believed other processes were, at least in part, responsible for the apparent payoffs. Respondents also cited the peer-review of test cases and the emphasis on testing features according to requirements as factors contributing to assuring feature coverage.

Testing conducted by ACUS toward the end of the project not only served to detect defects before deployment, but also to validate that the baselined features had been implemented in the software product. The anticipation of such testing likely encouraged developers to assure feature coverage throughout the development project. Developers commented that "the use of rationale and test scenarios gave a broader conceptual view of what was required and how we were to demonstrate that we had met the requirement."

Further, peer-review inspections of test cases contributed to the testers' ability to test coverage. Testing was successful

in part because "reviews [were] done with designers and test case authors to validate the 'how' of each test case." The outcomes of these reviews were potentially enlightening to both parties and, according to the respondents, these reviews were successful in revealing issues: "[during] reviews, test specs came under [scrutiny], the resulting issues were sometimes abundant."

Enhanced feature understanding, change management, and project tracking led to *managed requirements creep*. Change management was instrumental in preventing unconstrained requirements creep. The success of ACUS' software control board, an integral part of this achievement, confirms the board's role as a means to control software change [30]. Merely by virtue of implementing a formal change management process, engineers were dissuaded from making discretionary changes. Traceability established within the requirement specifications which were used by project tracking to monitor resources helped to prevent creep from significantly affecting progress.

The structured nature of requirements artifacts helped enable change management, ultimately giving project managers control of requirements creep. Requirements churn, which had been so common in past projects, was controlled by relying on a rigorous requirements change process that limited all but the most critical changes. One engineer reported that the approval process itself was significant: "[it] had a big impact: it made people analyze and think twice about the changes they were considering."

When changes were necessary, change requests were considered in the context of project progress. Respondents indicated that improvements in project tracking that had occurred because of the revised REP enabled "identification of schedule risks," making it "easier to forecast resource crunches." Managers said they could effectively assess change requests and that the management process provided "firm control and visibility."

Change management and feature sizing led to *effective project scope negotiation*. By being able to better estimate resources required to implement proposed features and through careful control of committed-to requirements, ACUS enjoyed far more effective feature negotiations. In this project, the project negotiation was made into a discrete project phase approximately eight months long, in contrast to previous projects when negotiations with external business and marketing units had been conducted, to some degree, throughout the development life-cycle. The requirements process enabled development managers to refine feature requests from the marketing unit into more reasonable chunks and then immediately analyze requests for obvious shortcomings. One team-lead commented that "having requirements done early. It became obvious we could not deliver all of the expected functionality, so we agreed to cut them. Previously, we would not have known until it was too late and then everyone would have to go into a mad rush." This empowered managers to understand requests and provide rigorous justification for rejecting or accepting feature commitments. Meanwhile, customers were also motivated to negotiate in the face of tight change management controls, effective after the requirements phase.

Further, it was understood by both stakeholders at the outset of the project that the revised REP would prevent major features changes after the negotiated requirement set had been committed to. The threat of limited change motivated ACUS to seek detailed information before commitment and motivated marketing to prioritize their needs in light of ACUS' finite capabilities. This new understanding led to "more effective expectation management" or, as one manager put it, "made it much easier to get everyone singing from the same sheet." Change control not only worked in ACUS' favor: The change management process invited participation from the marketing unit who were permitted to comment on change proposals.

Traceability links, peer-reviews, and requirements validation led to *fewer defects*. At ACUS, defect rates appeared to have changed significantly under their revised RE practice. Both internal, predeployment defects (system test bugs) and postdeployment defects showed marked decreases (see Table 1). One engineer, after being informed there had been fewer internal defects, appeared genuinely surprised. His explanation illustrates the long-term ramifications of more accurate estimations/scheduling and hints at the wide-ranging holistic effects of process change: "Wow—I'm impressed if that is the case as there has been much better testing. Process changes have made engineers more aware of how their piece fits in with the rest. I also feel that the schedules were more realistic so there was no temptation to 'bang it in' and move on to the next task—each change was more rigorously unit tested before being integrated." However, one respondent did indicate that defect entry was made more difficult under the new process as requirements engineers had to "... fill in several documents. People are reluctant to enter defects, knowing it will only add to their workload." Notwithstanding this observation, postdeployment defect rates and customer satisfaction suggest that product quality had increased.

In the same way that the peer-review improved testing efficacy, assuring feature coverage, more effective testing also improved defect statistics. When respondents were asked why defects in this project were lower compared to projects in the past, participants suggested that the project had benefited form "improved peer review" and due to "the review process and testing inspections related to requirements, which were done as part of the revised REP."

Common ground and cross-functional teams in feature development led to *more effective communication*. Many engineers praised the apparent improvement in communication during this project compared to the past. In many cases, they attributed these effects directly to the revised requirements process. For example, one engineer noted that the process had helped him by "reducing meetings and promoting good communication to other teams"; another noted that they "felt freer to talk to others." Open and effective dialog which had been engendered during group analysis sessions continued to foster improved communication across the different functional groups. Requirements acted as common ground during development, eliminating superfluous communication that had formerly been necessary to seek clarifications and coordinate among developers. In the revised REP, there was "good communication

with other teams" and "more consistency across functional groups."

Feature decomposition, specification conformance, and team reorganization led to *reduced rework*. The decomposition of features into detailed requirements resulted in specifications which engineers could rely on and conform to. These specifications were now used consistently to reflect an understanding common to members of the cross-functional teams. Engineers make micro-decisions throughout their work that reflect their understanding of customer needs. In the past, confusing unreliable specifications led to rework due to implementation of unwanted features when engineer interpretations diverged from each other.

Basing technical specifications, such as designs or test cases, on more accurate requirements specifications provided consistent and informative direction for engineers. Clarifying claims of reduced rework. One engineer attributed "better designs ... design does get rid of stupidities." Another said that "less functionality was missed due to constant reinforcement of requirements in [specifications]," in effect saving the team from having to refit existing development to add missed features.

Further, the organization of teams into cross-functional units contributed to cross-pollination of ideas and information; on the rework issue, one manager indicated that cross-functional teams provided "more changes to expose people across the product to everything. More eyes on design and code." In clarifying his position on reduced work, one manager indicated that the "involvement across teams had positive effects. In the past we worked in silos."

8 Implications for Research and Practice

This research has a number of important implications for both researchers and practitioners. First, we believe that our case study has made an important step in increasing the understanding of the relationship between requirements engineering theory and industrial practice. Investigating the payoffs of requirements practice and the relationship between RE processes and improvements in productivity, quality, and risk management as the literature suggests has proven to be a difficult but worthwhile task. According to evidence collected during our study, both quantitative project metrics and qualitative engineer perceptions show how requirements engineering practice was perceived as having produced long term benefits that have had an impact throughout the project. Not only were improvements in the REP linked to improved productivity, better quality, and effective risk management, but it also enhanced other development practices such as testing, peer-review, and project tracking. Moreover, the REP appears to have improved collaboration and may have led to cultural change that values cooperation, quality, and customer satisfaction. These findings generate interesting new directions *for research*.

This work should encourage future research to further understand the relationship between REP and improvements in other processes. Our study, in the attempt to understand the particular effects of changes in software processes as a result of the revised RE practice, suffered from the lack of proven instruments for gauging such interactions. Thus, valuable topics for future research include the development of instruments for more objectively assessing and measuring the interactions between REP and other processes such as change management, peer review, and testing. What are the relevant and meaningful criteria to judge improvements in peer reviews, for example, as a result of better RE processes?

We also identified that some of the REP components had more impact on these results and research should focus on investigating them more closely. In particular, those REP components that emphasize collaboration may be critical to producing the successful effects we have documented.

In this direction, to determine whether individual REP components had different effects, we added the REP component scores (scoring described in Section 6.1) across all processes and subprocesses. These combined scores suggest that certain REP improvements dominated in their contributions to other processes. In particular, the component scores show that *traceability* and *decomposition* dominated other components. Surprisingly, *group analysis sessions* and *cross-functional teams* showed by far the lowest scores compared to the other four components. This contrasts with the earlier results which suggested the overwhelming importance of these two components, as presented in Section 5.3. The respondents' opinions appear to have shifted, suggesting the importance of technical outcomes of the RE process: rating traceability, requirements validation, and decomposition very highly.

Perhaps this shift can be attributed to the passage of time between questionnaires. It is possible that respondents to the early questionnaire were more aware of the "soft" components of the REP, while the tangible end results of the process were more memorable in minds of our latest subjects. Furthermore, it is possible that the tangible outputs of the REP were perceived as having the greatest impact on other subprocesses which used those REP outputs as inputs. These difficulties highlight the subtle difference in our data, one that addresses the effect on the engineer's ability to accomplish work, the other that addresses the effect on other processes. Alternatively, it is possible that, during the passage of time between questionnaires, the collaborative habits promoted by the soft REP components were adopted as part of a cultural change within ACUS, making respondents less sensitive to the components' true impact.

However, we believe that the constant that pervaded the progress made at ACUS is collaboration, which emerged as a powerful theme in many of the improvements that have been discussed so far. Compared to the isolated, siloed work culture that had previously dominated, the changes in the new RE process have served to join the organization in a united effort. Reorganizing teams cross-functionally allowed for open lines of communication. Effective interactions between ACUS and their marketing unit produced manageable commitments and realistic expectations. Teams brought together during feature analysis sessions developed the requirements that were so critical later in the project. The requirements themselves provided a basis for communication, coordination, and work. Peer-reviews helped to assure high product quality.

It is not surprising that an improved software development project, dependent on the team building it, should exhibit improved team behavior. If it was possible to separate the effects of human improvement from that of technical improvement, we might see that the greater part of improvement had been due to more effective collaboration throughout the project.

For *software practice*, these insights provide practitioners with further incentives for adopting RE processes and offer more concrete guidance for such initiatives. Our case study provides evidence of how such a process could be used to optimize development activities by tailoring REP improvements to address specific weaknesses in their existing processes.

In this direction, it is encouraging how much progress ACUS was able to achieve despite making relatively inexpensive, straightforward changes to their process. ACUS did not use any sophisticated RE methodologies, instead depending on structured negotiations, engineer estimations, and a well-defined change management procedure.

Perhaps the real challenge is identifying these problem areas. Organizations may not be fully aware of their own dysfunction until they try different approaches and carefully reflect upon their own development practices. We describe here a number of strategies that could establish a foundation for effective customer-supplier cooperation and help promote mutual expectations while encouraging the development and validation of accurate requirements:

Iterative Feature Decomposition. Committing to high-level features from customers without consideration from the development organization is likely an invitation for unrestrained feature creep and missed deadlines. Instead, both stakeholders should understand that provision of high-level features is only the beginning. A process should promote systematic feature analysis and decomposition by the development team, who can then seek clarifications from and provide effort estimations to marketing. This is an iterative processes that empowers both stakeholders to reach an agreeable outcome despite their geographical separation.

Frequent Negotiation Sessions. Understanding the needs of the customer and the limits of the supplier are prerequisites to reaching an agreement. A requirements process that facilitates frequent negotiations among remote stakeholders is an effective way to overcome mistrust and power struggles. Rigorous effort estimations can empower the development organization to inform requirement prioritization to align development capability with corporate strategy and market needs.

Change Management. Change is inevitable, but process can assure that change affects the project in a controlled manner. A process can motivate stakeholders to get appropriate requirements established by specifying an approval process which takes effect when the project scope is agreed to and baselined. Requiring stakeholder approval, irrespective of location, ensures that all parties will remain fully informed of requirements change.

Early test scenarios. Test scenarios are necessary for conducting system testing and requirements validation over the development period. But, conceiving test scenarios as early as possible during the development and analysis of requirements assures that the final realization of the requirements is considered early. This practice can expose developers to validation issues and testers to development issues, helping to flush out potential problems earlier, leading to high quality software.

Cross-functional requirements analysis sessions. Analyzing requirements is a necessary component of any requirements process. Conducting such an analysis as a group helps to ensure sufficient exploration of ideas. Attendance by representatives from major functional departments establishes a common understanding among groups, helps break down social barriers, and ensures that requirements are considered from a variety of perspectives. Cooperation and collaboration within the organization promotes a productive team environment.

These strategies represent a small but important contribution toward providing detailed advice to practitioners. By providing supporting empirical evidence, these strategies are framed by the context in which they are known to have been successful. One hopes that this comprehensive coupling of strategy, evidence, and contact provides effective, adoptable approaches to improving software risk management, quality, and productivity.

9 Threats to Validity

Any research results grounded in empirical study need to be understood within the strengths and limitations of the particular research methodology. Whereas case study research is invaluable in obtaining the rich insights presented in this paper, it inevitably suffers from threats to the validity of its results. Three types of validity are discussed in the next sections: construct validity and internal and external validity, respectively.

9.1 Construct Validity

Construct validity refers to the degree to which inferences can legitimately be made from the operationalizations in a study to the theoretical constructs on which those operationalizations were based [34]. Constructs are chosen as "labels" of attributes of people or aspects in the phenomenon under study, based on theoretical considerations, and the important questions are whether the constructs are appropriately chosen so as to measure what is intended to be measured. Intentional and representational validity of constructs are discussed next.

Intentional validity: Do the constructs we chose capture what we intended to study?

The goal of our research was to study improvements in the organization as a result of improvements in the RE process. In our exploratory and explanatory case study, we used reports from the literature to guide our observations of improvements and chose, as discussed in Section 2, to look for expected benefits in the areas of developer productivity, software quality, and risk management. To operationalize these concepts, we chose constructs that we learned from the literature, for example, decreases in rework for productivity, decreases in the number of defects for quality, and increases in estimation ability for risk management,

respectively. In addition to guidance from the literature, we also considered the particular context of our subject organization. We specifically investigated what management and engineers at ACUS would consider as improvements in productivity, quality, and risk management. To this end, early interviews with management and developers led us to choose the constructs outlined in Table 1 (second column) to operationalize productivity, quality, and risk management. Furthermore, in some cases, we could identify subconstructs for these constructs: for example, that problem understanding depended on the engineers' ability to reveal details, dependencies, and complexities of features, as discussed in Section 5.2.1. Similarly, we found that improvements in communication were not limited to "more effective communication," but also reduced superfluous communication. Further, increased feature coverage, decreased feature creep, and higher quality project negotiations were all perceived as measures of improved risk management.

Representation validity. Do the constructs we chose translate well into observable phenomena?

This was the first time the RE process was rigorously defined at ACUS and historical data was very limited. Where possible, this research endeavored to collect and analyze quantitative data. For example, data was collected on project estimation, where estimations were collected before and after the RE process was revised. We were also able to consider product quality in terms of support requests and postdeployment defects, both of which were carefully measured in both this project and in past projects. For many aspects, however, no such data had ever been collected. Instead, we relied on the extensive experience of the ACUS engineers and managers to assess the impact of the REP improvements and to provide comparison to previous practice. Our dependence on the respondents' perception and recollection is undeniable although unavoidable. However, this is not unusual: Organizations that seek to implement RE process improvements typically lack quantitative data concerning their existing RE practice [20]. Moreover, practitioner opinion has value as it is a reflection of the satisfaction of process implementation and indicates the extent to which the process will be adopted. RE process improvements depend on the adoption of new processes to work. Such adoption has to happen at the individual level before it will be apparent in the project or organization [20]. Clearly then, perceptions among practitioners is a key issue, as are the operationalizations based on those perceptions.

Likert and Likert [23] explain the significance of perceptions:

> People act on the basis of what they perceive the situation to be, whether the perceptions are accurate or grossly inaccurate. Since behavior is based on perceptions, the existence of each of them is a fact to be considered. Similarly, the frustrations, attitudes, loyalties, and hostilities felt by each member and the information and misinformation possessed by each are facts as is their evaluation of the merits and desirability of each particular course of action under consideration.

9.2 Internal Validity

Whereas a discussion about internal validity is most often made in experimental design, it is important in case study research whenever inferences of the cause and effect relationship are made. Thus, internal validity refers to the degree to which we are successful in eliminating confounding variables within the study itself and is concerned with the question:

Are the values of the dependent variable solely the result of the manipulations of the independent variable?

The results of our case study suggest that that significant improvements in the organization were related to improvements in the RE practice and the improvements in the organization can be regarded as changes in the dependent variable (state of practice in the software organization) as a result of changes in the RE practice (independent variable). In critically assessing our results, we list here three other possible alternative theories to the observed changes in the dependent variable (software practice at ACUS), which can be thought of as confounding variables. First, it is possible that new management at ACUS, a change which occurred just prior to the software process improvement, was in fact responsible for the majority of changes at ACUS, rather than any particular process. There is no question that the change in management at ACUS, namely, of their project and product manager at the beginning of the project before the software process improvement initiative had started, was an important factor of success at ACUS. Strong management support and the credible promise of change that comes with new management was most likely instrumental in ensuring that the new requirements process was adopted by engineers, as found in another study of REP improvement [20]. But, although new management helped to assure process adoption, there is no reason to believe these new managers could assure process efficacy. In fact, this is evident from the difficulties ACUS encountered during their subsequent project in which they changed their processes again with a result that was far less successful.

A second possibility, implied in the comments of some engineers, is that evident payoffs were due to the product's maturity, rather than improvements in process. The questioning of engineers occasionally elicited obviously skeptical responses. Of these responses, many identified product maturity as playing a major role in producing the payoffs in question. For example, there were suggestions that this project had fewer defects than previous projects because, during past projects, inherent defects in the underlying architecture had been addressed. However, this project was the third point release of their product since its last major revision. Selected managers were specifically asked to compare the significance of this project compared to previous projects and, by all accounts, this project represented a comparable degree of feature additions and complexity. The consistency in defect rates between the first and second point releases indicated limited variability between at least the first two point releases, suggesting relatively constant defect insertion and removal rates. To accept this theory, however, would require dismissing all other engineer responses that strongly attribute much of ACUS' success to the revised requirements process. In the end, it is impossible to completely discount the effect of code maturity without significant analysis of code changes between previous versions.

Third, it is possible that other process changes, unrelated to requirements or change management, could have been accountable for the positive changes. Although

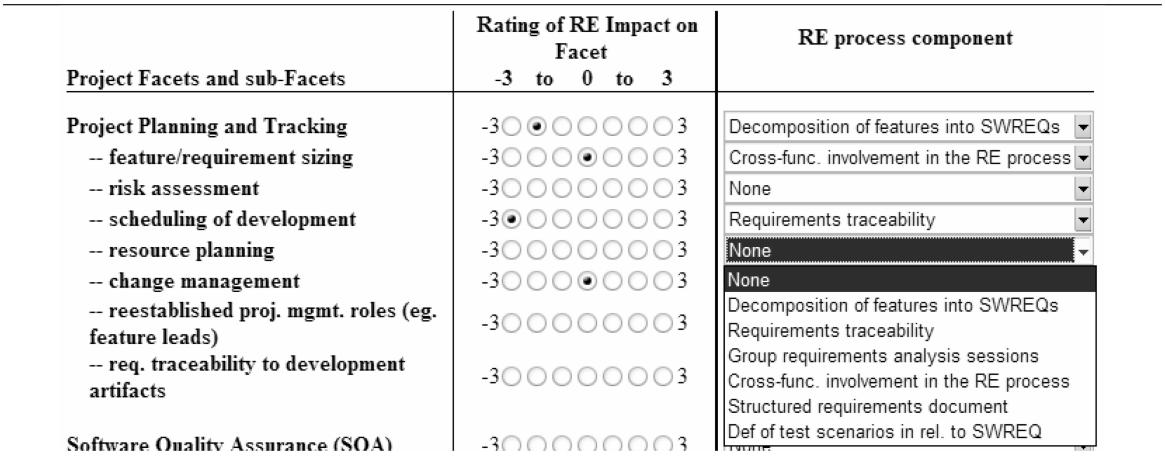

Fig. 6. Snapshot of the Web-questionnaire used in stage 3 of the case study.

requirements management had been identified as the most deficient of ACUS' process area, ACUS did make revisions to some of its other processes. However, an inspection of the process documents suggests that these changes pale in comparison to the far more sweeping changes made related to requirements engineering. Some changes only required documentation of what had formerly been without record, for example, many project management processes were altered to require explicit status documentation and report production. In other cases, many relatively minor process changes were made to support the revised requirements process. For example, the formulation of the software change control board was made a subprocess of the SCM process area; however, the board's purpose was to officiate change requests made necessary by the revised requirements process.

And, fourth, the Hawthorne or placebo effect may have affected engineer opinion. We believe, however, that the respondents, knowledgeable about ACUS processes over the last 15 years (in particular, managers), were critical of the process change. Not only had the project members shown resistance to the revision of the RE process at ACUS [6], but interview data after project completion indicates the respondents' ability to critically reflect on their experiences with the process change. Exposure to subsequent projects in which the requirements processes varied again appears to have given the respondents further perspective and insight about the effects of the RE practice and process interactions discussed here.

9.3 External Validity

Do the results of this study generalize beyond the organization studied?

In addressing this issue, we believe that ACUS is representative of a fairly large class of software development organizations. Software companies that we expect to benefit from similar results on process improvement when following the RE process described here would have similar characteristics to ACUS, as follows:

First, ACUS staff is at about 150 members, working in small teams led by a team lead and technical managers, all headed by a project manager and a product manager. In obtaining information on product features, ACUS maintains a strong relationship with the corporate marketing department, who plays the role of customers. This customer relationship is similar to organizations that receive requirements and funding from separate or internal corporate entities as well as organizations that build custom solutions for a single customer. ACUS's software itself forms the basis of a mature product line that is highly customizable. Overall, we believe that these attributes are very characteristic of many other software development organizations.

Second, the challenges experienced by ACUS can be summarized by: ineffective customer negotiations, teams within the organization who operated autonomously and independently of each other, and requirements that were routinely revised without regard to the impact on development. Many companies that would be assessed at CMM level 1 would, by definition, be enduring similar difficulties. In other words, the evidence collected from ACUS is clearly relevant to organizations which are burdened by similar concerns and wish to improve their internal communication, customer negotiations, change control, and quality control in testing and inspections.

In contrast, companies that are considerably smaller may not suffer from internal communication issues merely by virtue of their compact size. This may in part explain why rigorous RE processes do not appear to produce beneficial effects as readily in small organizations as in large. For example, El-Emam and Birk [8] provide evidence that REP maturity correlates with an organization's software quality; however, the relationship is only true for medium to large organizations.

Finally, requirements management and control comes at a price, costing added overhead in considering change requests and dampening the software's rate of change. This is a desirable outcome for an established software product where features can be included in one version or another. But, for a from-scratch speculative software project, such demands may suffocate the development necessary to produce an innovative new solution.

In summary, these results are likely very applicable to industrial developers who share many of the same organizational and software characteristics and who also suffer from some of the same problems that ACUS does. Of course, it may appear self-evident that the solution to the problem is generalizable to others with the same problem;

TABLE 4
Perceived Impact of the REP on the Other Processes and Their Constituent Subprocesses

	Number of responses in each impact category						
Scale for REP impact on each process and constituent sub-processes	**-3**	**-2**	**-1**	**0**	**1**	**2**	**3**
Project Planning	0	0	0	1	4	5	0
feature sizing	0	0	1	1	2	4	4
risk management	0	0	2	4	3	4	0
scheduling	0	0	0	4	3	6	0
resource planning	0	0	0	4	3	6	0
change mgmt	0	0	0	4	3	4	2
responsibility allocation	0	0	0	4	4	3	0
requirements tracing	0	0	0	0	4	6	3
SQA	0	0	0	1	6	3	0
tracking	0	0	0	2	2	5	2
SQA teams	0	0	1	10	0	1	0
meeting and reports	0	0	0	7	3	1	0
deviations	0	0	0	6	2	3	0
SCM	0	0	0	4	6	1	0
baselining	0	0	0	4	4	3	0
change mgmt	0	0	0	4	4	2	1
code change metrics	0	1	0	6	1	2	1
role adherence	0	0	0	7	2	2	0
software change control board	0	0	0	8	3	0	0
SCM tools	0	0	0	5	4	2	0
Dev	0	0	0	2	2	6	1
Team reorganization	0	0	0	3	4	6	0
split leads	0	0	1	5	4	3	0
planning	0	0	0	7	1	5	0
Specification conformance	0	0	0	3	4	4	2
code inspection	0	0	0	8	3	1	0
Testing	0	0	0	1	4	4	2
automation	0	1	0	3	0	3	1
requirements validation	0	0	0	0	5	5	3
peer-review	0	0	0	2	3	5	3
repeated test	0	0	1	6	3	2	1

the challenge is identifying the problem. Companies may not be fully aware of their own dysfunction until they try different approaches and carefully reflect upon their own development practices. It is in this direction that our research attempted to unveil the rich and beneficial interaction between RE and other processes in one particular organization, while making the best effort to describe the details of the RE practice as well as the organization such that practitioners are able to make decisions of how to use these insights in their own organization.

10 Conclusions

While challenging, the empirical study into the practice of requirements engineering at ACUS has proven very beneficial in unveiling details that are much needed though lacking in our software engineering research and practice. To add to the inherent difficulties of field research, our study had to overcome the challenges of a lack of a well-defined theoretical base in the area of process interactions and ways of operationalizing the aspects under study. It is our belief that our systematic study has appropriately addressed the threats to validity and our results will benefit both research and practice of software development.

Appendix A

A snapshot of the Web questionnaire used in stage 3 of the case study is shown in Fig. 6. Each line indicates a process and its constituent subprocesses. The full Web form and instructions for completion can be found at http://vigilant.segal.uvic.ca/acus/.

Appendix B

Description of the development processes and its constituent subprocesses. The following provides a description of each process area and descriptions for each of the area's constituent subprocesses. Some subprocesses are followed

TABLE 5
Results of the Chi-Square Test

PROCESS	Constituent sub-processes						
PROJECT PLANNING	Feature sizing	Risk mgmt	Schedul-ing	Resource planning	change mgmt	Responsi-bility alloca-tion	Reqts tracing
p=0.000528	**0.000757**	0.202396	**0.014228**	**0.014228**	**0.014228**	0.069954	**0.000002**
SQA	Tracking	SQA teams	Meetings and re-ports	Deviations			
0.000528	**0.001992**	0.125786	0.915106	0.3594032			
SCM	Baselining	Code change mgmt	Change metrics	Role adherence	S/w change control board	SCM tools	
0.069954	0.069954	0.069954	0.915106	0.915196	0.915196	0.240955	
DEVELOP-MENT	Team re-organiza-tion	Split leads	Planning	Specifica-tion con-formance	Code in-spection		
0.001992	**0.002367**	0.202396	0.492457	**0.002367**	0.759463		
TESTING	Automa-tion	Require-ments valida-tion	Peer review	Repeated			
0.000191	0.531971	**0.000002**	**0.000285**	0.492457			

The highlighted values indicate statistical significance at p<0.05.

by an abbreviated short name in brackets; this short name is used in the charts and tables provided in Section 4.

Project Planning and Tracking: This process area is largely a managerial responsibility wherein the software project is initially planned, schedules formulated, resources allocated, and milestones determined. Once the project is under way, subsequent tracking and monitoring of the project progress also falls within this process area. Seven subprocesses of project planning and tracking were identified:

- **Feature sizing:** Estimations of required effort to design, test, document, and implement features or requirements.
- **Risk assessment:** Determination of the technical risk of implementing particular components of the software system.
- **Scheduling:** Planning the length of time for specific project phases to complete.
- **Resource planning**: Allocating developers to specific features and project roles.
- **Change management:** Review and control of change requests from developers.
- **Responsibility allocation:** Assigning lead roles for the implementation of particular features.
- **Requirements tracing:** Ensuring that traceability links between requirements and other design artifacts are maintained.

Software Quality Assurance: This process area's role is to track software design and development, specifically with respect to ensuring a high standard of quality is maintained. Four subprocesses were identified:

- **Tracking:** Monitoring the progress of implementation and feature testing.
- **SQA team (teams):** The formation of a team responsible for SQA.
- **Meetings and reports (meetings):** Regular meetings and reports are conducted and produced by the SQA team regarding software quality during development.
- **Deviations:** Review and control of process and major project deviations during development.

Software Configuration Management: This process area's role is to maintain consistency of project artifacts and, in particular, to assure effective software configuration management (SCM). Artifacts subject to management not only include source code artifacts, but all formal project documents, including project plans, reports, designs, test cases, etc.

- **Baselining:** Determine base software versions and milestone feature sets.
- **Code Change management:** Review and control of change requests from developers.
- **Change metrics (metrics):** Provide rudimentary measure and extent of change in code artifacts.
- **Role adherence (roles):** Monitor and ensure that development roles are being fulfilled, particularly with respect to those responsible for the SCCB (see below) and managerial issues.
- **Software change control board (SCCB):** This group of people was created to review, discuss, and approve developer change requests.
- **SCM tools:** Prescribe the use of a tool or set of tools to manage version control on intermediate development artifacts such as documents and source code.

Development: This process area constitutes all aspects concerning the design and implementation of the software product, but does not include documentation or testing, which are both addressed separately.

- **Team reorganization (teams):** Development staff were reorganized into teams responsible for implementation of requirements.

TABLE 6
Raw Responses on the REP Component that Contributed Most to the REP Impact on the Process and Its Associated Subprocesses

	Decomposition	Traceability	Group Sessions	Cross functional teams	Structured Req document	Testing According to requirements
Project Planning[7]	2	1	0	0	1	0
(1.29) feature sizing	7	0	3	0	0	1
(1.11) risk mgmt	0	1	3	2	2	0
(0.90) scheduling	5	0	0	1	2	0
(0.90) resource planning	2	1	1	2	2	0
(1.11) change mgmt	1	2	0	0	4	0
(0.83) responsibility allocation	1	1	1	4	2	0
(0.76) requirements tracing	0	9	1	1	1	0
SQA	0	2	0	0	0	1
(1.03) tracking	0	4	0	0	2	0
(0.67) SQA teams	0	1	0	0	0	0
(0.69) meeting and reports	1	0	0	1	0	0
(0.90) deviations	0	2	0	0	1	0
SCM	1	2	0	0	1	0
(0.83) baselining	2	1	0	0	1	0
(1.00) change mgmt	1	2	0	0	3	0
(1.37) code change metrics	0	0	0	0	1	0
(0.82) role adherence	1	0	0	0	1	0
(0.47) sw change ctrl board	0	1	0	0	0	0
(0.79) SCM tools	0	1	0	0	1	0
Development	2	0	1	1	0	0
(0.83) team reorganization	0	0	1	5	1	0
(0.95) split leads	1	1	0	3	0	0
(0.99) planning	3	1	0	1	0	0
(1.04) specification conformance	2	3	0	0	1	0
(0.67) code inspection	0	1	0	0	0	1
Testing	0	1	0	0	1	4
(1.64) automation	0	0	0	0	0	4
(0.80) requirements validation	0	2	0	0	0	8
(1.03) peer review	1	5	0	1	0	2
(1.11) repeated test	0	2	0	0	0	2

- **Split leads:** Organizationally, lead development responsibility is shared by two separate individuals, one responsible for managerial issues and one responsible for technical issues.
- **Planning:** Emphasize and follow plans used to direct software development.
- **Specification Conformance:** Software developers and designers are expected to follow and be guided by feature proposals, requirements specifications, and design specification.
- **Code Inspection:** Development artifacts, such as source code, are inspected for defects.

Testing: Processes related to testing are responsible for the development test scenarios, test plans, and test execution to ensure quality and validate software products.

- **Automation:** Some testing was conducted in an automated fashion.
- **Requirements Validation (validation):** Test scenarios and detailed test cases were written against requirements to validate promised functionality.
- **Peer-Review:** Test artifacts such as test scenarios and tests themselves were peer reviewed.
- **Repeated Test (repeated):** Many tests were repeatedly conducted to validate the maintenance of functionality (i.e., regression testing).

APPENDIX C

The perceived impact of the REP on the other processes and their consituent subprocesses is shown in Table 4.

APPENDIX D

The results of the chi-square test are shown in Table 5.

APPENDIX E

The raw responses on the REP component that contributed most to the REP impact on the process and its associate subprocesses are shown in Table 6 and the weighted/scored responses on the REP component that contributed most to the REP impact on the process and its associated subprocesses are shown in Table 7.

TABLE 7
Weighted/Scored Responses on the REP Component that Contributed Most to the REP Impact on the Process and Its Associated Subprocesses

	Decomposition	Traceability	Group Sessions	Cross functional teams	Structured Req document	Testing According to requirements
Project Planning	4	1	0	0	2	0
(1.29) feature sizing	17	0	5	0	0	1
(1.11) risk mgmt	0	1	5	2	3	0
(0.90) scheduling	7	0	0	2	4	0
(0.90) resource planning	3	1	2	4	3	0
(1.11) change mgmt	2	3	0	0	8	0
(0.83) responsibility allocation	1	1	2	5	3	0
(0.76) requirements tracing	0	17	1	2	3	0
SQA	0	3	0	0	0	1
(1.03) tracking	0	9	0	0	5	0
(0.67) SQA teams	0	2	0	0	0	0
(0.69) meetings and reports	1	0	0	2	0	0
(0.90) deviations	0	3	0	0	2	0
SCM	1	3	0	0	1	0
(0.83) baselining	3	2	0	0	1	0
(1.00) code change mgmt	1	2	0	0	6	0
(1.37) change metrics	0	0	0	0	3	0
(0.82) role adherence	1	0	0	0	2	0
(0.47) sw change ctrl board	0	1	0	0	0	0
(0.79) SCM tools	0	1	0	0	2	0
Development	5	0	2	2	0	0
(0.83) team reorganization	0	0	2	8	2	0
(0.95) split leads	2	1	0	4	0	0
(0.99) planning	5	2	0	2	0	0
(1.04) specification conformance	4	4	0	0	2	0
(0.67) code inspection	0	2	0	0	0	1
Testing	0	2	0	0	2	9
(1.64) automation	0	0	0	0	0	7
(0.80) requirements validation	0	3	0	0	0	16
(1.03) peer review	3	11	0	2	0	3
(1.11) repeated test	0	3	0	0	0	4

ACKNOWLEDGMENTS

The authors thank the ACUS participants in the study, as well as Janice Singer, Stuart Faulk, Dan Berry, Susan Sim, Brian Gaines, Jim Brosseau, Elizabeth Hargreaves, and Adrian Damian for comments on drafts of the paper, as well as the anonymous reviewers who provided feedback on earlier version of this paper. Thank you also to Carolyn Campbell for her editorial assistance.

REFERENCES

[1] D. Berry, D. Damian, A. Finkelstein, D. Gause, R. Hall, and A. Wassyng, "To Do or Not to Do: If the Requirements Engineering Payoff Is So Good, Why Aren't More Companies Doing It?" *Proc. Requirements Eng.*, 2005.

[2] A. Borjesson and L. Mathiassen, "Successful Process Implementation," *IEEE Software*, pp. 35-44, July/Aug. 2004.

[3] J. Broadman and D. Johnson, "Return on Investment from Software Process Improvement as Measured by U.S. Industry," *Crosstalk*, vol. 9, no. 4, pp. 23-29, 1996.

[4] F. Brooks, "No Silver Bullet: Essence and Accidents of Software Engineering," *Computer*, vol. 20, no. 4, pp. 10-19, Apr. 1987.

[5] C. Claus, M. Freund, M. Kaiser, and R. Kneuper, "Implementing Systematic Requirements Management in a Large Software Development Programme," *Proc. Fifth Int'l Workshop Requirements Eng.: Foundation of Software Quality*, pp. 33-42, 1999.

[6] D. Damian, D. Zowghi, L. Vaidyanathasamy, and Y. Pal, "An Industrial Case Study of Immediate Benefits of Requirements Engineering Process Improvement at the Australian Center for Unisys Software," *Int'l J. Empirical Software Eng.*, vol. 9, nos. 1-2, pp. 45-75, Mar. 2004.

[7] D. Damian, J. Chisan, L. Vaidyanathasamy, and Y. Pal, "Requirements Engineering and Downstream Software Development: Findings from a Case Study," *Int'l J. Empirical Software Eng.*, vol. 10, no. 3, 2005.

[8] K.E. El-Emam and A. Birk, "Validating the ISO/IEC 15504 Measure of Software Requirements Analysis Process Capability," *IEEE Trans. Software Eng.*, vol. 26, no. 6, pp. 541-566, Nov./Dec. 2000.

[9] K. Fosberg and H. Mooz, "System Engineering Overview," *Software Requirements Eng.*, R. Thayer and M. Dorfman, eds., pp. 44-72, 2000.

[10] A.F. Hutchings and S.T. Knox, "Creating Products: Customers Demand," *Comm. ACM*, vol. 38, no. 5, pp. 72-80, 1995.

[11] T. Hall, S. Beecham, and A. Rainer, "Requirements Problems in Twelve Software Companies: An Empirical Analysis," *IEE Proc.—Software*, vol. 149, no. 5, 2002.

[12] J. Herbsleb and D. Goldenson, "A Systematic Survey of CMM Experience and Results," *Proc. Int'l Conf. Software Eng.*, pp. 323-330, 1996.

[13] W. Humphrey, T. Snyder, and R. Willis, "Software Process Improvement at Hughes Aircraft," *IEEE Software,* vol. 8, no. 4, pp. 11-23, 1991.
[14] ISO/IEC TR 15504-1:1998, *Information Technology—Software Process Assessment,* ISO, 1998.
[15] C. Jones, "Strategies for Managing Requirements Creep," *Computer,* vol. 29, no. 6, pp. 92-94, June 1996.
[16] S. Jacobs, "Introducing Measurable Quality Requirements: A Case Study," *Proc. Fourth Int'l Symp. Requirements Eng.,* pp. 172-179, 1999.
[17] M. Jirotka and J. Goguen, *Requirements Engineering: Social and Technical Issues.* Academic Press, 1994.
[18] M. Kauppinen and S. Kujala, "Starting Improvement of Requirements Engineering Processes: An Experience Report," *Proc. Third Int'l Conf. Product Focused Software Process Improvement (Profes),* pp. 196-209, 2001.
[19] M. Kauppinen, S. Kujala, T. Aaltio, and L. Lehtola, "Introducing Requirements Engineering: How to Make a Cultural Change Happen in Practice," *Proc. IEEE Joint Int'l Requirements Eng. Conf. (RE '02),* pp. 43-51, 2002.
[20] M. Kauppinen, M. Vartiainen, J. Kontio, S. Kujala, and R. Sulonen, "Implementing Requirements Engineering Processes throughout Organizations: Success Factors and Challenges," *Information and Software Technology,* vol. 46, no. 14, pp. 937-953, 2004.
[21] H. Kaindl, S. Brinkkemper, J.A. Bunenko, B. Farbey, S. Greenspan, C. Heitmeyer, J.C. Leite, N. Mead, J. Mylopolous, and J. Siddiqi, "Requirements Engineering and Technology Transfer: Obstacles, Incentives and an Improvement Agenda," *Requirements Eng. J.,* vol. 7, pp. 113-123, 2002.
[22] S. Lauesen and O. Vinter, "Preventing Requirement Defects: An Experiment in Process Improvement," *Requirements Eng. J.,* vol. 6, pp. 37-50, 2001.
[23] R. Likert and J.G. Likert, *New Ways of Managing Conflict,* p. 165. New York: McGraw-Hill, 1976.
[24] K.R. Linberg, "Software Developer Perceptions about Sofware Project Failure: A Case Study," *Systems and Software,* vol. 49, pp. 177-192, 1999.
[25] E. Mayo, *The Human Problems of an Industrial Civilization.* New York: Macmillan Co., 1933.
[26] B.A. Nuseibeh and S.M. Easterbrook, "Requirements Engineering: A Roadmap," *The Future of Software Eng.,* A.C.W. Finkelstein, ed., IEEE CS Press, 2000.
[27] M. Paulk, "A Comparison of ISO 9001 and the Capability Maturity Model for Software," CMU/SEI-94-TR-12, 1994.
[28] J. Procaccino, J. Verner, S. Overmyer, and M. Darter, "Case Study: Factors for Early Prediction of Software Development Success," *Information and Software Technology,* vol. 44, pp. 53-62, 2002.
[29] B. Regnell, P. Beremark, and O. Eklundh, "A Market-Driven Requirements Engineering Process—Results from an Industrial Process Improvement Programme," *Requirements Eng. J.,* vol. 3, no. 2, pp. 121-129, 1998.
[30] S. Robertson and J. Robertson, *Mastering the Requirements Process.* Addison-Wesley, 1999.
[31] SEI, 1995: Software Eng. Inst., *The Capability Maturity Model: Guidelines for Improving the Software Process.* Addison Wesley, 1995.
[32] I. Sommerville and P. Sawyer, *Requirements Engineering: A Good Practice Guide.* John Wiley & Sons, 1997.
[33] T.S. Group, "The CHAOS Report," The Standish Group International, 2003, www.standishgroup.com.
[34] W. Trochim, "Research Methods Knowledge Base," http://www.socialresearchmethods.net/kb/, 17 May 2006.
[35] J.A. Villanlon, G.C. Augustin, T.G. San Feliu, and A.A. Seco, "Experiences in the Application of Software Process Improvement in SMEs," *Software Quality J.,* vol. 10, no. 3, pp. 261-273, 2002.
[36] H. Wohlwend and S. Rosenbaum, "Software Process Improvement in an International Company," *Proc. Int'l Conf. Software Eng.,* pp. 212-220, 1993.
[37] R.K. Yin, *Case Study Research: Design and Methods.* Thousand Oaks, Calif.: Sage Publications, 1994.

Daniela Damian is an assistant professor at the University of Victoria, Canada, where she leads research at SEGAL, the Software Engineering Global interAction Laboratory. Her research lies at the intersection of software engineering, computer supported cooperative work, and human computer interaction. Her current projects are in the areas of requirements and software processes in geographically distributed teams and the development of collaborative tools to support global software teams. She has recently acted as a guest editor of an *IEEE Software* special issue on global software development and as program cochair for the First International Conference on Global Software Engineering.

James Chisan received the MS degree in computer science in 2005 from the University of Victoria and the BS degree in computer science in 2001 from the University of Calgary. His fields of interest are requirements engineering, software process improvement, and global software development. He has worked as a software developer for IBM's Database Technologies group and as a senior analyst for HSBC Bank. Currently, he is practicing intellectual property law as a technology specialist at Fish & Richardson.