Empirical Software Engineering, 10, 255–283, 2005.

Requirements Engineering and Downstream Software Development: Findings from a Case Study

DANIELA DAMIAN DanielaD@uvic.ca
Department of Computer Science, University of Victoria, P.O. Box 3055, Victoria, BC V8W 3P6, Canada

JAMES CHISAN Chisan@uvic.ca
Department of Computer Science, University of Victoria, P.O. Box 3055, Victoria, BC V8W 3P6, Canada

LAKSHMINARAYANAN VAIDYANATHASAMY Lnv.Samy@unisys.com
Unisys Australia Limited, 1C Homebush Bay Drive, Rhodes, NSW 2138, Australia

YOGENDRA PAL Yogendra.Pal@unisys.com
Unisys Australia Limited, 1C Homebush Bay Drive, Rhodes, NSW 2138, Australia

Editor: Lionel Briand

Abstract. Requirements management is being recognized as one of the most important albeit difficult phases in software engineering. The literature repeatedly cites the role of well-defined requirements and requirements management process in problem analysis and project management as benefiting software development throughout the life cycle: during design, coding, testing, maintenance and documentation of software.

This paper reports on the findings of an investigation into industrial practice of requirements management process improvement and its positive effects on downstream software development. The evidence reveals a strong relationship between a well-defined requirements process and increased developer productivity, improved project planning through better estimations and enhanced ability for stakeholders to negotiate project scope. These results are important since there is little empirical evidence of the actual benefits of sound requirements practice, in spite of the plethora of claims in the literature. An account of these effects not only adds to our understanding of good requirements practice but also provides strong motivation for software organizations to develop programs for improvement of their requirements processes.

Keywords: Empirical studies of software process, industrial experience in process improvement, requirements management and practice, software productivity.

1. Introduction

Improving the software process to improve overall software development has been an on-going endeavor for both industrial practitioners and academics for many years. In software engineering, and in particular the software process area, issues relating to requirements engineering (RE) have been repeatedly cited. Requirements specification is regarded as a critical stage of software development, with the claim that software development problems could be better addressed with "good" RE practice (Brooks, 1987; Curtis et al., 1988; Sommerville, 1996).

While software engineering is benefiting from the development of models and standards for software process improvement and assessment such as the ISO 9001 standard for quality management systems (Quality Standards, 1987), and the Software Engineering Institute's Capability Maturity Model for Software (CMM) (CMM, 1991; Paulk et al., 1993; SEI, 1995), the relatively new field of RE does not enjoy well-established, proven strategies for improving or assessing the requirements process. This leads to a strong need for assessment and measurement of effects of rigorous RE practice in software development. Many 'practical guides' naturally focus on the RE process within the larger software development process, but deliberately present their material vaguely, reminding practitioners that good requirements engineering [process] depends on the organization, its development process, its tools and particular circumstances (Sommerville and Sawyer, 1997), or even that '*every project needs a different process*' (Robertson and Robertson, 1999). These sentiments offer little comfort to the practitioner. These same guides warn that revolutionary change is not practical; that instead careful evolutionary improvements are more fruitful. Practitioners are encouraged to measure results to gauge effectiveness, but topics on empirical assessment are largely left to the imagination, as an exercise for the reader. Further, the role of requirements engineering in software development has been discussed in the literature as important in planning activities such as: determining the nature of the problem, exploring solutions through feasibility studies, and ultimately 'deciding precisely what to build.' Brooks (1987) further notes the opportunity for RE to improve all subsequent stages of the development life-cycle, ultimately leading to broader improvements in software quality and user satisfaction.

Industrial case studies that investigate the role and effects of "good" RE process and practice in improving software development are thus very important in providing evidence that is useful to practitioners and furthering research in this area. In this paper, we present findings from a case study of a software development organization that is undergoing process improvement with particular focus on improving the requirements engineering activities. The goal of the case study was to observe the improvements in the RE process and practice, and to empirically assess the impact on the product software downstream development. The organization's software process improvement was investigated over a period of 18 months (over the full life-cycle of a project) and the benefits of the improvement with a focus on RE process are reported. The findings in this paper complement the evidence we collected in an earlier assessment of the same RE practice (6 months after process improvement was initiated), evidence to which we referred to as "immediate benefits" at the ISESE 2002 conference (Damian et al., 2002) and which was published in the Journal of Empirical Software Engineering (Damian et al., 2004).

We report here the findings of the second part of the case study, which examined what we refer to as long-term benefits of improved requirements process throughout the software development life cycle, during design, coding, testing and documentation of software. The evidence reveals a strong relationship between a well-defined requirements process and increased developer productivity, improved project planning through better estimations, as well as an enhanced ability for stakeholders to negotiate project scope. These results are important since there is little empirical evidence of the actual benefits of sound requirements practice, in spite of the claims in the literature (e.g. Brooks, 1987; Curtis et al., 1988; Sommerville and Sawyer, 1997; The Chaos report, 1997). An account

of these effects not only adds to our understanding of good requirements practice but also provides strong motivation for software organizations to develop programs for improvement of their requirements processes.

The paper is structured as follows. Background information on the case study describes the company, its challenges in requirements practice and its process improvement initiative (Section 3.1). This is followed by a description of our empirical investigation into the role of "good RE practice" in Section 3.2, the research method (Section 3.3) and a detailed account of the aspects that we investigated in Section 3.3.2. The findings on each of these aspects are briefly outlined in Section 4 and discussed in detail in Section 5. Furthermore, an important note is that the RE process improvements at ACUS occurred in parallel with initiatives for improvement in other processes in the organization, such as software quality assurance, project planning and project tracking. The possibility of limitations in our study as a result of the interaction between the requirements engineering process and other processes is discussed in Section 7, where we outline our plan for future research to overcome these limitations. The paper concludes by presenting limitations of the case study and avenues for future research.

2. Related Work

A review of existing literature on requirements engineering and software process improvement (SPI) reveal several models to incrementally improve requirements engineering process and the purported benefits reaped by such improvements. Three predominant improvement models are the CMM, ISO/IEC 15504 and the process capability model developed by Sommerville and Sawyer (1997). These models all describe (or imply) increases in productivity, enhanced quality, improved risk management and other benefits to the software development process.

The Capability Maturity Model (CMM), developed by the Software Engineering Institute (SEI), defines a key process area (KPA) that specifically addresses requirements management. Its goals are to establish a baseline for software engineering and management used to guide software plans, products and activities that are consistent with system requirements. Paulk et al. (1993), describing CMM, cite hypothesized benefits in: (1) productivity: cost decrease, shorter development time, and increased quality; and (2) project performance management: more accurate, less variable project performance forecasts. The International Standards Organization (ISO) also provides the ISO/IEC 15504, a standard that describes therein the "Develop software requirements" process. This process, given successful implementation, promises that requirements will be congruent with customers' stated and implied needs, correct and testable. Although ISO suggests that assessment results indicate an organization's ability to achieve productivity, it is ambiguous whether this can be achieved via individual processes alone, or whether a combination of processes is required (El Emam et al., 1998). Furthermore, ISO is said to rely heavily on the expertise and training of assessors rather than its prescribed standards (Paulk, 1994). Finally, Sommerville and Sawyer (1997) define a process capability model for requirements engineering that provides a detailed set of implementation-ready industrial practices to improve the RE process. Sommerville and Sawyer argue that

higher maturity in this model will yield improved consistency in project risk and software quality and a '*capability to solve unforeseen requirements problems*.'

Unfortunately, empirical evidence on these benefits in the software industry is more challenging to find and categorize. Case studies that report on concrete effects of specific RE processes improvements that illustrate direct relationships between RE improvements are limited. In what follows, we consider existing related work; where necessary, when evidence relates to the benefits of SPI in general, we have attempted to infer a relationship to the RE process specifically. Further, to provide a rationale for the aspects we investigated in this study, the following section is largely organized according to the prevailing effects that other work has identified (i.e., productivity, quality, risk management). Where available, information on respective empirical methodologies is included.

A study conducted by El Emam and Birk (2000) examined whether requirements analysis process capability assessment measures (in this case the ISO/IEC 15504 standard) predict development performance among a variety of projects. The study assessed project performance by considering only the broad project-wide issues by defining performance in terms of post-mortem factors such as customer satisfaction, productivity, requirements satisfaction and morale. Although the study does indicates that REP capability assessment is related to project performance, particularly among large companies, it does not provide evidence on the specific impact of a requirements process throughout a project, such as the impact on developer decision making or project communication.

Productivity can take many forms, but essentially falls into two camps: increased development effectiveness as might be envisioned from the use of a new tool or process, or increases in efficiency, such as preventing rework (i.e., unwanted features) or lowering the cost of development. Lauesen and Vinter (2001) conducted a field experiment that considered RE performance primarily in terms cost-efficacy or hours saved, although they also found significant impact on software quality and the ability to meet schedule commitments. They report that particular techniques, such as user scenarios and early usability testing were clearly superior to others techniques, according to their metrics. Wohlwend and Rosenbaum (1993), in their account of long-term SPI improvements at Schlumberger, found that a successful experience can simply be a matter of on-time software delivery.

Next, there are many claims about improved quality realized through RE process improvement. Herbsleb and Goldenson (1996) conducted a survey among SEI assessed organizations and found that mature organizations, according to CMM, exhibited significant improvements in self-assessed product quality and customer satisfaction. Following software improvement initiatives at Schlumberger, engineering teams that had formally been plagued with delivering incomplete functionality, began to ship software that was 'complete' and 'correct' (Wohlwend and Rosenbaum, 1993).

While productivity and quality are critical factors in the development of software, Broadman and Johnson (1996) surveyed and interviewed 35 companies to find that, in fact, many companies look to implement SPI primarily as a means of reducing their exposure to risk. The companies they surveyed expressed a keen interest in the accurate assessment of costs and scheduling while decreasing variability in project success and/or performance. Although costs and scheduling can be considered productivity concerns,

forecasting in the initial stages of a project is clearly a matter of risk management (Humphrey et al., 1991) and tightly related to activities of requirements management. In Paulk's (1994) comparison of ISO 9001 with the CMM, he notes that both CMM and ISO share the same common goal to '*consistently* improve project performance' (emphasis added), implying reduced variability across projects.

There are other beneficial collateral effects that have been documented from successful SPI initiatives. For example, Wohlwend and Rosenbaum (1993), notes that after successful improvement developer morale improved markedly. Others (Broadman et al., 1996) suggest that companies have also observed less overtime, improved confidence, less turnover and increased intra-organizational co-operation. Hall et al. (2002) conducted a case study to understand the challenges organizations face regarding requirements processes and confirmed Herbsleb and Goldenson's (1996) findings that such challenges are often organizational in nature. In terms of REP effects they suggest that companies with higher-software maturity assessments enjoy better staff retention rates. Humphrey et al. (1991) in their case study of SPI at Hughes Aircraft, found that 'Pride [from continuous improvement] feeds on itself' and leads to success.

3. Case Study Description

3.1. The Company, Challenges in Requirements Practice, and Process Improvement Initiatives

The study of Requirements Engineering practice and process improvement was conducted at the Australian Center for Unisys Software (ACUS). ACUS is the software development group within an international multi-site organization with headquarters and marketing divisions in the US. The software product developed is product line software, where current releases provide enhanced functionality of the releases deployed to the customers, and which are integrated with hardware devices developed at other business units worldwide. Thus features for new software product releases are a result of the agreement between the US (marketing) business unit and ACUS, and consider both market needs (representing current as well as potential customer needs) and product strategy requirements (representing technology and engineering direction in line with the organizational strategy).

One of the significant challenges ACUS was facing when we started our study was the lack of experience with a formal requirements management process. This was exacerbated by the fact that the major stakeholders (i.e., Product managers, end-users) are geographically distributed across several continents. Before a RE process improvement initiative began, the situation could be summarized as follows (more details could be found in (Damian et al., 2002, 2004) and which was published in the Journal of Empirical Software Engineering (Damian et al., 2004):

- Requirements were communicated from product management and marketing in the US to ACUS in the form of one-line statements of "required features." There was not a well-defined or direct line of communication with the software clients or end-users.

- The features were not clearly defined or documented, did not provide sufficient information and thus were not fully understood by developers. They were ambiguous in stating the required functionality, making it difficult to plan for change.

- Individual designs were built for these features, depending heavily on designers' interpretation of those feature requests.

- The problem was compounded by inadequate change management practice and no record of the reasoning behind requests of particular features (referred to as "requirements rationale"), which should have been accessible via traceability links within requirements.

As a result, the software development projects at ACUS always experienced significant requirements creep. The requirements negotiations between the development and US product management unit were ineffective because ACUS management had difficulty in (1) understanding the requested features and (2) providing reasonably accurate estimates of development effort.

In August 2001 software process improvement was initiated at ACUS and Requirements Engineering was one of the key areas of improvement. The RE process was revised to include:

- Requirements analysis sessions to refine the feature requests and derive more detailed technical requirements (referred to simply as requirements). These sessions involved: (1) cross-functional team participation, (2) group analysis sessions, (3) high-level design sessions attempting to understand the impact of the requested features on the architecture of the existing release, (4) well-defined specification of requirements to include a more structured description of the requirements in a requirements document and (5) the definition of traceability links to test scenario were conceived during the

Box 1: Requirement example from ACUS, including rationale and test scenario

Initial Feature: Scriptable Interface

Requirement Description: EA Developer shall provide a version of the D2L utility that allows the exchange of affected screen definitions from Graphical Interface Workbench to EA Developer.

Rationale: Customers may already have painted versions of the affected screens in GIW. They will likely wish to keep those existing painted screens, and be able to enhance them in the future.

Test Scenario: Enhance screens in a GIW environment. Load the provided model into EA Developer. Use D2L to transfer the enhanced screens from the GIW environment to EA Developer, into the installed model. Verify that the enhanced screens are available in EA Developer and can be further enhanced in EA Developer Painter.

Box 2: Sentence template constructs (in bold) and an example of a full text requirement parsed with the sentence template (used with the permission of Halligan, R. (2000))

Full text requirement: When in full operation the computer system shall provide thirty per cent reserve in channel capacity.

Initiator of Action:	The computer system
Conditions for Action:	in full operation
Action:	shall provide
Constraints for Action:	
Object of Action:	reserve in channel capacity
Refinement/Source of Object	of 30%
Refinement/Destination of Action:	

sessions; these links were later stored with the requirements in Requisite Pro, a requirement management tool. An example of a requirement is provided in Box 1 together with its rationale and test scenario. This requirement is one of many decomposed from the initial feature request (i.e., "Scriptable Interface"). A sentence template was used to structure the description of each requirement; this is illustrated in Box 2 with an unrelated requirement.

- Revised change management process. Those problems, identified by development or management, which affected requirements or other functional areas, were formally recognized in a change request document. This document described the nature of the change, its extent and an estimation of effort of implementation. These factors were to be considered by other engineers and decision makers who were expected to review and approve the requested change. Once approvals were given by all the mandatory approvers, the Project Manager authorized the change and it proceeded to implementation.

3.2. Study Motivation: An Empirical Assessment of the Impact of Improved Requirements Engineering Practice on Downstream Software Development

We conducted an initial empirical assessment of the improvements in the requirements practice before design commenced, in March 2002, and at that time the organization had shown visible improvement. As reported in the ISESE 2002 paper (Damian et al., 2002), our findings indicated tangible benefits as well as *perceived long-term* benefits during design and testing. We found that due to more clearly defined, specified, and better

understood requirements, ACUS showed an enhanced ability to address the market needs and product strategy requirements. The catalyst behind these improvements included project management leadership, managing the human dimension, collaboration among stakeholders and senior management support.

However, our early empirical assessment was insufficient in providing evidence of improvement beyond an initial enthusiasm with the revised requirements practice. Developers had been so frustrated by poorly defined feature requirements that additional study was necessary to determine whether their enthusiasm was well placed or if they were merely hoping for the best. An assessment at the end of the development life-cycle was needed to address the *long-term* benefits of the improved RE practice such as higher software quality and productivity in software development.

Therefore an assessment was conducted to verify whether the *perceived* long-term benefits (identified in the earlier phase of the study) were realized. The results of this second empirical assessment are the topic of this paper.

3.3. Design of the Case Study

3.3.1. Data Collection Procedures

Thirty-one project members from several functional departments (i.e., software engineering, documentation and management) participated in this stage of the study. They were invited to participate in the study based on their knowledge of the product, development process and 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 management process in the last 15 years.

The data collection methods included a questionnaire, interviews, and document inspection. All interviews, open-ended and semi-structured, were recorded and informally analyzed for patterns in responses. The requirements process documentation was studied, as well as other (current and historical) project information. Additionally, some project artifacts were considered including, requirements specifications, change requests and entries within the requirements management tool. This being the first time the requirements process was rigorously defined, historical data was very limited and instead we relied on the extensive professional experience of ACUS engineers at ACUS to provide comparison to previous practice.

The majority of our evidence was collected anonymously to protect respondents from the scrutiny of the organization. In particular, the names of participants involved were largely unknown to ACUS senior management, to remove the possibility of apprehension to provide accurate information. To further avoid the risk of a perceived apprehension towards job loss if confidential data is provided to the senior management, the questionnaires were administered in paper copies and collected individually by the first author.

Furthermore, respondents were encouraged to be critical of their experiences with the new processes in an effort to further improve them. We are confident that these answers and opinions are relatively objective.

3.3.2. Aspects Investigated

Below we describe the aspects of the process improvement that we investigated. Initial data gathering on these aspects was conducted through a questionnaire (see Appendix) and followed up through interviews. In the text below, each section describes a category of questionnaire items, together with the underlying rationale.

General Feedback About the Role of Revised RE Process at ACUS. Given that this was a follow-up of the initial assessment of the RE practice at ACUS (immediately after the 'requirements phase'), we wanted to obtain general feedback on longer-term effects of the revised RE process in the software development life-cycle, after the design, coding and testing phases were completed. In particular, to obtain the project members' judgment on how important the revised requirements process was, how it affected the work of the individual project members and whether the time spent on the 'requirements phase' was sufficient. We also wanted to obtain insights into which individual components of the revised RE process had greater effect in the software development than others to be able to frame advice for practitioners who wish to adopt only parts of the RE practice used at ACUS.

More Specific Aspects: Better Definition of the Problem and Potential Benefits. As Brooks (1987) indicates, the purpose of the early development stages is "determining what to build." The role of Requirements Engineering within the Software Development process is to define the problem and to describe the functional features of the envisioned solution to solve that problem. At ACUS, this activity included a detailed account of the nature of required system features, how they function, their dependencies and inter-dependencies on other features in the system architecture. The description also included qualifiers to describe non-functional characteristics of how features should behave, such as performance, portability and so on. One purpose of exploring requirements was to systematically explore the problem space while attempting to define functional characteristics of the software system to address this problem space.

To understand the effects of improved feature refinement during subsequent development activities, we asked engineers to comment whether they felt the process had revealed further feature details and how important that information was depending on the development activity (design, implementation, testing and documentation). One effect that we anticipated was less rework conducted during development as a tangible manifestation of better problem understanding and more detailed feature definition, since dependence on guesswork and arbitrary invention was potentially reduced. Evidence on how developers used the information about requirements during later stages is important and we investigated which tasks were improved by an increased understanding of requirements. Building on the evidence from the early assessment we investigated whether the revised RE process enabled developers to make more informed decisions, improve their sense of ownership, and improve their sense of accountability.

Additionally, we investigated whether better definition of features upfront improved communication and in particular whether or not it helped resolve common complaints of excessive communication and consultation that was necessary to clarify ambiguous

features as experienced in previous projects. Of special interest was finding out whether this "improved" communication improved productivity and product quality.

More Specific Aspects: Estimation. The investigation of planning, scheduling and resource allocation is important in understanding whether the revised requirements practice had longer term effects in the project development. These activities are conducted throughout the software development process and rely heavily on estimations that can be enriched by the fruits of requirements engineering. They are particularly important in the early stages of development, precisely when there is a distinct scarcity of information to support meaningful, accurate estimations. These estimations serve to approximate the extent of required effort, inherent risk and time needed to fulfill project objectives. Requirements engineering provides critical information by providing a vehicle with which to consider the extent of features and their complexity. In addition, these estimations are often separated by feature and by development phase: design, implementation and so on, driving demand for further detailed information.

To investigate the extent by which requirements engineering process improved the estimations conducted at ACUS, team-leaders and managers were asked how important they felt the RE process was to estimation in general and to estimations made according to development phase. Additionally, we were interested in understanding the effects of the more thorough analysis of features and their refinement into technical requirements. We compared the accuracy of the estimations of effort performed before the requirements analysis (referred to "pre-analysis estimations" henceforth) with those made after the requirements analysis sessions (referred to "post-analysis estimations henceforth). Furthermore, we asked participants to comment on why estimations may have been inaccurate and how they might improve the accuracy of those estimations in the future. Beyond exploring the issue, this questioning was meant to consider how the RE process effects might be impeded by practical matters.

More Specific Aspects: Change Management. Change management concerns an organization's ability to address change. New requirements can emerge and existing requirements can change throughout the development process (Sommerville and Sawyer, 1997). Not only is it important to control change, but it is important to address those changes as soon as possible as it has been shown that it is far more costly to correct requirements errors later rather than earlier (Boehm, 1988). Changes are responsible for quietly altering the context of the software problem in fundamental ways; studies of software projects (The Chaos report, 1997) have found that changing requirements and specifications were often blamed as significant challenges to project success.

At ACUS we performed a detailed analysis on change management artifacts, primarily change request documents. In particular an examination was made to determine the extent of the analysis of impact when a change was requested. This was complemented by an investigation of change management's effect on the software development process at ACUS, by asking managers and engineers whether or not the revised requirements process enhanced their ability to assess the impact of requirements changes (i.e., Whether communication with other important stakeholders during the decision making process was helpful).

4. Case Study Findings

The findings of the case study are summarized in the sections that follow and are organized by the aspects investigated. Some participants chose not to answer all questions on their questionnaire, thus the number of responses received for each question is also reported. Themes that emerged from these findings are discussed in detail in Section 5 that follows.

4.1. General Feedback on the Role of the Revised RE Process in the Project

At ACUS the revised RE process required a cultural shift in the mind set of the engineers as they moved from a silo and solo based requirements analysis model to a cross functional and team based model. This is typified by the considerable amount of time that was spent in group requirements analysis meetings. 92% of 24 participants felt that the revised requirements process was important (Q1) and 71% of 17 engineers would spend even more time on the requirements phase in the future (Q3). In participants' words (Q2), it was as simple as helping to "discover goals more easily," "make the boundaries of features clearer" or "understand what needed to be done." For some the important effect is to broaden developer thought to a more comprehensive perspective: "[improvement] made me more aware of the impact in other areas, it made me think of the total package," "[requirements] formed a useful framework which underscored the whole design."

4.2. Better Definition of the Problem and Potential Benefits

4.2.1. Improved Understanding of Details, Dependencies and Complexities of Features

Engineers unanimously agreed (100% of respondents) that the requirements process revealed further details, dependencies and complexities of features (Q5). Chart 1 (23 respondents, Q6) shows the percentage of responses on the importance of understanding requirements in later phases of development, indicating that many engineers felt that the revised RE process was particularly productive and worthwhile use of resources.

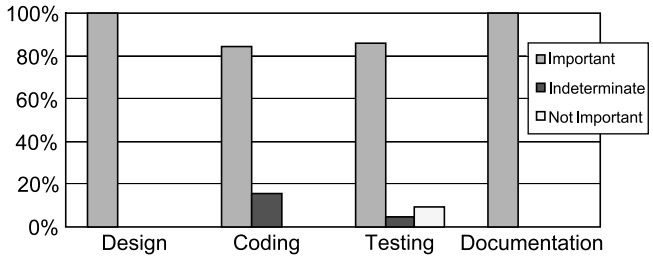

Chart 1. Importance of understanding requirements during design, coding, testing and documentation.

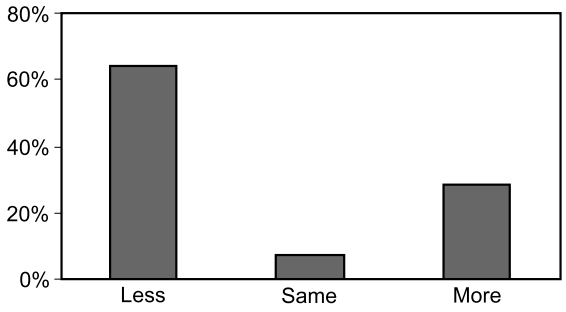

Chart 2. Amount of rework needed in the project as compared to previous projects.

The requirements analysis sessions "helped to identify dependencies between requirements very early," and "identify features that would have otherwise been unaccounted for until much later."

4.2.2. Reduced Wasted Effort

In terms of tangible benefits evident in reduced rework results were also very positive. While a majority of respondents (see Chart 2, 64%, 23 respondents, Q8) felt that there had been less rework under the revised process, 30% indicated there was more. Some respondents qualified their choice: "there is always some level of rework due to inevitable technical issues."

4.2.3. Intangible Benefits: Improved Communication

In the earlier assessment intra-team and inter-team communication was perceived to be improved due to revised RE process. When asked whether this effect continued beyond requirements and into later stages of development most developers agreed (Chart 3, 24

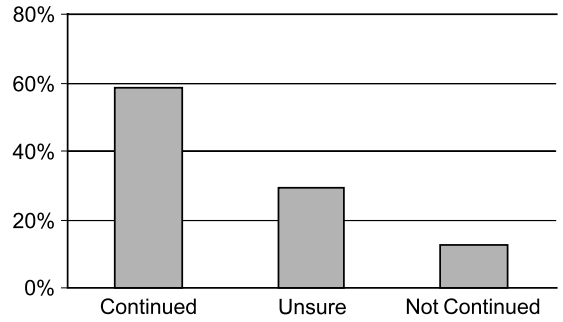

Chart 3. Extent to which improved communication continued beyond requirements phase.

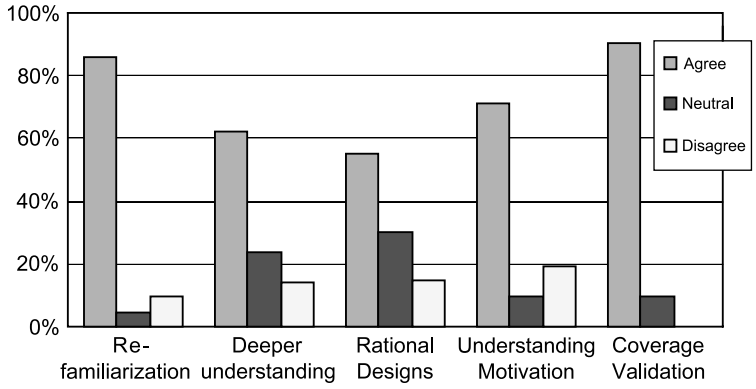

Chart 4. Agreement that requirement artefacts enabled post-requirement activities.

respondents, Q10). Of those who responded neutrally and negatively to this question half were from test and documentation departments within the organization. This significant representation is explained by comments made by managers that indicate that testing and documentation were sometimes left out of communication during analysis sessions.

When asked about communication respondents felt that 'clarification' communication had been reduced, but that confirmations were still sought (Q10). The requirements provided broader perspective and answered questions relating to rationale. As Chart 4 (23 responses, Q7) shows, requirements were used extensively by developers, most significantly to re-familiarize themselves with the characteristics of the feature (86%), indicating that developers sought re-iteration of information on features and that they were able to utilize the requirements artifacts to aid them in that task; and to validate coverage of features (90%). This confirms requirements' role as a means to checklist system functionality during testing and review. Engineers responded favorably in other areas too, reporting that requirements had impact on deepening understanding, facilitating rational designs and providing feature rationale.

4.2.4. Intangible Benefits: Informed Decisions

Further, data (see Chart 5, 21 respondents, Q9) indicates that improved feature understanding contributed significantly to developers' ability to make informed decisions. A somewhat less convincing though still positive result was with respect to the process helping to foster a sense of accountability (66% of responses). Creativity and ownership enhancements were less sure, but as one manager suggested this may have been due to the perception that the more rigorously defined requirements provided less room for creativity among designers and implementers. Improved understanding of the features also enabled ACUS to negotiate with its US-based product managers from a position of strength, given that the negotiations were based on firm data. These negotiation sessions were shorter given that the discussions were centered on tangible information. The scope creep was

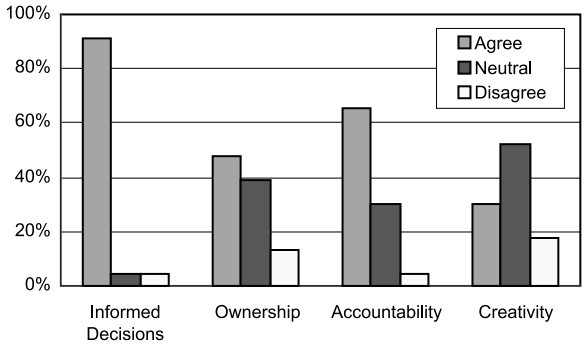

Chart 5. Agreement that requirements had intangible benefits.

minimized as the requirements analysis sessions helped ACUS to understand and define very clearly what is being captured.

4.3. Estimations

The responses from 15 team leaders and managers provide evidence of strong agreement that the requirements were an important element of estimation. In particular, Chart 6 (13 respondents, Q14) illustrates that over 80% of respondents found that the thorough analysis of features was important in estimating effort required during design and implementation. Respondents were appreciative of the more thorough feature definitions and the added focus that the requirements process reaped, ultimately preventing the need for extensive guessing and unwarranted commitment to feature development. Managers attributed fewer estimation errors to systematic estimation of smaller units (i.e., technical requirements), a benefit of the improved requirements process.

We also analyzed the quantitative data available on estimations in the project. At ACUS, estimations of effort required for design, coding, test and documentation were prepared (1) during an initial feasibility study, before starting the new RE process and

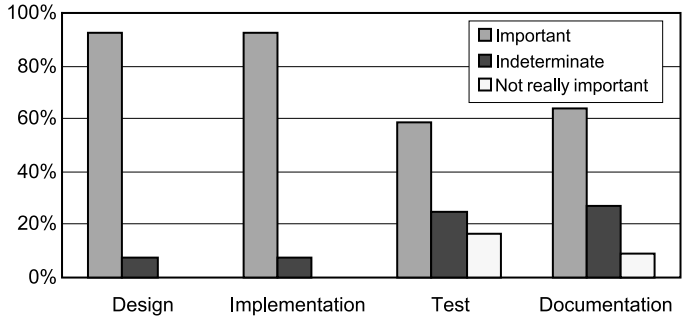

Chart 6. Importance of requirements in estimation for design, implementation, testing and documentation.

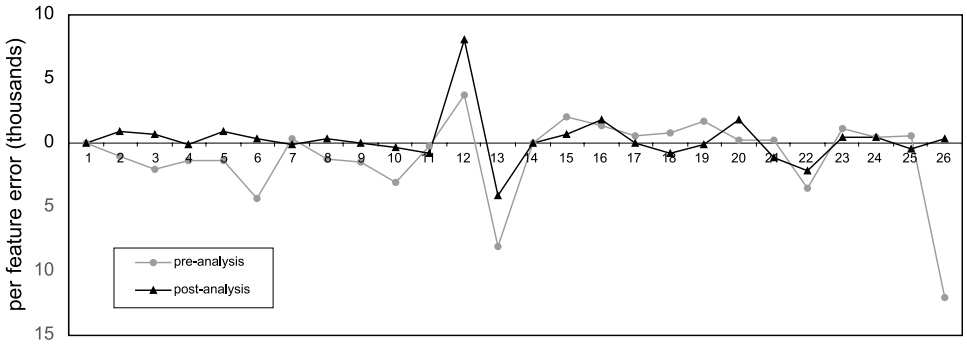

Chart 7. Difference between actual effort expended and estimations. (In person-hours per feature, for 26 features in the project.)

before requirements analysis, and (2) after requirements analysis when features were detailed into more manageable requirements, using the revised RE process. Finally, actual values were calculated at the end of the project. These estimates and actual values were collected for each of the project's 26 features. Chart 7 shows the difference between actual expended efforts and the effort estimations made before and after requirements analysis. Values above the x-axis correspond to optimistic estimations that are lower than actual required effort (ie., actual – estimate > 0).

Chart 7 shows that the estimations made after requirements analysis were closer to final effort, while the estimations before requirements analysis were predominantly pessimistic. Further, Chart 8 below ignores 'the-direction' of error, instead showing absolute estimation error on the left-hand y-axis (referred to as "delta pre- and post-analysis"), and cumulative [absolute] estimation error (shown by horizontal dotted-lines) on the right-hand y-axis. In particular, the cumulative error clearly indicates the superior accuracy of the post-analysis estimations as compared to the pre-analysis estimations.

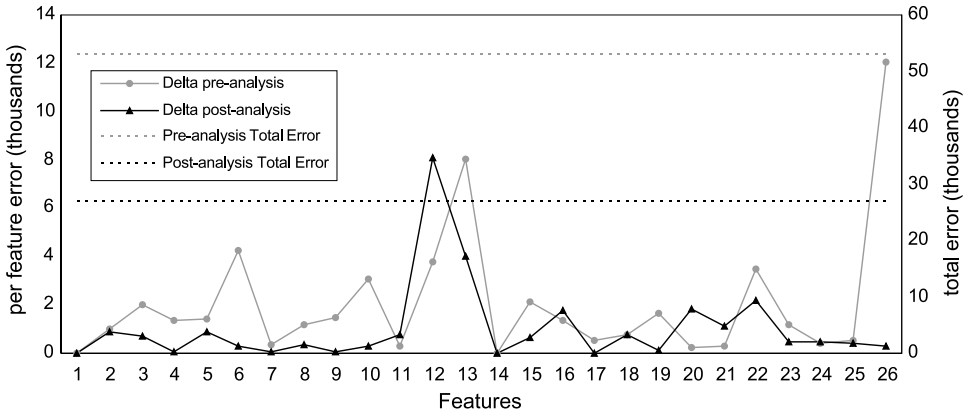

Chart 8. Estimation (absolute) error per feature and total cumulative error (in person-hours).

Table 1. Change effects on artefacts.

Artifacts affected by change	Number
Requirements specifications	50
Functional specifications	50
Design specifications	16
Documentation	13
Test plans	6
Other (e.g., project plans)	23

While it is true that one would expect estimations to improve as the project progresses, no development effort occurred between these estimates except for the requirements analysis phase itself. This leads us to conclude that the improvement in estimation can be primarily attributed to an enhanced ability to use the requirements in the estimation process.

4.4. Change Management

The results of this investigation provided some strong insights. 77 change requests were recorded throughout the development life-cycle (2 during requirements definition, 37 during design, 31 during coding and unit testing and 7 during integration testing). Change management artifacts cited documents that those changes would affect. As shown in Table 1, out of 158 documents cited over 50 involved modifications to the requirements specification. This suggests that the team had often accessed the documents and artifacts affected by change. Finally, it may be worth noting that while these change requests may have affected the source code, the organization did not collect this data since relationships between requirements and code were not being maintained.

Approximately seven (on average 7.06) engineers and managers were involved in every change request. As shown in Table 2, almost all change requests involved wide participation including engineering managers, program manager, project managers and perhaps most significantly, consistent involvement from testing (40%) and documentation groups (49%).

Table 2. Proportion of change requests involving particular roles.

Functional role of participants	Proportion of requests in which the role participated
Engineering Manager	100%
Product Manager	65%
Project Manager	91%
Testing	40%
Documentation	49%

The revised RE process was positively assessed in providing a mechanism for controlling change: "change control process made changes more documented ... it made people analyze about the changes before they were approved." In particular, 7 out of 8 team-leads indicated that the requirements process provided them with enhanced ability to assess impact of changing requirements, the only dissenting neutral response indicated that "[it was] difficult to say, from my perspective. The particular area I worked on was unaffected by any of the change requests."

5. Discussion

5.1. Increased Productivity Through Increased Understanding of Features

The evidence in this study indicates that the revised Requirements practice had significant positive impact affecting the later stages in the development, beyond merely providing definition and refinement of the features during the 'requirements stage.' Compared to previous projects, productivity increased: rework lessened, communication effectiveness increased and capacity for informed decision making improved.

In the absence of a well defined measure, we believe that productivity has been increased, in part, by helping engineers, as a few mentioned, "focus on individual work items." The cross-functional involvement during analysis sessions achieved more thorough up-front thinking and "mutually agreed understanding of what was required to be done," and further served as an "ice-breaker" such that developers were more willing to discuss issues with their counterparts during later stages of development: "This improved coding efficiency many-fold."

The increased effectiveness of the communication can be discussed from two perspectives. On the one hand, the findings indicate that the improved understanding of details, dependencies and complexities of features significantly decreased the volume of "clarifications," often redundant communication. The requirements served as a means for low-cost, broad-based knowledge dissemination. In fact, the better-defined and structured RS served as a tool to propagate information about the features and their interdependencies. Project members used it as a persistent resource in their daily activities, as a "solid foundation" to seek information on requirements and on which they based their designs and programming decisions on, avoiding unnecessary and redundant communication. We believe that the enhanced ability to make more informed decisions at the micro level led to less time spent fixing bugs and ultimately resulted in higher developer productivity.

On the other hand, the requirements specification increased the quality of the communication, the other dimension along which effectiveness of communication can be considered. Project members could more effectively reference detailed descriptions of features and requirements and test scenarios (documented in RS) when ambiguities had to be removed. The quality of the communication was further increased because the requirements analysis sessions involved extensive cross-functional interaction. This way different departments were brought together to communicate collectively on the

particular requirement, providing an opportunity for engineers to understand the responsibilities and concerns of other functional roles in the larger development team.

5.2. Estimations

This study presents a clear indication of the role of requirements practice in improving estimation in software development and project management. At ACUS, the estimation process required team leaders, in consultation with their engineers, to make effort estimates for particular features. The overwhelming response from those involved in the estimations suggests improvement in estimation due to improved RE practice. The fact that the post-analysis estimates were much closer to the actual efforts strongly indicate that the requirements analysis sessions that generated technical requirements were important in enabling more accurate estimations.

Although pre-analysis estimates are expectedly inaccurate, due to the ambiguity of 'one-statement' feature requests, one might expect those estimates to be optimistic, due to easily overlooked technical detail, rather than the decidedly pessimistic estimations made at ACUS. This is likely because ACUS had, in past projects, over-committed to features causing severe schedule overruns. Furthermore, it appears that engineers were unsure of the requirements' usefulness in making estimations for documentation and especially testing, as manager comments suggest that involvement from different functional units in the estimation process was a new phenomenon. Managers felt test and documentation departments did not fully grasp how requirements could be used to improve estimations. Pre-analysis estimates also included a component of rework, which to some extent did not eventuate, given the improvements resulting from earlier definition of the work scope. This added to the volatility of the pre-analysis estimates, given that pre-analysis estimates were 'best guesses' without the benefit of a clear understanding of the requirements.

When asked why discrepancies might exist between estimations and actuals, participant opinions varied widely. A common complaint was simply the lack of information, originating from natural uncertainty one would expect in the early phases of development. They suggested that more extensively detailed requirements would have contributed to more accurate estimations alluding to the inevitable challenge in striking a balance between information gathering and estimation accuracy. Other problems included: complexity of the project, unfamiliarity with new technology and tools, and poor understanding of individual skill levels.

In summary, ACUS' estimation ability was enhanced largely because of their ability to use the more detailed information on features analyzed as part of the revised RE process. Managers in particular were very positive with these results compared to relatively poor estimates in past projects, especially when considering the lack of training in estimation methods provided to some project areas. Additionally, more accurate estimations were useful during change management, discussed in the next section. Improved estimates enabled early identification of resource constraints motivating the early revision of project commitments to accommodate these constraints. As the project manager noted:

"the revised RE practice enabled us to conduct better estimates and project planning and as a result, cull functionality for which we did not have adequate resources. In prior releases, we would have attempted to deliver more functionality as we would have had a poorer understanding of the requirements and their scope."

5.3. Change Management

Managing change is one of the most significant challenges in software development. In particular, changes in requirements often arise from external events originating outside the organization, for example: unpredictable market conditions or customer demands. Change can also originate internally from within the organization, typically from the natural refinement of requirements as the solution space is more thoroughly explored during design or implementation. ACUS has reported that past projects have exhibited very poor ability to react to change resulting in uncontrolled feature creep. In this study, evidence suggests that the revised practice in requirements had a positive effect on the ability to control change. It resulted in an increased level of stakeholder involvement that provided a more thorough analysis of impact and more informed reactions in face of change.

The fact that 50 modifications to the Requirements Specification (RS) were made as a result of 77 change requests (the majority during design and coding) suggests that engineers were by and large successfully able to leverage traceability links back to requirements, thus facilitating more accurate impact analysis. Further, the fact that only 4 modifications to the RS were initiated during integration testing (as a result of 7 change requests) suggests that most of the requirements-related changes were addressed before integration testing. Assuming that integration testing exhibited similar rigor as testing in earlier phases, this is a very positive result.

The process of managing requirements changes also facilitated an open channel for communication and decision making. During change requests, the consistent simultaneous involvement by senior management and developers, designers and testers has an important implication: change requests served to bring together not only the functional organization horizontal alignment (designers, developers, testers and documenters), but also vertical alignment of organizational responsibility (engineers, teams leads, technical managers and executive management). Essentially these change requests can be seen to provide a precisely focused forum in which developers and decision makers could more openly communicate. In the face of requests for change, ACUS senior management no longer had to make decisions without thorough information on the impact of change. The involvement of several organizational layers (e.g., designers, team leaders) provided for more informed decisions and an increased ability to reject changes that could not be accommodated.

At the same time, revisions in the RE practice need to be considered along with the improvements in the change management process at ACUS when heightened ability to control change is discussed. Defining and enforcing a strict change management protocol not only provided infrastructure for assessing change requests, it also demanded additional effort from developers initiating change. This additional cost to introduce change was seen

to help prevent ad hoc changes altogether. Those changes that were considered important (enough so that the change process is initiated) were rigorously scrutinized for impact through the reviewer comments provided by engineering roles that participated in the change request document — not only with respect to design and implementation, but also testing and documentation. It is likely that the cross-functional communication and involvement evident in change artifacts, as described above, contributed significantly to their more holistic assessment.

In summary, improved requirements change management at ACUS provided project members and senior project stakeholders enhanced ability to negotiate any requests for changes and thus to prevent feature creep throughout the development life-cycle.

6. Advice for Process Improvement Practitioners

Our investigation of perceived benefits of the various components of the RE process improvement at ACUS provide evidence about the usefulness of individual RE components in the software development. Although everyone agreed that the process was important overall, it is clear from the results shown in Chart 9 that some aspects of the process were far more critical than others, at least according to engineer perception. Parts of the requirements process at ACUS that were clearly identified as having considerable positive effect include team involvement, analysis sessions and test scenarios. The practical consequence of this result is of particular importance for organizations that are optimizing their exposure to new process initiatives, namely those organizations looking to initiate a RE process revision, or those endeavoring to streamline an existing RE process. This latter issue has already become relevant for ACUS in their efforts to curtail costs. In particular, we note that team involvement is largely accomplished through participation in the analysis sessions, amplifying their importance. This makes analysis sessions an attractive prospect for process analysts and leads us to advise on their minimal inclusion in RE process improvement.

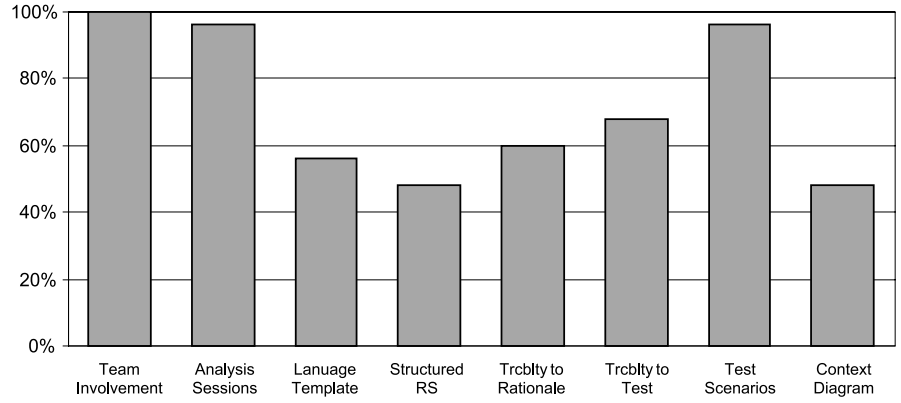

Chart 9. Percentage of affirmative responses to importance of components of the revised RE process in the software development.

Acknowledging that these recommendations are based on a single case study, it is not our intention to claim that they are general results. We do believe however, that organizations that exhibit similar characteristics as ACUS could greatly benefit from these recommendations. In particular, the following are critical organizational characteristics that are likely to have contributed to these significant improvements: ACUS is a product line development organization that had suffered from inadequate communication with their customer representatives who had routinely requested very high-level product features. These features were highly ambiguous and in many cases required deep technical understanding of the system to fully appreciate and understand. The development culture bred highly factional teams that tended to work in isolation rather than in open cooperation.

Finally, practitioners should be aware that although ACUS implemented a whole set of changes to the requirements process, it may seem premature to recommend only a subset of that process. However, while there is no particular reason why these components should be separable, there is also no particular reason why they should not. In fact, some of these components do appear to be disjoint, for example, the language template appears quite independent from developing test scenarios during requirements, or recording traceability information. Therefore, we feel confident in making these recommendations despite this caveat.

For all of the above reasons, analysis sessions involving participants from cross-functional teams are very helpful because they would involve people that understand the language and come together to achieve the shared understanding of the feature.

7. Limitations

Limited quantitative historical data was available from past projects, and thus an evaluation of improvement through experimental comparisons to previous projects at ACUS was impossible. As a result, we had to rely on the input from project participants and knowledge of past practice at ACUS. This input emerged as a valuable resource since the majority of these people have been employed at ACUS for the last 5–15 years, bringing a wealth of anecdotal information about weaknesses of the software process. Thus, from a practical perspective, not being able to quantify the similarities or differences of performance in past projects, and the reliance on opinions of expert software engineers, has minimum consequence when discussing the advantages afforded to the practicing researcher or practitioner. Due to the nature of their former process, objective measurements would have been impossible to make. For example, metrics on change management could not be developed or maintained in the absence of a well-defined change management process. In contrast, the effort re-estimation conducted after the analysis phase of development enabled us to analyze improvements in the estimation as reported in this study.

Furthermore, there is a strong claim that could be argued discounting our reliance on human perspective, given the inherent bias that could exist in the study subjects. While their enthusiasm for the process improvement initiative could highly influence their

perspective, self-assessment toward optimism or even their productivity, as a result of the Hawthorne Effect (Mayo, 1933), there are compelling reasons to rely on their testimony nevertheless. First, much of the enthusiasm detected early in the study and reported earlier (Damian et al., 2002) appeared to have waned considerably as the project continued to proceed under the new processes. Secondly, for many engineers, additional process can be viewed as a restriction on the freedom to conduct development in their preferred manner, for this reason it is just as reasonable to expect indifference, if not opposition, to affect their responses.

Finally, the RE process improvements at ACUS occurred in parallel with initiatives for improvement in other process areas such as software quality assurance, project planning and project tracking. We recognize that it is very difficult to ascertain to what extent those other improvements have confounded the effects we observed in this study. Or, for that matter, to evaluate the interaction between the requirements engineering process and other processes, in particular to assess whether the RE process was of help or of hindrance to these processes and vice-versa. One could consider this to be the strongest limitation of our study and in general of assessing improvements in RE practice in an organization that is involved in a larger improvement initiative. Stronger evidence could have been brought by an assessment that designed an experiment that compared similar projects where only RE process differed, however this is a difficult ideal to achieve in an industrial, empirical investigation when organizations are motivated to improve their productivity rather than conduct scientific experimentation. In lieu of this, we are left to rely on the expert opinion of practitioners within the organization to identify individual effects of RE process changes. Although the potential for unpredictable process interaction exists even within more stable, well-established environments, we acknowledge that this interaction requires further study. It is the topic of our future research to achieve a better understanding of the interaction between improvements in the RE process and the other processes in this organization. In particular, we plan on gathering data about the extent to which improvements in the requirements process enabled or hindered improvements in other process areas. This will allow us to develop a map of interaction between the requirements process and other processes in the organization and to identify which other processes were most affected by improvements in requirements engineering.

8. Conclusion

This paper reported the results of a case study that investigated the role of "good RE" practice in software process improvement. Data from the 18 month software development project was analyzed for effects of more rigorous RE practice in later stages of the development such as design, coding and testing, as well as other project related activities such as project planning and scope management. Evidence was found to confirm claims in the literature of the role of requirements processes in increasing productivity, improving project planning, better problem definition and more effective communication in project development.

The study adds to the scarce evidence on the role of improvements in the requirements management practice, invaluable for researchers to define areas for further research and for practitioner to define similar improvements and their appropriate benefits. The study provides an example of how practitioners can analyze their process improvement besides other more general ways of assessing software process improvement. ACUS achieved CMM Level 2 at the time of this empirical assessment. Requirements Managements was a Key Process Area that scored an improvement from 5.4 to 7.4. This suggests that the revised RE practice also contributed to software process improvement in the context of CMM assessment. While CMM identifies that there was a good process in place and it was practiced, the assessment here sheds light on the effectiveness of the revised RE process and hence its role in software process improvement. It is also important to consider the assessment of process improvement at ACUS over a long term period spanning multiple projects. This study provides an early indication that a shift in culture has begun at ACUS. The most significant effect of the revised RE practice at ACUS was the transition from the "silo culture" to the cross-functional and team involvement in activities of requirements analysis, which had tangible effects during design, coding and testing. However, culture change is a long term process, and this initial improvement has at best created a foundation for assessment of future changes, measurement and ultimately continued improvement. The new methods introduced in this project (such as different components of the RE process, change management) can continue to be used to collect historical data and conduct quantitative comparative analysis of this data, providing evidence beyond the participants' anecdotal evaluation of improvement initiatives. Actual change in the organization's culture with respect to the attitude towards rigorous RE processes will have to be assessed in the future, and forthcoming projects will indicate whether this strong beginning has materialized in culture change at ACUS.

Acknowledgments

Many thanks go to the study participants, software engineers and other project members at ACUS. Financial assistance was made through NSERC.

Appendix: Questions in the Questionnaire

General Feedback About the Role of Revised RE Process at ACUS (Section 4.1 in paper)

1. How important do you feel the revised requirements process was at ACUS? Why?

Very Important	Important	Indeterminate	Somewhat Important	Not Important at all
□	□	□	□	□

2. How did the revised RE process help YOU in your work?
3. With respect to the 'requirements phase,' in the future would you spend more or less time and/or effort on this phase of development: (Why?)

Far More	More	About the same	Less	Far Less
□	□	□	□	□

4. The revised RE process consisted of several components, as shown below. Please indicate, which ones were most important/useful in your work, in activities such as design, condign implementation or documentation.

Components of the RE process	Design	Implementation	Testing	Documentation
Involvement from multiple teams	□	□	□	□
Group analysis sessions	□	□	□	□
Sentence template in writing req's to achieve consistency, in the capture sessions	□	□	□	□
Use of a structured RS	□	□	□	□
Traceability links to requirements rationale	□	□	□	□
Traceability links to test scenario	□	□	□	□
Definition of the test scenario at RE stage	□	□	□	□
Use of context diagram in the capture sessions	□	□	□	□

More Specific Aspects: Better Definition of the Problem and Potential Benefits (Section 2.4.2 in paper)

5. Do you feel that the improved requirements process revealed further details, inter-dependences and complexities of features?

Strongly Agree	Agree	No Effect	Disagree	Strongly Disagree
□	□	□	□	□

6. In your design, coding, testing, or documentation activities, how important was it to understand the features and technical requirements?

	Very Important	Important	Indeterminate	Somewhat Important	Not Important at all
Design	□	□	□	□	□
Coding	□	□	□	□	□
Testing	□	□	□	□	□
Documentation	□	□	□	□	□

7. Have you, during design, implementation, testing, or documentation, made use of the technical requirements that motivated particular features?

	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree	In which activities listed above?
To re-familiarize yourself with the feature	□	□	□	□	□	
To gain deeper understanding of the feature	□	□	□	□	□	
To facilitate 'rational' designs	□	□	□	□	□	
To understand the motivation behind the feature	□	□	□	□	□	
To ensure complete coverage/ compliance with requirements	□	□	□	□	□	

8. Compared to projects in the past: has there been more or less rework during development (but before deployment):

Far More	More	About the Same	Fewer	Far Fewer
□	□	□	□	□

9. Did understanding help with:

	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree	Don't Know
. . . facilitating more informed accurate decisions?	□	□	□	□	□	□
. . . improving your sense of ownership?	□	□	□	□	□	□
. . . improving your sense of accountability?	□	□	□	□	□	□
. . . inspiring you to be more creative in providing a solution?	□	□	□	□	□	□
. . . improving the communication with the customers (BI)	□	□	□	□	□	□

10. Based on an early assessment, the requirements process improved communication among developers. Did communication generated by these sessions continue outside the sessions and in later stages of development?

Very Improved	Improved	Unsure	No Improvement
□	□	□	□

11. In an earlier assessment, it was reported that engineers often only vaguely understood the requirements and often had to "walk to others' cubicles (or phone) and ask clarifications." To what extent do you believe the revised RE process removed that in the current project?

12. How do you believe the communication inspired by the requirements sessions improved or deteriorated (a) productivity or (b) product quality.

More Specific Aspects: Estimation (Section 2.4.3 in paper)

13. How important was the use of features and technical requirements in the estimation process?

Very important	Important	Indeterminate	Not really important	Not important at all
□	□	□	□	□

14. How important was the use of technical requirements in the estimation for:

	Very important	Important	Indeterminate	Not really important	Not important at all
Design	□	□	□	□	□
Implementation	□	□	□	□	□
Testing	□	□	□	□	□
Documentation	□	□	□	□	□

15. What was the source of failure or inaccuracy, if any, in the above-mentioned areas (resource allocation and planning)?
16. In thinking about your estimates, can you think of reasons for discrepancies? With respect to (a) Design, (b) Implementation, (c) Testing and (d) Documentation
17. How could you have improved your estimations?
18. Should those estimates have been improved? How?

More Specific Aspects: Change Management (Section 2.4.4 in paper)

19. Did the revised requirements process provided you with enhanced ability to assess impact of changing requirements?

References

Boehm, B. 1988. A spiral model of software development and enhancement. *IEEE Computer* 21(2): May, 61–72.

Broadman, J., and Johnson, D. 1996. Return on investment from software process improvement as measured by U.S. industry. *Crosstalk* 9(4): 23–29.

Brooks, F. 1987. No silver bullet: Essence and accidents of software engineering. *Computer* 20(4): April, 10–19.

CMM, Capability Maturity Model for Software, 1991: *CMU/SEI-91-TR-24*, Software Engineering Institute, Carnegie Mellon University.

Curtis, B., Krasner, H., and Iscoe, N. 1988. A field study of the software design process for large systems. *Communications of the ACM* 31(11): November, 1268–1287.

Damian, D., Zowghi, D., Vaidyanathasamy, L., and Pal, Y. 2004. An industrial case study of immediate benefits of requirements engineering process improvement at the Australian Center for Unisys Software. *International Journal of Empirical Software Engineering* 9(1–2): March, 45–75.

El Emam, K., and Birk, A. 2000. Validating the ISO/IEC 15504 measure of software requirements analysis process capability. *IEEE transactions on Software Engineering* 26(6): June, 544–566.

El Emam, K., Dourin, J., and Melo, W. 1998. In: K. El Emam, J. Dourin and W. Melo, (eds.), *SPICE: The Theory and Practice of Software Process Improvement and Capability Determination*. IEEE CS Press.

Hall, T., Beecham S., and Rainer, A. 2002. Requirements problems in twelve software companies: An empirical analysis. *IEE Proceedings-Software* 149(5): 153–160.

Halligan, R. 2000. *TAA's SE Training Courseware*. Halligan Corporation Pty Ltd.

Herbsleb, J., and Goldenson, D. 1996. A systematic survey of CMM experience and results. In *Proceedings of the International Conference on Software Engineering*. ACM Press, pp. 323–330.

Humphrey W., Snyder, T., and Willis, R. 1991. Software process improvement at hughes aircraft. *IEEE Software* 8(4): 11–23.

Lauesen, S., and Vinter, O. 2001. Preventing requirement defects: An experiment in process improvement. *Requirements Engineering Journal* 6: 37–50.

Mayo, E. 1933. *The Human Problems of an Industrial Civilization*. New York, NY: Macmillan Co.

Paulk, M. 1994. A comparison of ISO 9001 and the capability maturity model for software. *TR: CMU/SEI-94-TR-12*.

Paulk, M., Curtis, B., Chrissis, M. B., and Weber, C. V. 1993. Capability Maturity Model Version 1.1. *IEEE Software* 10(4): 18–27.

Quality standards: Quality management and quality assurance standards. 1987. *Int. Org. for Standardization*.

Robertson, S., and Robertson, J. 1999. *Mastering the Requirements Process*. London: Addison-Wesley.

SEI, 1995: *Software Engineering Institute: The Capability Maturity Model: Guidelines for Improving the Software Process*. Addison Wesley.

Sommerville. 1996. *Software Engineering*. England: Addison-Wesley.

Sommerville, and Sawyer, P. 1997. *Requirements Engineering: A Good Practice Guide*. England: John Wiley & Sons.

The Standish Group. *Chaos*. 1997. At http://www.standishgroup.com/chaos.html.

Wohlwend, H., and Rosenbaum, S. 1993. Software process improvement in an international company. In *Proceedings of the International Conference on Software Engineering*, pp. 212–220.

Dr. Daniela Damian is an Assistant Professor in the Department of Computer Science at the University of Victoria, where she holds the NSERC University Faculty Award. Her research lies in the areas of Software Engineering, Human-Computer Interaction and Computer-Supported Cooperative Work. While interested in studying collaborative software engineering in geographically distributed software teams in general, her particular research focus is on requirements management activities and process improvement in global software projects. Daniela leads SEGAL (Software Engineering Global interAction Laboratory), where together her research group she is researching methodological as well as technological support for collaborative tasks (and in particular requirements management) in global software development. Her goal is to achieve an understanding of the challenges that software companies face because of geographical, temporal and cultural differences between software stakeholder groups, and to design and evaluate potential solutions to overcome these challenges.

James Chisan I am a Master's student currently conducting research at the Software Engineering Global interActions Lab (SEGAL) at the University of Victoria. My interests are primarily in requirements engineering as it pertains to improving software engineering. I endeavour to improve our understanding of how requirements engineering affects software development and then how to optimize and tailor requirements processes to address particular business objectives. I am also investigating the use of awareness in enhancing the efficacy and efficiency of requirements engineering by leveraging the intermediate artifacts of software development to provide developers up-to-date requirements information.

Lakshminarayanan Vaidyanathasamy (LNV Samy) is the Director of Australian Centre of Unisys Software, which is a Unisys Organisation and has over 20 years experience in IT and Software Engineering. LNV Samy has a Bachelor of Engineering - Honours from University of Madras, India, Master of Technology in Software Engineering from Macquarie University, Australia and Master of Business Administration from Macquarie Graduate School of Management, Australia. LNV Samy is a member of Australian Computer Society, American Society for Quality and IEEE. LNV Samy has published many papers in Software Engineering.

Yogendra Pal is a Senior Project Manager at the Australian Centre of Unisys Software responsible for the research and development of the Enterprise Application Environment products for global distribution. Yogendra has a Bachelor of Engineering from RMIT and is a member of IE Aust. He has been involved in Project Management of development projects in Defence and Commercial sectors for the past 15 years and has been at the forefront of introducing and managing Process Improvement Initiatives within the SEI-CMM Framework for various organisations. Yogendra has conducted research on Process Improvement and published papers on this research, primarily in the area of introduction of Requirements Management processes in the commercial R&D world.