

12-5-2007

Improving Practices in a Small Software Firm: An Ambidextrous Perspective

Nannette Napier

Follow this and additional works at: http://scholarworks.gsu.edu/cis_diss

Recommended Citation

Napier, Nannette, "Improving Practices in a Small Software Firm: An Ambidextrous Perspective." Dissertation, Georgia State University, 2007.

http://scholarworks.gsu.edu/cis_diss/18

This Dissertation is brought to you for free and open access by the Department of Computer Information Systems at ScholarWorks @ Georgia State University. It has been accepted for inclusion in Computer Information Systems Dissertations by an authorized administrator of ScholarWorks @ Georgia State University. For more information, please contact scholarworks@gsu.edu.

Permission to Borrow

In presenting this dissertation as a partial fulfillment of the requirements for an advanced degree from Georgia State University, I agree that the Library of the University shall make it available for inspection and circulation in accordance with its regulations governing materials of this type. I agree that permission to quote from, or to publish this dissertation may be granted by the author or, in his/her absence, the professor under whose direction it was written or, in his absence, by the Dean of the Robinson College of Business. Such quoting, copying, or publishing must be solely for scholarly purposes and does not involve potential financial gain. It is understood that any copying from or publication of this dissertation which involves potential gain will not be allowed without written permission of the author.

Nannette Patterson Napier

Notice to Borrowers

All dissertations deposited in the Georgia State University Library must be used only in accordance with the stipulations prescribed by the author in the preceding statement.

The author of this dissertation is:

Name: Nannette Napier

Address: 2423 Idlewood Way, Snellville, Georgia 30078

The director of this dissertation is:

Name: Dr. Lars Mathiassen

Department: Center for Process Innovation

Address: 4th Floor, 35 Broad Street NW, Atlanta, Georgia 30303

Users of this dissertation not regularly enrolled as students at Georgia State University are required to attest acceptance of the preceding stipulations by signing below. Libraries borrowing this dissertation for the use of their patrons are required to see that each user records here the information requested.

Name of User

Address

Date

IMPROVING PRACTICES IN
A SMALL SOFTWARE FIRM:
AN AMBIDEXTROUS PERSPECTIVE

BY

NANNETTE PATTERSON NAPIER

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree
of
Doctor of Philosophy
in the Robinson College of Business
of
Georgia State University

GEORGIA STATE UNIVERSITY
ROBINSON COLLEGE OF BUSINESS
2007

Copyright by
Nannette Patterson Napier
2007

ACCEPTANCE

This dissertation was prepared under the direction of the candidate's Dissertation Committee. It has been approved and accepted by all members of that committee, and it has been accepted in partial fulfillment of the requirements for the degree of Doctor in Philosophy in Business Administration in the Robinson College of Business of Georgia State University.

Dean: H. Fenwick Huss
Robinson College of Business

Dissertation Committee:

Chair: Dr. Lars Mathiassen

Dr. Sandeep Purao

Dr. Bala Ramesh

Dr. Vijay Vaishnavi

ACKNOWLEDGEMENTS

Although the title page of this dissertation contains one name, this work would be impossible without the efforts of the many people who have supported me throughout this journey. I'd like to begin by acknowledging the tireless work and patience of my dissertation supervisor, Dr. Lars Mathiassen. At various times through the years, he has served as critic, counselor, copy editor, and mentor. He could always be counted upon to give 100% to our efforts, respond quickly to my written drafts, see the positive when I was in doubt, and challenge me to clarify my thoughts and sharpen my arguments. Through this relationship, I have learned valuable lessons about conducting and writing rigorous, relevant research, and I look forward to continued collaboration in the future.

I also want to express my sincere gratitude to Sandeep Puroo, Bala Ramesh, and Vijay Vaishnavi who served on my dissertation committee. At several critical moments, they stepped in to provide important assistance so that this dissertation could advance to the next stage. In addition, this research would not have been possible without the help of the employees of *TelSoft*. To maintain the anonymity of the organization, I will withhold names; however, I especially want to thank the managers represented as part of the Problem Solving Team and Software Coordination Group who trusted us to observe and intervene in their daily operations.

Over the years in the program, I have had the pleasure of collaborating with a number of researchers. I would first like to thank the co-authors who have worked with me on the papers that form Part II of this dissertation: Roy Johnson, Jonathan Kim, and Dan Robey. Their contributions have greatly improved the quality of the work submitted, and I look forward to working with them on furthering this work. I am particularly indebted to Roy Johnson for his sponsorship through the GAANN grant which provided financial support through the first three

years of the program. I would next like to thank the researchers I have collaborated with on other research papers (listed alphabetically): Lily Chen, Yi Ding, Steve Du, Mark Keil, Melody Moore-Jackson, Adriane Randolph, Veda Storey, Carl Stucke, Hiro Takeda, and Felix Tan. In various ways, you have each served as mentors and I have learned from each of you something about how to be a better scholar. Thank you.

Many others inside and outside of the university have played an integral role in helping me through this experience and preparing me for a career in academia: the instructors at Georgia State University, other CIS PhD students that arrived with me in 2003, the KPMG PhD Project, Southern Regional Education Board (SREB), and African American Researchers in Computing Sciences (AARCS). Last but not least, I want to acknowledge and thank my family for their acceptance, support, and encouragement as I pursued this goal: my parents (James and Irma Patterson), husband (Junior Napier), and kids (Chantal, Sean, Alyssa, and Carlton).

ABSTRACT

IMPROVING PRACTICES IN
A SMALL SOFTWARE FIRM:
AN AMBIDEXTROUS PERSPECTIVE

By

NANNETTE PATTERSON NAPIER

AUGUST 29, 2007

Committee Chair: Dr. Lars Mathiassen

Major Department: Computer Information Systems

Despite documented best practices and specialized tools, software organizations struggle to deliver quality software that is on time, within budget, and meets customer requirements. Managers seeking improved software project outcomes face two dominant software paradigms which differ in their emphasis on upfront planning, customer collaboration, and product documentation: plan-driven and agile. Rather than promoting one approach over the other, this research advocates improving software management practices by developing the organization's ambidextrous capability. Ambidextrous organizations have the ability to simultaneously succeed at two seemingly contradictory capabilities (e.g. discipline and agility) which leads to enhanced organizational performance.

Overall, this study asks the question: How can an ambidextrous perspective facilitate improvement in software practices? Driven by this question, and based on a two year action research study at a small software firm, *TelSoft*, the objectives of this research are to:

1. Identify dualities involved in improving software practices
2. Design interventions based on these dualities to improve software practices
3. Explore the process of becoming an ambidextrous software organization

The resulting dissertation consists of a summary and four papers that each identify and address particular dualities encountered during software process improvement. The first paper asserts that both process-driven and perception-driven inquiry should be used during assessment of software practices, presents a model that shows how this combination can occur, and demonstrates the use of this model at *TelSoft*. The second paper explicates two theories for understanding and resolving issues in requirements engineering practice – repeat-ability and response-ability – and argues for the need to negotiate between the two. The third paper identifies a tension between managing legacy and current processes and proposes a model for

software process reengineering, a systematic process for leveraging legacy processes created during prior SPI efforts. Finally, the fourth paper applies the theoretical lens of ambidexterity to understand the overall change initiative in terms of the tension between alignment and adaptability.

The study used a variety of data sources to diagnose software practices, including semi-structured interviews, software process documents, meeting interactions, and workshop discussions. Subsequently, we established, facilitated, and tracked focused improvement teams in the areas of customer relations, requirements management, quality assurance, project portfolio management, and process management. Furthermore, we created and trained two management teams with responsibility for ongoing management of SPI and project portfolio management respectively. We argue that these activities improved software practices at *TelSoft* and provided a stronger foundation for continuous improvement.

Keywords: Ambidexterity, software process improvement (SPI), action research, requirements engineering assessment, action planning, software process reengineering, software management.

Table of Contents

| | |
|--|-----|
| Part I: Research Summary | 11 |
| Chapter 1: Research Focus | 12 |
| Chapter 2: Theoretical Background | 17 |
| Chapter 3: Research Approach..... | 24 |
| Chapter 4: Review of Results..... | 36 |
| Chapter 5: Discussion | 44 |
| References..... | 56 |
| Part II: Research Papers | 62 |
| Paper 1: Combining Perceptions and Processes | 63 |
| Paper 2: Negotiating Repeat-ability and Response-ability..... | 78 |
| Paper 3: Managing Legacy and Current Processes | 102 |
| Part III: Problem Solving Cycle..... | 159 |
| Prologue | 160 |
| Chapter 1: Initiating | 162 |
| Chapter 2: Diagnosing | 163 |
| Chapter 3: Intervention Cycle 1 | 168 |
| Chapter 4: Intervention Cycle 2 | 173 |
| Chapter 5: Learning | 175 |
| Appendix A: Comprehensive List of Problem Solving Documents..... | 177 |
| Appendix B: Problem Solving Cycle Documentation..... | 182 |
| References..... | 259 |

Part I: Research Summary

Chapter 1: Research Focus

1.1 Research Domain

Despite documented best practices and specialized tools, software organizations struggle to deliver quality software that is on time, within budget, and meets customer requirements. In fact, the Standish Group (2004) reports that 53% of all information technology (IT) projects are late or over budget; an additional 18% either fail outright or are cancelled prior to completion. All indications are that the environment in which software is developed will continue to challenge rather than ameliorate the situation. Increasingly, the business environment is characterized by frequent requirements changes, rapid technological advances, and time-to-market pressures (Ramesh, Pries-Heje et al. 2002).

Given this dismal state of affairs, what strategies should software managers use to increase the likelihood of successful project outcomes? In general, managers face two dominant software development and improvement paradigms which differ in their emphasis on upfront planning, customer collaboration, and product documentation: plan-driven and agile. Plan-driven approaches, such as the Software Capability Maturity Model (SW-CMM), Bootstrap (Kuvaja and Bicego 1994), or SPICE (Rout 1995), emphasize discipline through documentation of project milestones, requirements, and designs; such approaches are most appropriate for large products and teams, mission-critical systems with stable requirements, and a culture that thrives on order (Boehm 2002; Boehm and Turner 2004). Agile approaches, such as extreme programming (Beck 1999), Crystal Methods (Cockburn 2000), or adaptive software development (Highsmith 2000), emphasize responsiveness and flexibility by giving priority to people and prototypes over processes and documentation (Agile Alliance 2001; Highsmith and Cockburn 2001); these approaches are most appropriate for small products and teams where there are highly dynamic requirements, flexible, knowledgeable experts, and a culture that is amenable to changing situations (Boehm 2002; Boehm and Turner 2004).

In some cases, characteristics such as team size, developer skills, company culture, and project goals clearly indicate whether plan-driven or agile methods are more appropriate (Boehm 2002; Boehm and Turner 2004). Increasingly, however, clear cut situations are falling away to an environment in which managers seek the benefits of both discipline and agility and therefore need to take advantage of techniques associated with both plan-driven and agile methods. Some studies have examined how agile approaches can comply with the guidelines of the SW-CMM and its successor, Capability Maturity Model Integration (CMMI) (Paulk 2001). Empirical case studies have also begun to appear that show how this combination can occur (Baker 2005; Salo and Abrahamsson 2005). However, the literature is only beginning to provide guidance on combining these approaches.

The effective integration of opposing capabilities would, in effect, require software firms to become ambidextrous. Ambidexterity is the ability to pursue simultaneously contradictory capabilities such as exploration-exploitation (Tushman and O'Reilly III 1996), alignment-adaptability (Gibson and Birkinshaw 2004), flexibility-efficiency (Adler, Goldoftas et al. 1999), and flexibility-rigor (Lee, DeLone et al. 2006). Ambidextrous organizations compete by optimizing efficiency, cost, and incremental innovation while at the same time exhibiting flexibility, speed, and radical innovation (Tushman and O'Reilly III 1996). Moreover, studies

have begun to provide empirical support for the “ambidexterity hypothesis” (i.e. that increased ambidexterity leads to enhanced organizational performance) (Gibson and Birkinshaw 2004; He and Wong 2004). In the context of global information systems (IS) project teams, Lee et al. (2006) found that successful teams were ambidextrous, using coping strategies that exhibited both flexibility and rigor. Thus, focusing on becoming ambidextrous could serve as an alternative means for software organizations to improve.

Although the anticipated benefits are significant, achieving ambidexterity is by no means straightforward. Each contradictory capability requires different and often incongruent systems, processes, and beliefs, thereby creating conflicts and dilemmas that are challenging to resolve (Tushman and O'Reilly III 1996; Floyd and Lane 2000; Gibson and Birkinshaw 2004). How, then, can managers design and develop ambidextrous organizations? Within the organizational management literature, two general approaches have been suggested: structural and contextual ambidexterity. With structural ambidexterity, managers create separate business units within the organization which specialize in one required capability, and the top management team bears responsibility for coordinating contributions of the two units to achieve ambidexterity at the organizational level (Gibson and Birkinshaw 2004). With contextual ambidexterity, the responsibility for achieving ambidexterity is shared by members within a single business unit. To create a high performing business unit, the top management team is advised to create an organizational context which facilitates both alignment and adaptability through appropriate performance management and social support (Gibson and Birkinshaw 2004).

The IS literature on ambidextrous software organizations lags behind the organizational management literature on ambidexterity in at least two important ways. First, IS researchers are still at the definitional stages of understanding the competing capabilities that software organizations must master to become ambidextrous, such as flexibility-rigor (Lee, DeLone et al. 2006; Lee, DeLone et al. 2007) and agility-discipline (Boehm 2002; Boehm and Turner 2004). More work can be done to clarify relevant dualities which can then form the foundation for future research. Second, IS researchers have chiefly adopted the language of structural ambidexterity in designing ambidextrous solutions. For example, consistent with structural ambidexterity, Vinekar et al. (2006) define ambidextrous systems development organizations as consisting of a traditional, plan-driven subunit and an agile subunit. However, IS researchers have only briefly mentioned contextual ambidexterity as an appropriate means for becoming ambidextrous. These two factors highlight the need for the IS literature to deepen its appreciation for the dualities associated with ambidextrous software organizations and to broaden its understanding of the ways in which ambidexterity can be achieved.

1.2 Research Design

Overall, this study asks the question: *How can an ambidextrous perspective facilitate improvement in software practices?* Accordingly, this research examines the *dualities* associated with ambidexterity, the *design* of interventions to resolve these dualities, and the *process* of becoming ambidextrous. Hence, the following research objectives are investigated:

1. Identify *dualities* involved in improving software practices
2. *Design* interventions based on these dualities to improve software practices
3. Explore the *process* of becoming an ambidextrous software organization

Taking an ambidextrous perspective, this work embraces the idea of duality. A duality highlights two elements that at the same time exhibit tension and complement each other:

“A duality is a single conceptual unit that is formed by two inseparable and mutually constitutive elements whose inherent tension and complementarity give the concept richness and dynamism.” (Wenger 1998, p. 66)

Each element of the duality can be present, but more or less to some extent. By putting them together, we acknowledge that there is a relationship between the two and can focus on their interactions (e.g. how discipline influences agility and vice versa).

To meet these research objectives, the Center for Process Innovation (CEPRIN) at Georgia State University (GSU) initiated an action research project with *TelSoft*, a small software company wanting to improve its software practices. Small software organizations, independent companies consisting of less than 50 software developers and projects of fewer than 20 people (Software Engineering Institute 2006), represent an excellent setting for studying dualities involved in improving software practices as well as ambidexterity. Key characteristics of small software organizations include reliance on a few projects servicing known customers, overburdened employees performing multiple roles, and a tendency to rely on individual judgment over standardized processes (Horvat, Rozman et al. 2000). Furthermore, the culture in these companies attracts employees with a desire for autonomy and a disdain against heavy standards (Software Engineering Institute 2006). To be successful, these organizations must be agile and adapt quickly to environmental changes and frequent customer requests (Ramesh, Pries-Heje et al. 2002; Mathiassen and Vainio 2007). At the same time, they can benefit from increasing discipline and alignment across all employees; if processes are left undocumented and to the discretion of individual preferences, practices may not be efficient and important knowledge may be lost when individuals decide to leave the organization. Therefore, managers within small software organizations must learn to effectively balance discipline and agility while making adjustments for the specific context in which they operate (Boehm and Turner 2004).

The overall research methodology is collaborative practice research (CPR), a form of action research that emphasizes methodological pluralism and collaboration between researchers and practitioners (Mathiassen 2002). The goal of action research is to “contribute both to the practical concerns of people in an immediate problematic situation and to the goals of social science by joint collaboration” (Rapoport 1970). Action research can hence be conceptualized as containing two concurrent and interacting learning cycles – a problem solving cycle that addresses the practical concerns and a research cycle that addresses the need for scientific knowledge on the part of the researchers (McKay and Marshall 2001). Over the two years of this collaboration, a number of interventions were designed to increase ambidextrous capability and improve organizational performance. Through close collaboration with our industry partner, *TelSoft*, we used theory to influence the organizational change agenda and to observe the process of change over time. The final phase of the research project evaluated the effectiveness and impact of these interventions. Overall, the collected process data (Langley 1999) permitted investigation of becoming a more ambidextrous software organization.

TelSoft was founded in 1971, with the mission to be a premier software services firm in the telecommunications and utility industries. The company has approximately 500 employees with fewer than 50 dedicated to building and customizing geographic information systems (GIS) software. *TelSoft* emerged as an ideal research site because they had many troubled software projects: software releases were shipped late, ran over budget, and contained deviations from agreed upon requirements. *TelSoft*'s customers frequently requested requirements changes; however, important stakeholders within *TelSoft* were not always informed of these changes in a timely fashion. Because the company attributed these problems to issues with its processes for discovering, managing, and changing requirements, *TelSoft*'s management initially requested that we focus on the requirements engineering (RE) process. However, when the diagnosis revealed problems in areas such as software process management, project portfolio management, and software vision management, we expanded our research interests to focus more broadly on improving software practices.

To guide the activities in the problem solving cycle, we adopted the IDEAL model (McFeeley 1996) – an acronym for Initiating, Diagnosing, Establishing, Acting, and Learning – to improve software practices. Each phase of this process provides an opportunity to make research contributions (e.g., identifying problems not sufficiently addressed in the literature, proposing methods for solving those problems, and studying change processes over time). During the diagnosing phase, we identified alternative assessment practices and proposed a method for combining process-based and perception-based evaluation. In support of the establishing phase, we explored the assumptions underlying the tensions of plan-driven and agile approaches to RE. During the acting phase, we proposed a process for integrating legacy software processes into software process improvement (SPI) by establishing a systematic process management discipline. During the learning phase, we reflected on the impact of the overall change process through the lens of contextual ambidexterity. We argue that these activities improved software practices at *TelSoft* and provided a stronger foundation for continuous improvement.

1.3 Dissertation Outline

The dissertation consists of three parts. In Part I *Research Summary*, we describe the objectives of the study in chapter 1, introduce the research domain in chapter 2, detail the research methodology in chapter 3, review the main results in chapter 4, and summarize the contributions in chapter 5.

In Part II *Research Papers*, we present the results from the research cycle: the full text of the four papers that comprise the dissertation. Each research paper selects a specific area within the domain of improving software practices, reviews relevant literature, uses data collected from one or more phases of the action research cycle, applies a specific data analysis method, and contributes to both research and practice as summarized in Table 1.

Table 1: Summary of Research Contributions

| Research Paper | Short Description | Main Contribution |
|-----------------------|---|---|
| Paper 1 | Combining Perceptions and Processes | Model for assessing RE practice which values insights from both process models and perceptions of key stakeholders (Napier, Mathiassen et al. 2006) |
| Paper 2 | Negotiating Repeat-ability and Response-ability | Two theories for understanding and resolving issues in RE practice: repeat-ability and response-ability (Napier, Mathiassen et al. 2006) |
| Paper 3 | Managing Legacy and Current Processes | Model for “Software Process Reengineering” that allows organizations to leverage legacy software processes when reengaging in improvement after initial failure (Napier, Kim et al. under review) |
| Paper 4 | Becoming Ambidextrous | Application of contextual ambidexterity to understand the overall change initiative in terms of the tension between alignment and adaptability (Napier, Mathiassen et al. under review) |

In Part III *Problem Solving Cycle*, we document the problem solving efforts at *TelSoft*, including the initial memorandum of agreement and the interview guides used during diagnosis and learning phases. A comprehensive list of documents produced during the collaboration is also provided.

Chapter 2: Theoretical Background

In this chapter, we summarize the current literature on ambidexterity and relate it to the specific challenges of small software organizations.

2.1 Ambidexterity

In this section, we review the organizational management literature on dualities associated with ambidexterity, proposed designs for achieving ambidexterity, and the process for increasing ambidextrous capability within an organization.

Dualities. For many years, researchers have been captivated by the tension associated with exploitation and exploration. Exploitation is associated with incremental improvement, learning through local search, refining existing products, and reuse of existing routines whereas exploration is associated with more radical improvement, learning through experimenting with technologies and ideas from outside the organization, and new product development (March 1991; Baum, Li et al. 2000; Benner and Tushman 2003). In short, exploitation is learning along the existing trajectory while exploration is learning that follows a new trajectory (Gupta, Smith et al. 2006).

The relative investment made in exploitation and exploration is a strategic choice with no predefined answer. On the one hand, organizations emphasizing exploitation can fall into a competency trap in which they get better and better at the same thing without being able to move to the next stage; whereas, organizations emphasizing exploration can fall into a failure trap in which they are unable to fully capitalize on the innovations they start (March 1991). To avoid the negatives of either one, organizations have been advised to strive for ambidexterity – the ability to simultaneously succeed at two seemingly contradictory capabilities such as the dualities of exploration-exploitation (Tushman and O'Reilly III 1996), alignment-adaptability (Gibson and Birkinshaw 2004), and flexibility-efficiency (Adler, Goldoftas et al. 1999).

Studies have begun to provide empirical support for the positive relationship between ambidexterity and organizational performance. Based upon surveys of 4,195 individuals within 41 business units of ten multinational firms, Gibson and Birkinshaw (2004) found a positive and significant correlation between ambidexterity and organizational performance. Focusing on the context of technological innovations, He and Wong (2004) found that the interaction of explorative and exploitative innovation strategies was positively related to sales growth. While some argue that there are contexts in which ambidexterity may not be necessary (Gupta, Smith et al. 2006), these results demonstrate the benefits of ambidexterity.

Design. Various definitions related to ambidexterity have been offered in the literature (see Table 2 for a summary). A business unit's ambidexterity has been described as having high levels of both exploratory and exploitative innovations (Jansen, van Den Bosch et al. 2005). Ambidextrous organizations are expected to compete successfully both in mature markets with existing customers by optimizing efficiency, cost, and incremental innovation as well in emerging markets with new customers by exhibiting flexibility, speed, and radical innovation (Tushman and O'Reilly III 1996). Recently, Gibson and Birkinshaw (2004) have distinguished

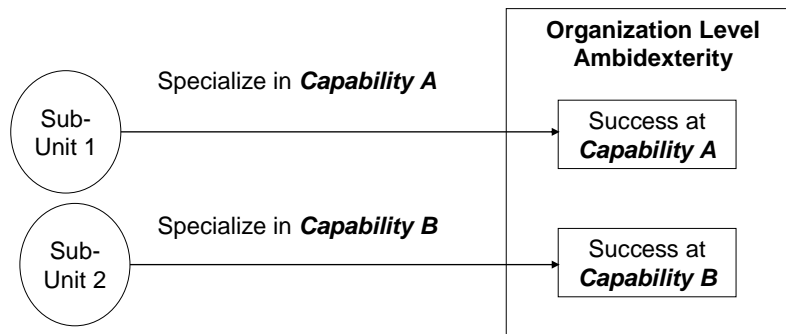
between structural and contextual ambidexterity in terms of the strategies used for achieving success at the dual capabilities of A and B.

Table 2: Definitions of Ambidexterity

| <i>Term</i> | <i>Definition</i> |
|----------------------------------|--|
| Ambidextrous organizational form | “Composed of multiple tightly coupled subunits that are themselves loosely coupled with each other. Within subunits, the tasks, culture, individuals, and organizational arrangements are consistent, but across subunits tasks and culture are inconsistent and loosely coupled.” (Benner and Tushman 2003, p. 247) |
| Ambidextrous organizations | [Have] “the ability to simultaneously pursue both incremental and discontinuous innovation and change” (Tushman and O’Reilly III 1996, p. 24) |
| Business unit’s ambidexterity | “Units characterized by high levels of exploratory and exploitative innovations” (Jansen, van Den Bosch et al. 2005, p. 352) |
| Contextual ambidexterity | “The behavioral capacity to simultaneously demonstrate alignment and adaptability across an entire business unit” (Gibson and Birkinshaw 2004, p. 209) |
| Structural ambidexterity | “Organizations manage trade-offs between conflicting demands by putting in place ‘dual structures’, so that certain business units – or groups within business units – focus on alignment, while others focus on adaptation (Duncan 1976)” (quoted in Gibson and Birkinshaw 2004, p. 209) |

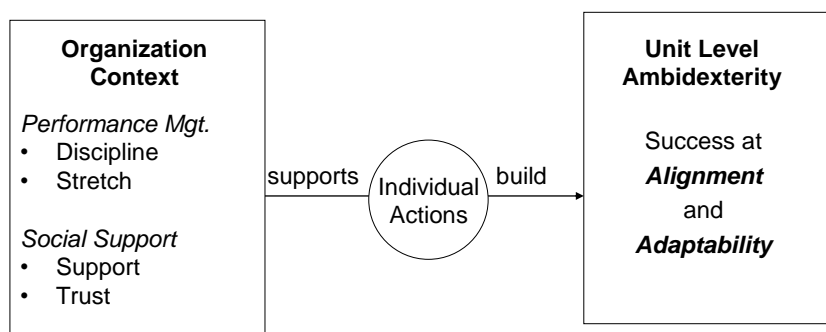
With structural ambidexterity managers create separate business units within the organization, each with a specialization in either A or B (Gibson and Birkinshaw 2004), see Figure 1. The top management team (TMT) ensures coordination between the two units such that the most promising innovations from the exploratory unit can mature and be effectively incorporated by the organization’s exploitative unit. The rationale for this separation is that the systems, processes, and beliefs required for exploration and exploitation are too incongruent to be found within the same unit. Organizations designed with this structure have been described as having an ambidextrous organizational form (Benner and Tushman 2002). Although case studies of various multinational organizations have illustrated the benefits of structural ambidexterity (Birkinshaw and Gibson 2004; O’Reilly III and Tushman 2004), this approach may not be suitable for companies with limited resources and dynamic environments.

Figure 1: Structural Ambidexterity (Gibson and Birkinshaw 2004)



With contextual ambidexterity the responsibility of achieving ambidexterity is shared amongst individual employees within a specific business unit, see Figure 2. Contextual ambidexterity requires simultaneous success at both alignment – capacity of employees within the business unit to work toward common goals – and adaptability – capacity of the business unit to quickly change in response to dynamic market conditions (Gibson and Birkinshaw 2004). This perspective recognizes that it is the day-to-day decisions of individual employees that shape alignment and, therefore, the TMT is charged with creating a facilitating environment which will lead to contextual ambidexterity. Following Ghoshal and Bartlett (1994), Gibson and Birkinshaw (2004) identify salient aspects of the organization context which can be manipulated: performance management and social support. The performance management context represents systems, processes, and beliefs related to meeting performance objectives set by the organization’s management (Gibson and Birkinshaw 2004). Discipline is an attribute that encourages people to voluntarily meet those objectives whereas stretch is an attribute that encourages people to strive for even more ambitious goals (Ghoshal and Bartlett 1994). The social support context represents systems, processes, and beliefs associated with member relationships (Gibson and Birkinshaw 2004). Trust is an attribute of the organizational context that encourages people to rely on one another whereas support is an attribute that empowers people to lend assistance to others (Ghoshal and Bartlett 1994).

Figure 2: Contextual Ambidexterity (Gibson and Birkinshaw)



From this review, we learn that ambidexterity requires more than just “success at A” plus “success at B.” It also requires the ability to coordinate and integrate the two. From the perspective of structural ambidexterity, integration is the responsibility of TMT allowing subunits within an organization to specialize and focus on specific concerns (Duncan 1976; Gibson and Birkinshaw 2004; Jansen, van Den Bosch et al. 2005). From the perspective of contextual ambidexterity, each individual employee is responsible for figuring out how to coordinate and integrate a concern for A with a concern for B (Gibson and Birkinshaw 2004).

Process. The process of building contextual ambidexterity is described as “complex, causally ambiguous, widely dispersed, and quite time-consuming” (Gibson and Birkinshaw 2004, p. 209-210). As we found no empirical studies that attempt to further describe this process, many practical questions related to achieving ambidexterity have not been addressed. Specifically, how can organizations develop and engage in ambidextrous practices and create and sustain organizational contexts that facilitate such practices? What enablers and barriers can managers expect and how might those be leveraged and resolved, respectively? How long does it take to become ambidextrous, and are there specific shortcuts which enable this process to go more quickly? Birkinshaw and Gibson (2004) provide some general lessons on where and how organizations can start developing ambidextrous capabilities: diagnose the organizational context; change key aspects of the context; ensure communication about ambidexterity throughout the organization; consider contextual and structural ambidexterity; and empower employees throughout the organization to participate. While these lessons serve as a starting point for understanding how to develop ambidexterity, much more is needed to understand how context and managerial practices interact over time and shape each other as organizations strive to become ambidextrous.

Most research focuses on measurement issues and supporting the relationship between ambidexterity and organizational performance. Researchers typically measure ambidexterity by measuring each part of a duality separately and then aggregating by multiplying the two together (Gibson and Birkinshaw 2004; Jansen, van Den Bosch et al. 2005), taking the difference (He and Wong 2004), or taking the sum (Lubatkin, Simsek et al. 2006). Then, researchers take snapshot measures of ambidexterity and performance to study whether there appears to be a relationship. While determining reliable measures is important, a limitation is that the work is largely cross sectional and based upon interviews and surveys. Such cross sectional research does not allow a look at how ambidexterity within an organization changes over time. Another important source for understanding organizational ambidexterity is therefore to look at actual work practices within organizations and how those practices change over time (Barley and Kunda 2001); collecting and analyzing longitudinal, qualitative data can provide insights into how and why people in organizations act and interact over time (Langley 1999).

2.2 Ambidextrous Software Organizations

In this section, we review the software literature on dualities associated with ambidexterity, proposed designs for achieving ambidexterity, and the process for increasing ambidextrous capability within software organizations.

Dualities. One perspective which has strongly influenced software organizations is the contrast between plan-driven and agile development approaches (Boehm 2002). Boehm and Turner (2004) describe various development and improvement approaches as varying along a planning spectrum based upon emphasis in upfront planning and documentation. At the most rigid end of the planning spectrum is inch-pebble management where every aspect of projects is planned and micromanaged. At the most lax end of the planning spectrum are hackers who plan nothing and shun documentation. Realistically, most development methods fall somewhere in between depending upon how the approach is interpreted and implemented within a specific organization.

With plan-driven approaches, the emphasis is on codifying important knowledge and creating reliable processes, and the underlying value is discipline (Boehm and Turner 2004). For example, with the SW-CMM, software processes are key to increasing organizational maturity: mature software organizations define processes and tailor them to specific projects; they establish an infrastructure for managing software processes; and they use quantitative measures to support continuous development of software processes (Paulk, Curtis et al. 1993; Paulk, Weber et al. 1995; CMMI Product Team 2002). Organizational maturity is indicated by satisfying key process areas associated with five levels: initial (1), repeatable (2), defined (3), managed (4), and optimizing (5); furthermore, organizations are advised on the order in which these key process areas should be improved. While plan-driven approaches can enhance predictability and provide high quality assurance, there are a number of risks that should be considered. First, such approaches can be expensive to put into practice; and adopting industry best practices may not fit closely the wants and needs of the organization (Iversen, Nielsen et al. 2002). Second, changing technical, market, or customer requirements could make the documented processes obsolete; therefore, the organization must also have processes in place to deal with these changes. Third, software engineers may resist the imposed structure provided by these approaches, perceiving these standards as a loss of autonomy or a hindrance to the creative development process (Adler, McGarry et al. 2005).

With agile approaches, the emphasis is on rapid change facilitated by close collaboration between customers and the development organization to continually refine and prioritize requirements; the underlying value is agility (Boehm and Turner 2004). Because requirements are expected to change, agile development occurs in short, iterative development cycles, and there is little attempt to predict future requirements. For example, in the Scrum software development methodology (Rising and Janoff 2000; Schwaber and Beedle 2001), small teams focus on producing working code during sprints, short time period punctuated by a client demonstration of progress. To accomplish this, there are daily scrum meetings led by a scrum master where developers state progress since the last meeting, list obstacles, and state goals for the day. When each sprint closes, it represents a new opportunity for planning and incorporating requirements from the backlog or changes identified by customers during the product demonstration. Although agile methods can speed time to market, there are risks associated with reliance on agile approaches. A short-term focus may lead to an inflexible architecture that does not meet future needs; emphasis on early success may lead to rework or code that does not scale; and customer liaison may not have sufficient time, commitment, or knowledge to guide projects (Boehm 2002; Boehm and Turner 2004).

A second perspective on the dualities within software organizations has been investigated within the context of managing globally distributed software development project teams (Lee, DeLone et al. 2006; Lee, DeLone et al. 2007). The two dualities mentioned here are IS project rigor and IS project agility. Consistent with plan-driven approaches, IS project rigor (Lee, DeLone et al. 2007) emphasizes adherence to defined processes and standards across the project. Indications of rigor include detailed project plans, documented software development processes, common technological environment, and formal communications. Consistent with agile approaches, IS project agility (Lee, DeLone et al. 2007) emphasizes anticipating, sensing, and efficiency responding to changing system requirements. IS project agility is indicated by quick turnaround on change requests. Being agile also means such changes can be accomplished with lower cost. Empirical investigations with global IS project teams have indicated the most successful teams are ambidextrous. In particular, successful project teams required agility to remain alert to any required changes and used rigor to ensure that those changes were systematically applied across the project team (Lee, DeLone et al. 2006).

Design. Two primary approaches for designing ambidextrous software organizations have been offered: one based upon risk management and the other on structural ambidexterity.

Using risk management, managers are advised to select an appropriate approach based upon project and company characteristics. Boehm and Turner (Boehm 2002; Boehm and Turner 2004) advise that project characteristics such as developer skill set, customer availability, and requirements predictability be evaluated and used to pick the approach that best fits the situation. If the main goals are speed and customer satisfaction, agile approaches may be more appropriate; however, if the main goal is a quality product and requirements are stable, then plan-driven approaches may be more suitable (Boehm 2002). When a combination of project characteristics or goals is present, the need for ambidexterity occurs, and managers are advised to use risk analysis techniques to determine the appropriate mixture of discipline and agility. Given that additional costs are associated with developing and maintaining each capability, managers should not assume that ambidexterity is necessary:

“Both agile and plan-driven methods have a home ground of project characteristics in which each clearly works best, and where the other will have difficulties. Hybrid approaches that combine both methods are feasible and necessary for projects that combine a mix of agile and plan-driven home ground characteristics.” (Boehm 2002, p. 69)

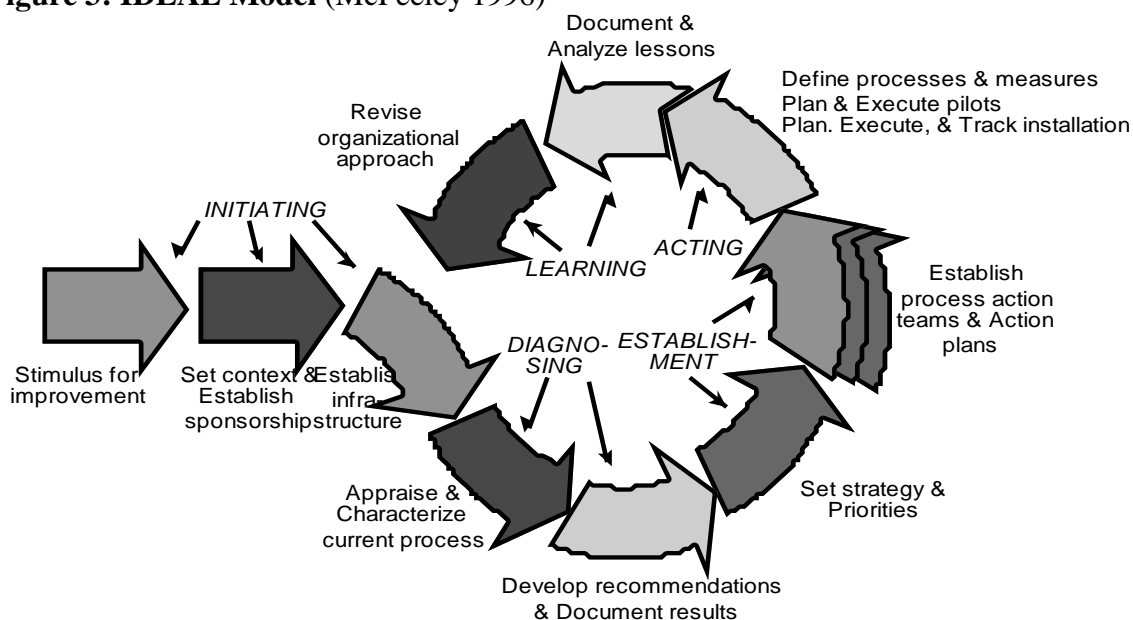
Using structural ambidexterity, systems development organization create a traditional subunit focused on exploitation and an agile subunit focusing on exploration (Vinekar, Slinkman et al. 2006). Each unit would differ with respect to management, desired skills, processes, and technology. In the traditional subunit, managers would use plan-driven approaches, developers would be tasked and rewarded as individuals, and conformance to standard processes and technology would be measured. In the agile subunit, managers would work as facilitators, developers would be tasked and rewarded within collaborative teams, and processes and technology would support incremental, evolutionary development. The perceived benefits of this separation include allowing the IS management team to learn and apply best practices from each subunit, allowing individuals within the organization to work in the culture that best matches

their personality, and providing a straightforward means of adding ambidexterity to an organization that is already proficient at either discipline or agility (Vinekar, Slinkman et al. 2006).

There are, however, limitations with the structural ambidexterity approach. First, it places the burden for ambidexterity solely on the top management team. By contrast, contextual ambidexterity encourages individuals within the organization to learn to become ambidextrous. Second, small firms may lack the resources or stability required for creating subunits dedicated to plan-driven and agile processes as advised by structural ambidexterity (Vinekar, Slinkman et al. 2006). Therefore, for small firms that operate in dynamic environments, the concept of contextual ambidexterity seems most feasible (Lubatkin, Simsek et al. 2006).

Process. The software literature provides very limited suggestions for managers that want to build ambidextrous software organizations. A short term solution for organizations that are lacking one set of skills is to obtain those skills through strategic partnering, whereas longer term solutions can be achieved by adopting sustained improvement efforts such as the People Capability Maturity Model (Curtis, Hefley et al. 2002) to improve staff capabilities (Boehm and Turner 2004). As the IDEAL model (McFeeley 1996) has been shown to be an effective means of making improvements in small software organizations (Kautz, Hansen et al. 2000), we adopted it as a framework for our research into making improvements at *TelSoft*. The IDEAL model (see Figure 3) was developed by the Software Engineering Institute to improve organizational maturity within software organizations. During the initiating phase, commitment is secured from the client to begin work on an improvement area. During the diagnosing phase, the researchers seek to understand the current problems and practices within the organization that may need changing. The establishing stage allows the researchers to plan action to be conducted in the acting phase. The learning stage is a time of critical reflection upon the lessons learned during earlier phases. This is also the time to decide whether to exit from the IDEAL cycle or whether an additional cycle will be required to meet project objectives.

Figure 3: IDEAL Model (McFeeley 1996)



Chapter 3: Research Approach

First, this chapter describes the selected research methodology: its definition, perceived benefits, inherent challenges, and evaluation criteria. Second, it describes the research process at *TelSoft* by discussing the research project's organizational structure as well as data collection and analysis techniques. For a more detailed description of data sources and improvement activities at *TelSoft*, see Part III of the dissertation. Chapter 5 applies the evaluation criteria to discuss the research cycle (McKay and Marshall 2001) and discusses the overall research contributions.

3.1 Research Methodology

This research is concerned with improving software practices. The term practice is used to describe meaningful action taken within a specific organizational or group context (Cook and Brown 1999). Software practices refer to software developers' and managers' everyday activities, routines, and processes directed toward increasing success for a portfolio of IS projects. Concerns at *TelSoft* included areas such as project portfolio management, project management, customer relationship management, software strategy, and software process management.

Like other action research based studies (Baskerville 1999), this research adopts an interpretive perspective. Interpretivists' ontological beliefs assume that reality is socially constructed by the actors within a particular situation. Interpretivists' epistemological beliefs require researchers to get actively involved in understanding the organizational context; therefore, a suitable research methodology must allow for observation and interaction in a field setting (Orlikowski and Baroudi 1991). In action research, the research team does not attempt an objective, value-neutral stance; instead, the researchers' beliefs and values play an active role in shaping and changing the organization.

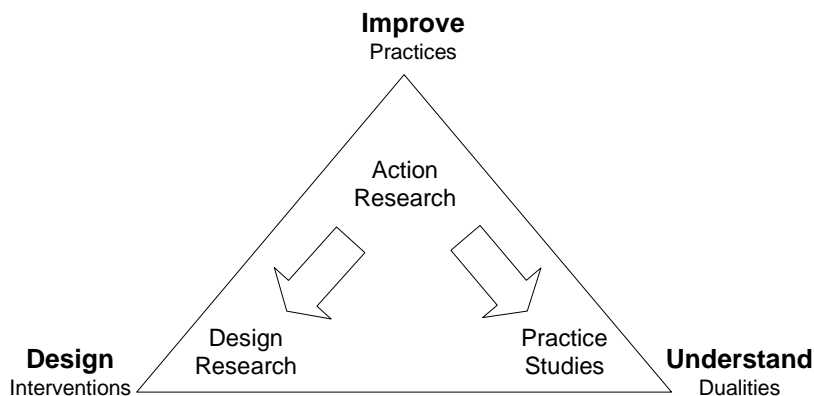
The overall research questions and objectives (see 1.2 Research Design above) as well as the researchers' ontological and epistemological stance should align and drive the research design (Orlikowski and Baroudi 1991; Mason 2002). Accordingly, we have selected CPR (Mathiassen 2002) as the most appropriate research methodology. CPR is a pluralist IS research methodology which generates meaningful contributions about software practices through close collaboration between researchers and practitioners. Methodological pluralism is appropriate for SPI because such highly complex real-world problems call for multiple perspectives to understand their richness (Mingers and Gill 1997; Mingers 2001). CPR aims to understand practice through interpretation, to support practice through designing artifacts, and to improve practice through making interventions. These research goals are accomplished by combining three different research approaches – practice studies, design research¹, and action research. In *practice studies*,

¹ In Mathiassen's article (2002), the term "experiment" was described as follows: Researchers "design normative propositions or artifacts, e.g. guidelines, standards, methods, techniques, or tools ... to create knowledge that can be used to plan, guide, or improve practice; the outcome is some form of artifact that has been developed and tested in relation to particular systems development disciplines" (Mathiasen 2002, p. 327). As this description is completely consistent with what is now commonly discussed in the IS literature as design science or design research (Hevner et al., 2004; Cole et al., 2005), we use the term "design research" here instead.

the goal is to understand practice through direct (e.g. case studies and observation) and indirect methods (e.g. interviews and surveys). In *design research*, the objective is to create innovative artifacts that solve wicked problems effectively and efficiently; these artifacts can be constructs which specify vocabulary and symbols, models that form new abstractions or representations, methods that codify algorithms or best practices that show feasibility of the idea (March and Smith, 1995; Hevner et al. 2004). In *action research* the objective is to “contribute both to the practical concerns of people in an immediate problematic situation and to the goals of social science by joint collaboration” (Rapoport 1970). Action research typically follows a learning cycle that consists of diagnosing, action planning, action taking, evaluating, and specifying learning (Susman and Evered 1978). CPR can lead to building and evaluating IS theories for analyzing, supporting, and improving software practices.

As illustrated in Figure 4, the goals of this research study were well aligned with CPR. Overall at *TelSoft*, we wanted to *understand* dualities involved in improving software practices (research objective 1), *design* appropriate interventions to address these dualities (research objective 2), and *improve* software practices by developing ambidextrous capabilities (research objective 3).

Figure 4: CPR-based Goals and Research Approaches (Mathiassen 2002)



In CPR, action research provides the overall structure for the research collaboration while practice studies and design research activities are incorporated as needed:

“Action research should be used as the basic form to establish a close relation to practice and to ensure the relevance of the research. But whenever feasible and useful this basic approach should be supplemented with experiments and practice studies.” (Mathiassen 2002, p. 339)

While CPR in this way combines different approaches, the overarching focus is on improvement and change and the dominating methodology is action research; practice research and design research elements are hence organized and presented as parts of overarching action research activities. Given this central role of action research in structuring this study, we next provide additional background about action research and show how this influenced the research process at *TelSoft*.

3.2 Action Research

“An action researcher is a person with a scientific attitude, an understanding of qualitative research principles, and understanding of the dynamics of change, and a commitment to studying problems that are relevant in real settings” (Cunningham 1993, p. 4)

The IS research community frequently debates the role of relevance in academic research. Proponents of basic research create knowledge for other academics and contend that the relevancy of their work to practitioners may only be appreciated in the future; supporters of applied research focus on solving the problems of today’s practitioners (Goldenson and Herbsleb 1995). As researchers strive to balance the dual goals of relevance and rigor, awareness has grown of action research as one possible solution. An appropriate balance can be achieved in a variety of ways. In fact, Baskerville and Wood-Harper (1998) describe as many as ten forms of action research including canonical action research (CAR) (Susman and Evered 1978), action science (Argyris 1985), and Multiview (Avison and Wood-Harper 1990). These action research forms differ according to their process model (iterative, reflective, or linear), structure (rigorous or fluid), typical involvement (collaborative, facilitative, or experimental), and primary goals (organizational development, system design, scientific knowledge, or training) (Baskerville and Wood-Harper 1998).

Checkland and Holwell (1998) conceptualize action research in terms of three key elements: an area of concern (A), a framework of ideas (F), and a methodology of inquiry (M). Explicating these elements at the beginning of the research project provides structure and focus, indicating which pieces of the many forms and variety of available data count as relevant data for your research. In this view of the research process, the researcher enters a real-world situation with an interest in a number of themes that apply within an area of concern (A). A specific methodology (M) is used to gain knowledge about the real-world problem and guide the intervention. The framework of ideas (F) is the theoretical perspective(s) explored within this context. The research process can yield insights in any of these three elements; for example, there can be lessons learned regarding the area of concern (A), suitability of the methodology (M), or extensions to theory (F).

McKay and Marshall (2001) expand on this idea by stating that action research contains two concurrent learning cycles, each having some version of A, F, and M:

1. Problem solving cycle that addresses the practical concerns of the industry partner (P: problematic situation; F: theoretical framing, and M_{PS} : methodology for addressing P).
2. Research cycle that addresses the need for scientific knowledge on the part of the researchers (A: area of concern; F: theoretical framing, and M_R : methodology for conducting researching into A).

The challenge for action researchers is to successfully navigate both inquiry cycles as well as the interdependencies between the two. Table 3 shows how these action research elements apply in the proposed dissertation work.

Table 3: Elements in the Action Research Intervention at TelSoft

| <i>Cycle</i> | <i>Element</i> | <i>Description</i> | <i>In this Study</i> |
|-------------------------------------|-----------------|--|---|
| <i>Problem Solving Cycle</i> | P | Problematic situation to be changed. Primary ownership lies with industry partner. | Improvement of software practice within <i>TelSoft's</i> Software Development group. |
| | F | Theoretical framing used to shape problem solving. | Adaptive organizations: A Sense-and-Respond Approach (Haeckel 1995; Haeckel 1999). SPI literature. Software engineering and RE literature. |
| | M _{PS} | Problem solving methodology. | The IDEAL methodology (McFeeley 1996). Interview, discussion, and workshops Process improvement teams. |
| <i>Research Cycle</i> | A | Area of Concern. | Improving Software Practices within the areas of <ul style="list-style-type: none"> ▪ RE assessment ▪ SPI action planning ▪ Software process management ▪ Project portfolio management. |
| | F | Theoretical Framing used to investigate A. Primary ownership lies with researchers. | Ambidextrous organizations (Birkinshaw and Gibson 2004; O'Reilly III and Tushman 2004). SPI literature. Software engineering and RE literature. |
| | M _R | Research Methodology. | Action research (Rapoport 1970; McKay and Marshall 2001; Davison, Martinsons et al. 2004). Collaborative practice research (Mathiassen 2002). |

There are important benefits to action research. Action research can lead to a rich data set based on a mixture of methods such as participant observation, interviews, document analysis, and surveys; the resulting data provide a strong foundation for supporting research that is high in external validity and relevance. Such characteristics make action research an excellent candidate for studying longitudinal organizational change processes (Pettigrew 1990). Baskerville and Wood-Harper (1996, p. 240) even state that “where a specific new methodology or an improvement to a methodology is being studied, the action research method may be the only relevant research method presently available.”

Key characteristics of the adopted action research design can be summarized in terms of the selected process model, structure type, involvement level, and primary goals (Baskerville and Wood-Harper 1998).

- The research process was *iterative* involving a repeating set of activities of diagnosis, action planning, action-taking, and learning. This supported *TelSoft* in applying learning from early experiences in the improvement effort.
- Within the meta-structure of the IDEAL model, the guidance was *fluid* with loosely defined activities. We allowed particular activities and specific improvement initiatives to emerge as the research process unfolded. This allowed more input from the practitioners involved and fitted the dynamic environment in which the industry partner operates.
- The research team's involvement was *facilitative*: the expertise of the research team guided the effort; however, practitioners took primary responsibility for resolving the encountered problematic situations.
- The primary goals of the research were *organizational development* (from the practitioners' standpoint) and *scientific knowledge* (from the research team's standpoint).

3.3 Research Criteria

To combat existing skepticism surrounding the validity of action research, it is important to exhibit rigor during data collection and analysis activities. However, managing the data collection process to adequately reflect on both the practical and research interests can be a challenge. Here, action researchers can learn from general recommendations for qualitative research such as techniques for documenting field notes (Miles and Huberman 1994), facilitating data analysis through computer software (Weitzman and Miles 1995), and demonstrating traceability between data and results (Lincoln and Guba 1985; Mason 2002). Overall, Checkland (1998) stresses the importance of the “recoverability” of action research projects. Recoverable research makes clear to “interested observers ... [the] processes and models which enabled the team to make their interpretations and draw their conclusions” (Checkland and Holwell 1998, p. 18).

To supplement this general advice, criteria for evaluating specific forms of action research have appeared in the literature. For example, Mårtensson and Lee (2004) propose three evaluative criteria for the usefulness of dialogical action research: (1) industry partner expresses that the problematic situation has been solved, (2) industry partner's expertise or knowledge has improved, and (3) the researcher's expertise or knowledge has improved. Davison et al. (2004) suggest five principles for guiding and evaluating canonical action research: creating a researcher-client agreement, using a cyclical process model, applying and extending theory, implementing an intervention, and reflecting upon the action. These five principles have been used in published canonical action research studies to provide evidence of validity (e.g. Lindgren, Henfridsson et al. 2004).

This research adopts six criteria for guiding the CPR-based research process. These criteria relate to roles, documentation, control, usefulness, theory, and transfer (Iversen, Mathiassen et al. 2004). Each criterion suggests questions that should be considered and addressed in planning the research and evaluating its validity (see Table 4).

Table 4: CPR Evaluation Criteria (Iversen et al., 2004)

| <i>Criteria</i> | <i>Questions</i> |
|-----------------|--|
| Roles | What are the researcher and practitioner roles? How do these roles develop over time? |

| <i>Criteria</i> | <i>Questions</i> |
|-----------------|--|
| Documentation | What data are collected to support the problem solving and research goals? How are these data collected? How is data quality ensured? |
| Control | How is the researcher-client relationship established? Who exercises authority over the process? To what degree are formalized control mechanisms adopted? |
| Usefulness | How is usefulness of the solution established in the problem situation? |
| Theory | How are frameworks used to support the study? How are results subsequently related to these frameworks? |
| Transfer | Under what conditions can the results be transferred to or adapted in other contexts? |

3.4 Research Partner

Case selection and description are important parts of qualitative research, and they are especially important in CPR. When compared against the sampling tradition of surveys, the use of a single case can seem particularly suspect. However, a single case can be especially valuable to study phenomena that are extreme, rare, or previously inaccessible; when it represents a typical instance; or when it allows the opportunity for a longitudinal study (Yin 2003). We find the use of a single case organization to be justified given the nature of action research, the fact that *TelSoft* is representative of other small software firms, and the opportunity to study the organization longitudinally.

When evaluating an industry partner, action researchers must consider potential ethical dilemmas, i.e. conflicts between the values and interests of researchers and industry partners (Rapoport 1970). First, the action researchers and industry partners must find one another acceptable. While Rapoport speaks about this from the standpoint of social responsibility, this also extends to the concern that the problems at the industry site are sufficiently interesting from a research perspective, that the subjects understand the real opportunities for improvement, and that a relevant theoretical framing exists (Kock 1997). My existing knowledge of *TelSoft* and its employees allowed us to feel confident that this was a suitable location. Second, issues of participant confidentiality and privacy was addressed by following the standards outlined by the Georgia State University Institutional Review Board (IRB) process (e.g. obtaining informed consent from employees, ensuring locked files and using pseudonyms). Third, researchers and industry partners might disagree over whether knowledge learned through the partnership may be shared with the research community. To prevent ethical dilemmas from arising later on in the project, we followed the principle of creating a researcher-client agreement (RCA) (Davison, Martinsons et al. 2004). Our RCA (called a “Memorandum of Understanding”) states the dual objectives of research and practice (see Part III, Appendix B.1). In addition, we agreed to use pseudonyms for the company and its employees in research writings.

The characteristics of the case organization help establish external validity, the domain to which findings can be generalized (Yin 2003). Accordingly, we next provide more details about

TelSoft's history and characteristics. Like other small software firms (Horvat, Rozman et al. 2000), *TelSoft* is oriented toward known customers in a niche market; it has high reliance on committed employees who perform many roles within the organization; and it has few resources devoted to innovation. Struggling to survive in a competitive environment, *TelSoft* frequently neglected innovation and adaptation, and instead emphasized known customers, products, and services. Although not considered a market leader, *TelSoft* has a reliable customer base consisting of two large customers that drive innovation to their core software products and several hundred smaller customers that use *TelSoft's* standardized geographic mapping software. Existing customers are also a major impetus for process improvement at *TelSoft*. In July 2000, *TelSoft* was prompted into process innovation by a major client's requirement for outside certification of its software capability by achieving level 2 on the SW-CMM (Paulk, Curtis et al. 1993; Paulk, Weber et al. 1995). However, after only one year of engaging in SPI, all resources associated with this initiative were abruptly reassigned when the client removed the certification requirement. Subsequently, no organized activity focused on improving management of individual projects or the project portfolio.

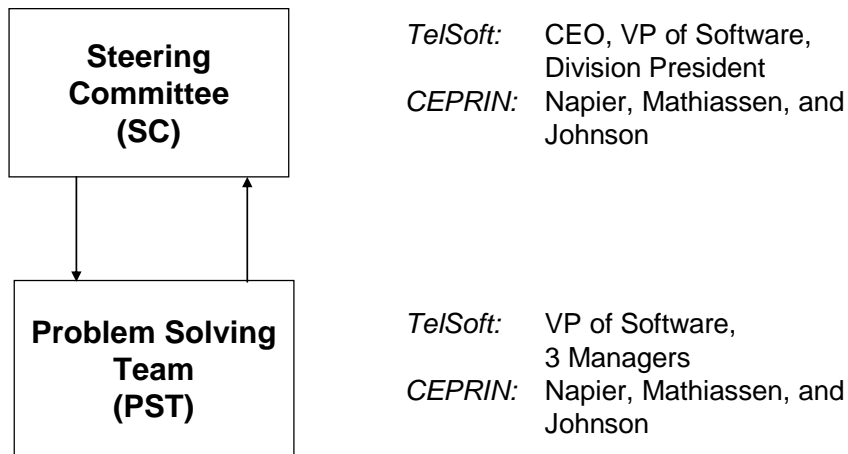
Prior attempts at technology-based innovation had gone poorly for *TelSoft*. In the late 1990s, *TelSoft* sensed that the introduction of spatial databases could revolutionize their GIS products. After years of investment, however, the company's CEO chose to terminate the project due to missed deadlines, inadequate functionality, and limited market success. From that point on, management was wary of developing new practices and pursuing new markets and was ordered by the CEO to halt all "speculative development" until further notice.

TelSoft management acknowledges that the company's biggest strength is its people: experienced software engineers with deep knowledge of its products, systems analysts with strong customer relationships, and managers willing to adapt quickly to customer requests. At the time our study began in 2004, *TelSoft* was forced to downsize its workforce, causing it to lose valuable customer and technical expertise, and also requiring that employees adopt additional roles and responsibilities. In addition, *TelSoft* was experiencing severe issues with their main customers: software releases were frequently shipped late, ran over budget, and contained deviations from agreed upon requirements. These issues prompted the management team to again invest in organizational innovation through the action research collaboration with Georgia State University.

3.5 Research Process

At the beginning of the initiative, the research team consisted of Nannette Napier, Dr. Lars Mathiassen, and Dr. Roy D. Johnson. The collaboration was managed by a steering committee (SC) of senior management from *TelSoft* and the research team (see Figure 5). The SC met 2-3 times per year as needed to oversee the project. More hands-on activities were completed by the problem solving team (PST) consisting of middle-level managers at *TelSoft* and the research team. Over the course of the initiative, the personnel and organizational structure of the collaboration evolved. For instance, the Division President was replaced in January 2005; temporary improvement teams were created that reported to the PST beginning in October 2005; a software coordination group (SCG) assumed the responsibilities of the SC in November 2005; and Dr. Johnson left the research team in April 2006. Part III provides more detail on these changes.

Figure 5: Managing Collaborative Practice Research (December 2004)



Data collection and documentation are essential for successful action research and qualitative research in general (Miles and Huberman 1994; Avison, Lau et al. 1999; Mason 2002). The study used multiple sources of evidence to corroborate findings (Miles and Huberman 1994; Mason 2002). These sources include: field observation, field notes, minutes from PST meetings, discussion and feedback from employee workshops on the improvement activities, diagnostic reports of software practices at *TelSoft*, and unlimited access to all *TelSoft*'s process documentation. During the diagnosing phase, the primary data sources were semi-structured interviews with 22 representatives from three major stakeholder groups: software development, internal customers, and external customers, as well as feedback workshops with employees. During the establishing and acting phases, we followed the progress of dedicated improvement teams by participating in team meetings, taking field notes, reviewing meeting minutes, and speaking informally with team participants. During the learning phase, an assessment was conducted to evaluate the initiative's impact, organizational structure, and overall perception by various stakeholders. Semi-structured interviews were again used and supplemented by an online survey sent to the broader software development group. Table 5 summarizes the data collection activities across the five phases of the research study and indicates the documents which are available in Appendix B of Part III.

Table 5: Data Collection at TelSoft

| | <i>Initiating</i> (8/13/2004 – 11/28/2004) | <i>Diagnosing</i> (11/29/2004 – 5/31/2005) | <i>Intervention Cycle 1</i> (6/1/2005 – 4/17/2006) | <i>Intervention Cycle 2</i> (4/18/2006 – 11/7/2006) | <i>Learning</i> (11/8/2006 – 3/31/2007) |
|----------------|--|---|---|--|--|
| Start of phase | First email sent to software development manager regarding possible collaboration | First diagnosing interview of software development manager | First PST meeting after all the diagnosing interviews were completed | Second Wave Kick-off Meeting | Second Wave Completion Meeting |
| Meetings | <p>Invitation to Collaboration with <i>TelSoft</i> management (10/12/04)</p> <p>First Problem Solving Team (PST) meeting (11/19/2004)</p> <p>Bi-weekly meetings of the Research Team</p> | <p>Number of management meetings:</p> <ul style="list-style-type: none"> • PST (5) • Steering Committee (SC) (3/16/2005) <p>Bi-weekly meetings of Research Team</p> | <p>Number of management meetings:</p> <ul style="list-style-type: none"> • PST (10) • SC (6/9/2005) • Software Coordination Group (SCG) (8) <p>Number of improvement team meetings:</p> <ul style="list-style-type: none"> • Combined Configuration Management-Quality Assurance (1) • Configuration Management (9) • Customer Relations (7) • Quality Assurance (10) • Requirements Management (6) | <p>Number of management meetings:</p> <ul style="list-style-type: none"> • PST (8) • SCG (6) <p>Number of improvement team meetings:</p> <ul style="list-style-type: none"> • Customer Relations (5) • Process Management (7) • Quality Results (6) | <p>Number of management meetings:</p> <ul style="list-style-type: none"> • PST (3) • SCG (4) |

| | Initiating <i>(8/13/2004 – 11/28/2004)</i> | Diagnosing <i>(11/29/2004 – 5/31/2005)</i> | Intervention Cycle 1 <i>(6/1/2005 – 4/17/2006)</i> | Intervention Cycle 2 <i>(4/18/2006 – 11/7/2006)</i> | Learning <i>(11/8/2006 – 3/31/2007)</i> |
|------------------------------------|---|---|---|--|---|
| Meeting Documentation | Private meeting notes <ul style="list-style-type: none"> • Invitation to collaboration • Researcher meetings (5) • PST meeting (1) Public meeting minute <ul style="list-style-type: none"> • PST meeting | Private meeting notes <ul style="list-style-type: none"> • Field notes reflecting upon interactions at <i>TelSoft</i> (11 days) • Notes from 22 interviews • Notes from research meetings (6) Public meeting minutes <ul style="list-style-type: none"> • PST meetings (3) Transcription <ul style="list-style-type: none"> • 5 interviews | Public meeting minutes <ul style="list-style-type: none"> • Configuration Management (5) • Customer Relations (4) • PST meetings (7) • Quality Assurance (5) • Requirements Management (2) | Public meeting minutes <ul style="list-style-type: none"> • Customer Relations (2) • Process Management (7) • PST meetings (3) • Quality Results (4) | Public meeting minutes <ul style="list-style-type: none"> • PST Action Items List (3) |
| Other Data Collection methods | None | 22 Assessment Interviews (11/29/2004 – 5/25/2005) Requirements engineering standardized assessment (3/30/2005) | None | None | <ul style="list-style-type: none"> • 10 Assessment Interviews (12/19/2006 – 2/25/2007) • Online survey sent to 25 <i>TelSoft</i> employees regarding SPI impact • Requirements engineering standardized assessment (6/19/2007) |
| Workshops or Group Status Meetings | None | Workshops to present and verify interview data <ul style="list-style-type: none"> • Software Development (1/19/05) • Internal customers Workshop (3/16/05) | First Wave Kick-off meeting (9/1/2005) Interim Status Presentation <ul style="list-style-type: none"> • Software Manager's meeting (3/15/2006) • Software Development staff (3/21/2006) | Kick-off Meeting for Second Wave (4/18/2006) | Second Wave Completion meeting (11/8/2006) |

| | Initiating <i>(8/13/2004 – 11/28/2004)</i> | Diagnosing <i>(11/29/2004 – 5/31/2005)</i> | Intervention Cycle 1 <i>(6/1/2005 – 4/17/2006)</i> | Intervention Cycle 2 <i>(4/18/2006 – 11/7/2006)</i> | Learning <i>(11/8/2006 – 3/31/ 2007)</i> |
|---------------------------|--|---|--|--|---|
| Key Project Documentation | <p>Invitation to Collaboration slides</p> <p>Memorandum of Understanding^</p> <p>Project Focus Document</p> <p>TelSoft Organization Chart</p> <p>IRB Protocol #H05176^</p> <p>“Managing Requirements in Providing and Innovating Software Services”</p> <p>TelSoft process documentation (53 files consisting of templates, process flows, guidelines, and examples)</p> | <p>Requirements Process Summary based upon interviews</p> <p>Interview Guide for Software Development Internal Customers, and External Customers^</p> <p>Phase 1 Diagnostic Report summarizing the interviews and standards assessment^</p> | <p>Software Charter^</p> <ul style="list-style-type: none"> • Reason For Being • Software Strategy • Policies <p>SCG Fixed Agenda^</p> <p>Outputs from each improvement team:</p> <ul style="list-style-type: none"> • Project Plan • Position papers • Process documents • Templates • Transition plan <p>First Wave Summary Report</p> | <p>Updated TelSoft’s website to include Software Charter and select process documents</p> <p>PST Fixed Agenda^</p> <p>Outputs from each improvement team:</p> <ul style="list-style-type: none"> • Project Plan • Position papers • Process documents • Templates <p>Second Wave Summary Report^</p> | <p>Final Project Assessment Reports:</p> <ul style="list-style-type: none"> • External customer interview summaries • SPI Impact results report^ • SCG assessment report • Requirements Engineering Assessment results^ |

As suggested by Miles and Huberman (1994, p. 56), data analysis was an ongoing process. This iterative nature of action research, in particular, assured that data collection and data analysis were intertwined. Thus, data analysis proceeded across project phases and informed activity in subsequent phases. For example, the research team met during the diagnosing phase to detect patterns emerging from the interview data and to reflect upon what was learned. We created interim reports and held status meetings with members of the software development group. For each research paper, an additional level of analysis was conducted was driven by specific research objectives and focused on a subset of the data collected. These detailed analyses are described in the research papers presented in Part II.

Chapter 4: Review of Results

In this chapter, we summarize the results and contribution each of the four papers within this dissertation. Chapter 5 elaborates further on the overall contribution and research implications while also reflecting on limitations of the study.

4.1 Paper 1: Combining Perceptions and Processes

The first paper is based upon our experiences in the diagnosing phase and details our search for an appropriate methodology for effectively assessing RE practice. When evaluating RE practice at *TelSoft*, we identified the duality of process-driven versus perception-driven assessment and developed a framework for combining both approaches.

Table 6: Paper 1 Summary

| | |
|------------------------|--|
| Area of concern (A) | RE assessment |
| Framework of ideas (F) | <ul style="list-style-type: none"> ▪ Process-based: Total quality management, process management ▪ Perception-based: Stakeholder analysis |
| Methodology (M) | <ul style="list-style-type: none"> ▪ Process-based: Requirements Engineering: Good Practice Guide (REGPG) assessment ▪ Perception-based: Semi-structured interviews and workshops |
| Research Questions | <ol style="list-style-type: none"> 1. What different insights are gained from process- and perception-driven assessments of RE practices? 2. How can processes and perceptions be combined in assessment of RE practices? |
| IDEAL Research Phases | Diagnosing, Learning |
| Contributions | <ul style="list-style-type: none"> ▪ Demonstrates importance of combining process-based and perception-based knowledge when evaluating RE practices ▪ Describes a combined RE assessment framework with steps and guidelines for conducting process-based and perception-based inquiry |

Researchers have used three main approaches to RE assessment: analyzing the RE-related data from generic software process assessments (e.g., SW-CMM or ISO/IEC 15504) (2000); applying a RE-specific version of the SW-CMM (Beecham, Hall et al. 2005); and, measuring adherence to best practices based on a dedicated RE maturity model such as the Requirements Engineering Good Practice Guide (REGPG) (Sommerville and Sawyer 1997; Sommerville and Ransom 2005). Although all three approaches acknowledge the importance of tailoring assessments to organizational needs, they each assume that RE is best assessed and improved by benchmarking against best practices (Nielsen and Pries-Heje 2002). This thinking is consistent with the ideas behind total quality management and process management (Deming 1986; Zbaracki 1998). Unfortunately, these process-driven approaches do not necessarily engage stakeholders in ways that increase buy-in and facilitate successful implementation of new practices.

An alternative approach to RE assessment would privilege perceived problems over prescribed processes (Nielsen et al., 2002) as suggested in Table 7. In the perception-based approach, stakeholder perceptions about strengths, weaknesses, and opportunities related to RE activities and artifacts drive data collection and analysis; stakeholders, rather than models, determine what is important to study by assigning priorities to problems; and, solutions are grounded in the specific context of the problematic situation. Perception-based assessment considers organizational stakeholders' perceptions of current and future practices as important sources for innovation and learning. The perception-based approach borrows from general stakeholder analysis (Lyytinen 1988; Pouloudi and Whitley 1997; Vidgen 1997). Like interpretive research, stakeholder analysis considers organizational actors' subjective meanings as important knowledge sources; therefore, researchers emphasize the specific terms and perceptions of each stakeholder and avoid presenting a priori concepts (Orlikowski and Baroudi 1991).

Table 7: Competing Assessment Approaches: Process-based and Perception-based

| | <i>Process-based</i> | <i>Perception-based</i> |
|---|---|---|
| What counts as data? | Prescribed processes; Deviations between current and best practices | Perceived problems; Stakeholder perceptions of problems |
| What determines focus of assessment? | A priori model of RE | Stakeholders |
| What is the source for solutions? | Tailored from ideal model of best practice | Grounded in context of the problematic situation |

This paper offers two primary contributions. First, it expands our knowledge of what constitutes legitimate, meaningful data when evaluating RE practices. This is done by explicitly characterizing the existing approaches as being process-based and by offering the complementary approach of perception-based RE assessment. In addition, the results from a process-based assessment (REGPG) and perception-based assessment are compared. The REGPG assessment identified *TelSoft's* strengths as being in the areas of documenting, eliciting, and describing requirements; areas for improvement were in analyzing, validating, and managing requirements. The company's overall RE maturity level was assessed at the lowest level: initial. The perception-based assessment identified some findings that complemented this assessment and other insights that were contradictory. At the same time, we found instances where one form of inquiry provided insight into an area that the other did not even address. These examples illustrate the benefit of combining the two sources of knowledge to obtain a more comprehensive view of RE practices.

Second, it creates an RE assessment framework which takes advantage of both kinds of knowledge. Using Gregor's (2006), classification for IS theories, this framework can be classified as a theory for design and action which gives specific prescriptions for assessing RE practices. This combined approach to RE assessment prescribes three steps: initiating the assessment, executing multiple inquiry cycles, and making recommendations based upon the findings. The paper also suggests activities that should be considered during each step and illustrates how this was done at *TelSoft*. We found this framework to be an effective tool in planning both the diagnosing and learning phases of the research collaboration at *TelSoft*.

4.2 Paper 2: Negotiating Repeat-ability and Response-ability

A manager trying to decide how to improve RE practices may choose from one of two competing theories about why current software practices are problematic and how problems are resolved: repeat-ability and response-ability (Napier, Mathiassen et al. 2006). Drawing upon the literature on software process improvement and the literature on agile software development, we suggest that these theories differ based upon their assumptions about: nature of requirements, requirements capture, requirements usage, change management, and improvement approach (as summarized in Table 9).

Table 8: Paper 2 Summary

| | |
|------------------------|---|
| Area of Concern (A) | SPI action planning |
| Framework of ideas (F) | <ul style="list-style-type: none"> ▪ Repeat-ability: Plan-driven development ▪ Response-ability: Agile development |
| Methodology (M) | Alternative templates strategy |
| Research Questions | <ol style="list-style-type: none"> 1. What assumptions distinguish repeat-ability from response-ability theories of RE? 2. How do repeat-ability and response-ability theories differ in assessing RE practice? 3. How do repeat-ability and response-ability theories apply to improving RE practice? |
| IDEAL Research Phase | Establishing |
| Contributions | <ul style="list-style-type: none"> ▪ Explicates two theories for understanding and resolving issues in RE practice: repeat-ability and response-ability ▪ Demonstrates how RE practices can be improved by considering both perspectives |

Repeat-ability holds that good requirements practices are plan-driven and follow a set of generic best practices for how to arrive at an agreed-upon baseline of software requirements. Repeat-ability is an important principle within the SW-CMM (Paulk, Curtis et al. 1993). In fact, the first step in increasing organizational maturity involves moving from an initial level to a repeatable level by reducing variations in practices (Humphrey 1989). From the repeat-ability perspective, requirements are textual representations of the desired software capabilities. Requirements knowledge is explicated as objects that are passed between requirements providers and requirements receivers. Requirements capture is a formal process that occurs before development work begins; it includes document review, discussion, and sign-off to indicate approval. Once sign-off has been obtained, a requirements baseline is established. Any changes to the requirements baseline must be documented and communicated to relevant stakeholders (Paulk, Curtis et al. 1993). The role of quality assurance is to verify that the completed software matches the requirements specification. If RE practices are problematic, this approach looks for missing or inefficient processes. The overall improvement approach in the repeat-ability paradigm is to institute best practices and reduce process variance (Humphrey 1989).

In contrast, response-ability holds that good requirements practices are adaptive and involve close collaboration and interaction between customers and developers to help develop

satisfactory software solutions. Response-ability is an important principle within agile development approaches (Beck 1999; Boehm and Turner 2004; Turk, France et al. 2005). In fact, one of the four basic principles of the Agile Manifesto is “Responding to change over following a plan” (Agile Alliance 2001). In the response-ability theory, requirements exist as shared understandings between stakeholders. Requirements knowledge is tacit, and the role of documentation is minimized. Customers play a critical role during software development as expressed in the principle “Customer collaboration over contract negotiation” (Agile Alliance 2001). Customers provide immediate feedback on interim versions of the software and set priorities for the next iteration. Requirements capture happens informally as part of ongoing conversations with customers. This incremental approach allows requirements changes to be incorporated into the next version of the software. If RE practices are problematic, this approach looks for breakdowns in communication with customers or between developers. The overall improvement approach is to increase customer satisfaction by enhancing collaboration to quickly adapt to customer requests.

Table 9: Competing Improvement Approaches: Repeat-ability versus Response-ability

| | <i>Repeat-ability</i> | <i>Response-ability</i> |
|------------------------|--|--|
| Nature of requirements | <ul style="list-style-type: none"> • Requirements represent software capabilities • Requirements are explicated as texts in documents | <ul style="list-style-type: none"> • Requirements are perceptions of software capabilities • Requirements are tacitly embedded in social relationships |
| Requirements capture | <ul style="list-style-type: none"> • Requirements are derived through specification • Interaction is formal | <ul style="list-style-type: none"> • Requirements are discovered through negotiation • Interaction is informal |
| Requirements usage | <ul style="list-style-type: none"> • Requirements are baselined and predate development • Requirements are stored with traceability to source code | <ul style="list-style-type: none"> • Requirements emerge through development • Requirements are expressed through software solutions |
| Change management | <ul style="list-style-type: none"> • Requirements changes are exceptions and must be managed | <ul style="list-style-type: none"> • Requirements changes are expected and must be embraced |
| Improvement approach | <ul style="list-style-type: none"> • The goal is to reduce process variance through best practices | <ul style="list-style-type: none"> • The goal is to increase customer satisfaction through collaboration |

This description of repeat-ability and response-ability represents the primary contribution of this paper. The two theories led to quite different inventories of problems and, as a consequence, also to quite different recommendations for improvement at *TelSoft*. In fact, there is little overlap between the two sets of findings. At the same time, both inventories of problems made sense to managers at *TelSoft*, and they were found to represent relevant and important issues related to RE practices. This application of the two theories suggests that they represent different and relevant perspectives on RE practices. In the end, *TelSoft* managers selected an improvement strategy that consisted of solutions from each category.

4.3 Paper 3: Managing Legacy and Current Processes

The third paper addresses the need for improving software process management at *TelSoft*. Our chosen approach to managing RE processes at *TelSoft* valued both exploiting legacy and exploring new processes.

Table 10: Paper 3 Summary

| | |
|------------------------|---|
| Area of Concern (A) | Software process management |
| Framework of ideas (F) | <ul style="list-style-type: none"> ▪ Business process change ▪ Legacy systems reengineering |
| Methodology (M) | Design and refine SPR principles and model |
| Research Objectives | <ol style="list-style-type: none"> 1. To define and identify principles for software process reengineering (SPR) 2. To propose and evaluate a model for SPR |
| IDEAL Research Phase | Acting |
| Contributions | <ul style="list-style-type: none"> ▪ Articulates the need for SPR ▪ Develops SPR principles and model ▪ Evaluates SPR model at <i>TelSoft</i> |

Once problems have been diagnosed and recommendations have been identified, the improvement approach under the repeat-ability paradigm recommends reducing variance by instituting best practices. These best practices become part of the organization's library of *software processes*: "the coherent set of policies, organizational structures, technologies, procedures, and artifacts that are needed to conceive, develop, deploy, and maintain a software product" (Fuggetta 2001, p. 560). In this paper, we further distinguish between legacy processes and managed processes. Legacy processes are software process descriptions that have not been carefully managed over time and consequently have become inconsistent with the organization's current policies and practices. By contrast, managed processes are software process descriptions that have a well-defined state, represent current organizational policies, and are explicitly monitored and controlled. Managed processes are in line to be approved and implemented into engineering practices.

To ensure that software processes are defined, documented, measured and controlled (Humphrey 1989; Krasner, Terrel et al. 1992), organizations need to practice software process management. Ideally, an organization would have a software process repository that contains only managed processes and no legacy processes. However, over time, organizations that have inadequate software process management discipline stand to continue generating legacy processes. This presents a challenge for the practicing SPI manager: Given the starting point of legacy processes within the organization, what is the best way to integrate these into a process repository with managed processes and at the same time establish software process management within the organization?

Two competing approaches here emphasize either exploitation or exploration (see Table 11 for summary). The exploitation approach focuses on reusing knowledge contained within legacy processes. Accordingly, legacy processes are evaluated for fit with current policies and practices. Legacy processes that are well-aligned are revised and become managed processes while legacy

processes that are misaligned are discarded. The exploitation approach is appropriate when the organization attaches value to the knowledge embedded within the legacy processes despite the need for cleanup. Following such a process would allow the organization to leverage existing software processes, and it would reinforce the beneficial contributions of prior improvement efforts.

The exploration approach starts with a clean slate and focuses on creating new knowledge. All legacy processes are ignored, and the managed processes are designed from scratch based upon current business requirements. This approach saves the time associated with filtering and revising existing documents; however, the organization bears the extra burden of inventing and designing new processes. Furthermore, such an approach does not allow the organization to leverage the investments made in existing process capabilities, it requires that all processes are designed from scratch, and it easily reinforces general mistrust in the value of SPI. Nevertheless, if there is a great distance between the legacy processes and current business needs, starting from scratch may seem more appropriate.

Table 11: Competing Process Approaches: Exploitation versus Exploration

| | <i>Exploiting legacy processes</i> | <i>Exploring new processes</i> |
|----------------------|--|--|
| Rationale | Aligning old processes with current policies and practices | Developing new processes in response to identified needs |
| Starting point | Legacy processes | Clean slate |
| Core activities | Filtering and revision | Invention and design |
| Knowledge management | Reuse existing knowledge | Create new knowledge |

This paper makes three key contributions related to dealing with exploiting and exploring legacy processes. First, we identify an important problem within the software process management community: our literature search revealed no mention of the problem of revival and renewal (i.e. trying to learn from previous efforts after a failed SPI initiative). A key point is that organizations' history with SPI impacts their ability to move forward. This is especially true for those that follow SPI approaches with a heavy focus on generic, documented processes that are tailored to individual projects. When these software organizations fail to institute proper process management practices or when they decide to reinvest in SPI, they may likely be confronted with a considerable portfolio of legacy processes. Future research needs to further appreciate this problem and reconsider how software organizations can effectively develop and implement process management solutions.

Second, we provide a general solution to this problem which we defined as software process reengineering (SPR):

“SPR defines criteria for transforming legacy processes; assesses existing software processes against these criteria; and selects which processes should be removed, innovated, or implemented. SPR establishes on that basis a repository of managed software processes and institutes a process management discipline to support continued improvement efforts.”

Rather than serving as an ongoing activity, SPR is a process that allows for transitioning from a chaotic state with low process discipline to a managed state with improved software process management discipline. Drawing upon literature on business process change and legacy systems reengineering, we identify principles and steps for conducting SPR. The heart of the SPR activity involves making commitments that are agreed upon by the assessors as to the difference between the current and desired state of process documents and repository and then putting a plan in place for making improvements. The guidelines provide a series of steps to consider when taking action.

Third, we demonstrate how the SPR model was used at *TelSoft* and evaluate its effectiveness. As other software organizations engage in SPR, their situation will be different from the one at *TelSoft*. Therefore, managers must carefully consider how to adapt the proposed SPR model to meet the organization's specific needs. Future research is needed to investigate the suitability of the model within other software organizations as well as to analyze its long-term effectiveness.

4.4 Paper 4: Becoming Ambidextrous

Drawing upon Pettigrew's guidance for contextualist inquiry (Pettigrew 1985; Pettigrew 1987), we show how performance management and social support context changed over time at *TelSoft*, resulting in improvements in alignment and adaptability. Based on these experiences, we propose a model for becoming ambidextrous through the processes of diagnosing, visioning, intervening, and practicing.

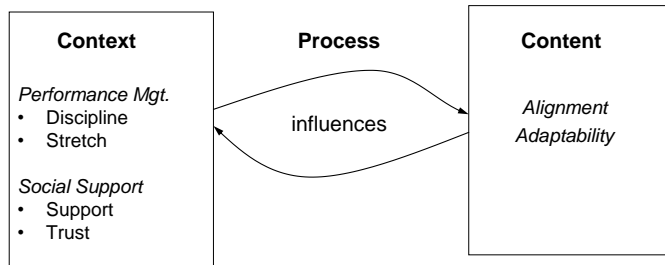
| | |
|------------------------|---|
| Area of Concern (A) | Project portfolio management |
| Framework of ideas (F) | Contextual ambidexterity |
| Methodology (M) | Contextualist Inquiry |
| Research Objectives | To explore how organizations can develop managerial practices and organizational contexts as they strive to become ambidextrous |
| IDEAL Research Phases | Diagnosing, Establishing, Acting, Learning |
| Contributions | Identified four phase process for becoming ambidextrous using contextualist inquiry perspective |

In this paper, our focus is on *TelSoft*'s attempt to improve project portfolio management, i.e. the systematic management of the company's projects in order to decide which projects should be added or removed as well as the relative priority of projects within that portfolio (Markowitz 1952; McFarlan 1981; De Reyck, Grushka-Cockayne et al. 2005). In software firms that are project-based organizations, project portfolio management is a core management activity requiring ongoing assessment of existing projects and new business opportunities (Clark and Wheelwright 1992; Hobday 2000). The primary mechanism that *TelSoft* used to increase project portfolio management was through the creation of the SCG in November 2005. The SCG consisted of four members: Division President, Vice President of Software, Development Manager, and Product Manager. At its monthly meetings, the group followed a fixed agenda covering status of current projects, business opportunities, improvement initiative, and strategy review. With the inclusion of the improvement initiative on its agenda, the SCG assumed the role of the steering committee. To raise awareness of customer relations issues, the SCG periodically

invited account managers to provide status on the customer relationship and identify areas of improvement.

We framed our inquiry into becoming ambidextrous as a contextualist study employing the methodology of action research. Contextualist inquiry is concerned with understanding how transformation efforts unfold in particular organizational settings focusing on the interactions between content, context, and process (see Figure 6). *Content* refers to the areas being transformed; in this case we focus on project portfolio management practices at *TelSoft*. *Context* refers to the outer environment in which the organization operates as well as the inner environment representing systems, processes, and beliefs within the organization. Following the definition of contextual ambidexterity, we are particularly interested in performance management and social support elements of the inner context. Finally, *process* refers to the actions and interactions between various interested parties as they attempt to transform practices. In our case, we focus on the actions and interactions related to building alignment and adaptability within *TelSoft*.

Figure 6: Contextualist Inquiry into Becoming Ambidextrous



The main contribution of this paper is a model for becoming ambidextrous consisting of four phases: diagnosing, visioning, intervening, and practicing. The model incorporates contextualist inquiry's two-dimensional approach by focusing on the horizontal unfolding of the change process across the four phases of the action research and the interaction between content and context.

Chapter 5: Discussion

In this chapter, we discuss the overall contribution of the research, evaluate the work against criteria for CPR-based research, and discuss implications for research and practice.

5.1 Research Contribution

Each of the individual papers addressed one or more of the overall research objectives in a specific area of improving software practices, see Table 12. In this section, we reflect more broadly on findings from both the research cycle (i.e. research papers summarized in chapter 4 and documented in Part II) as well as the problem-solving cycle (i.e. activities at *TelSoft* documented in Part III). For each of the three objectives, we consider what we learned about the overall research question: How can an ambidextrous perspective facilitate improvement in software practices?

Table 12: Relationship between individual papers and research objectives

| | <i>Objective 1 Dualities</i> | <i>Objective 2 Design</i> | <i>Objective 3 Process</i> |
|---------|--|---|---|
| Paper 1 | Perception Process | RE combined assessment approach | — |
| Paper 2 | Repeat-ability Response-ability | Improvement teams driven by policies and focused improvement areas | — |
| Paper 3 | Exploiting legacy processes Exploring new processes | Software process reengineering model | — |
| Paper 4 | Alignment Adaptability | SCG focused on project portfolio management | Four-step process: diagnosing, visioning, intervening, practicing |

Dualities. The first research objective was to identify dualities involved in improving software practices. Emphasizing tensions, conflicts, dilemmas, and paradoxes has been shown to be a useful way of making sense of and redesigning organizational practices (Van de Ven and Poole 1995). To that end, this research has expanded our understanding of the dual capabilities involved in the domains of RE assessment (Napier, Mathiassen et al. 2006), SPI action planning (Napier, Mathiassen et al. 2006), and software process management (Napier, Kim et al. under review). We identified three specific benefits to identifying dualities as suggested through this ambidextrous perspective.

First, identifying dualities challenged us to look beyond the dominant paradigm and to expose alternative viewpoints. For instance, with respect to RE assessment, we found that evaluation techniques predominantly emphasized alignment with best practices over the perceptions of key

stakeholders. Therefore, we presented a combined RE assessment approach that highlighted the importance of considering perceptions as part of the total evaluation (Napier, Mathiassen et al. 2006). Similarly, with respect to software process management, we found that the literature did not explicitly address how organizations could leverage the existing knowledge found in legacy software processes when reviving SPI initiatives. To address this, our SPR model developed principles for reengineering software processes as organizations transition to more systematic software process management (Napier, Kim et al. under review).

Second, identifying dualities prompted us to independently consider each perspective, thereby increasing information available for improvement. For instance, with RE assessment, we found the knowledge learned by combining both types of inquiry led to a richer diagnosis at *TelSoft* (Napier, Mathiassen et al. 2006). With respect to SPI action planning, we demonstrated that adopting either the repeat-ability or response-ability lens limited the diversity of resulting recommendations (Napier, Mathiassen et al. 2006); instead, managers at *TelSoft* developed an action plan that combined elements of both recommendations.

Third, identifying these dualities provided insights beyond the current emphasis on discipline and agility (Boehm 2002; Boehm and Turner 2004; Lee, DeLone et al. 2006; Lee, DeLone et al. 2007). For instance considering contextual ambidexterity, performance management and adaptability covered the recognized need for discipline and agility; adding social support, alignment, and stretch acknowledged that organizational context and culture are also important concerns for software managers. In this way, contextual ambidexterity can broaden the software community's focus.

Design. The second research objective was to design interventions based on the identified dualities to improve software practices. As summarized in chapter 4, we created two papers that specifically addressed approaches for managing the dualities identified in RE assessment and software process management (Napier, Mathiassen et al. 2006; Napier, Kim et al. under review). Looking broadly at the goal of improving software practices, the *GSU-TelSoft* collaboration itself was, in fact, an intervention designed to both improve software practices and increase ambidexterity. Considering the *TelSoft* SPI effort through a contextual ambidexterity lens, the intervention consisted of two primary activities: (1) establishing an effective organizational context and (2) increasing the alignment and adaptability of specific improvement areas (e.g. project portfolio management, quality assurance, configuration management, process management) (Napier, Mathiassen et al. under review).

Contextual ambidexterity states that performance management and social support facilitate ambidexterity and, consequently, organizational performance. The performance management context represents systems, processes, and beliefs related to meeting performance objectives set by the organization's management (Gibson and Birkinshaw 2004). Discipline is an attribute that encourages people to voluntarily meet those objectives whereas stretch is an attribute that encourages people to strive for even more ambitious goals (Ghoshal and Bartlett 1994). The social support context represents systems, processes, and beliefs associated with member relationships (Gibson and Birkinshaw 2004). Trust is an attribute of the organizational context that encourages people to rely on one another whereas support is an attribute that empowers people to lend assistance to others (Ghoshal and Bartlett 1994).

One of the main mechanisms we used to improve the performance management and social support context at *TelSoft* was through establishing the PST, SCG, and improvement teams (for more detail see Part III). Performance management was increased through these teams by creating a shared ambition amongst team members, developing standards for the teams as well as the development group, and ensuring the teams were provided with feedback on their work; social support was improved through these teams by striving for broad employee participation and providing the teams with autonomy (Ghoshal and Bartlett 1994). For example, as described in (Napier, Mathiassen et al. under review), when the SCG was formed, a project plan was created that explained the group's mission and proposed membership, a meeting with participants was held to explain this mission, and a fixed agenda was created which specifically listed the activities for the meeting. The group committed to meeting monthly, scheduling a year's worth of meetings from the beginning. Leadership for a specific agenda item was associated with each participant's regular job roles; therefore, they already had a personal stake in the topic being discussed. The initial SCG meetings were spent creating standards for the information that would be needed to enable decision making about project portfolio management. By requesting specific information and discussing project status, the SCG held *TelSoft*'s project managers more accountable for project outcomes. The GSU researchers were actively involved with the SCG as well as the other teams to provide immediate feedback and guidance as needed.

The design for the SPI initiative at *TelSoft* considered building both alignment – the capacity of employees within the business unit to work toward a common goal, and adaptability – the capacity of the business unit to change quickly in response to dynamic market conditions (Gibson and Birkinshaw 2004). To provide opportunities for increasing alignment, the intervention plan specified seven improvement areas as the focus for SPI, established improvement teams to work on specific objectives in one or more areas, and held the improvement teams accountable by requiring periodic status reports and presentations to the software development group. In addition, the SCG used input from the improvement teams to create nine software policies which served as operating principles for software development (see Part III, Appendix B.5). To encourage adaptability, we recommended that *TelSoft* abandon strict command-and-control approaches and use governing principles and defined roles to become a more adaptive enterprise (Haeckel 1995). Seen from the standpoint of sensing capability and responding capability (Overby, Bharadwaj et al. 2006), *TelSoft* needed to combine the ability to sense customer needs and technological and market opportunities while dynamically responding once aware of suitable opportunities. The SCG and customer relations team led efforts to address adaptability. These activities included increased emphasis on defining product strategy, actively seeking business opportunities outside of the telecommunications market, and more frequent face-to-face customer interactions.

Process. The third research objective was to investigate the process of becoming an ambidextrous software organization. Although ambidexterity is increasingly acknowledged as an important organizational capability, managers receive limited actionable advice on how it can be developed. To provide insight into becoming ambidextrous, we focused on the process of improving project portfolio management at *TelSoft*. More specifically, in paper 4 we analyzed the development of the SCG and the interaction between organizational context and alignment-

adaptability (Napier, Mathiassen et al. under review). Drawing from Birkinshaw and Gibson's arguments concerning contextual ambidexterity (2004) and Pettigrew's contextualist inquiry (1985; 1987), we generated a process model showing how alignment and adaptability practices improved over four phases of managed change: diagnosing, visioning, intervening, and practicing.

This research model draws attention to the dynamics of change and the interactions between process, context, and the content of planned change. For instance, we found that *TelSoft* first dealt with contextual issues (social support and performance management) before realizing improvements to content (alignment and adaptability). In fact, the main emphasis during the visioning phase was not on improving ambidexterity per se, but rather on transforming the context to better facilitate ambidexterity. Over time, managers should anticipate such shifts between improvements in context and content. The analysis also showed that transformation of context is not a simple progression of improvements. Although performance management and social support at *TelSoft* both improved across the phases, setbacks were apparent, especially during the intervening phase when social support suffered.

Prior research into ambidextrous organizations has considered ambidexterity as a property at the organizational, business unit, and individual levels (Tushman and O'Reilly III 1996; Gibson and Birkinshaw 2004). Our research also finds that the process of becoming ambidextrous can be applied to specific managerial practices within the organization. *TelSoft* had a number of management practices which might have been the focus of an innovation effort. At *TelSoft*, we identified project portfolio management as a key managerial activity in which the firm's ability to align and adapt was challenged.

5.2 Research Evaluation

In this section, we use six criteria for CPR-based action research to demonstrate validity of the research results (Iversen, Mathiassen et al. 2004) as well as its limitations.

Roles. Establishing and keeping good relationships throughout all phases of the collaboration is critical for action research. At the beginning of the *TelSoft* initiative, the researcher and practitioner roles and responsibilities were clearly outlined in the memorandum of understanding. The research team played a facilitative role (Baskerville and Wood-Harper 1998): they were viewed as experts responsible for organizing the change process and doing the bulk of the action involved, such as conducting the diagnostic interviews. My own role as a former employee allowed me "insider" status with the software engineers at *TelSoft*, privileging me to candid conversations about *TelSoft* management and skepticism about the possibility of change. The practitioner role involved just the core members of the PST. These *TelSoft* employees were supportive in terms of setting up meetings and introducing us to people. At this stage, most *TelSoft* employees did more listening and responding instead of actively providing a vision of something different that needed to be done in the future.

By the end of the initiative, the researchers' involvement changed from facilitative to collaborative (Baskerville and Wood-Harper 1998). The practitioners took more ownership and initiative in the SPI effort. During each intervention cycle, the PST increased participation of *TelSoft* employees at all levels of the organization through the improvement teams. The VP of

Software ran the PST independently of the GSU research team. Finally, my insider status shifted away from the software engineers toward the upper management team. While non-management employees were still forthright when I asked questions of them directly, my access to divergent opinions became much less frequent. At the same time, upper managers expressed company problems through unsolicited emails and “off-the-record” comments.

Documentation. Developing and maintaining a case study database enhances the reliability of qualitative research, permitting an independent audit of claims to be conducted (Yin 2003). Although the four research papers differ in data analysis approach, they strive to demonstrate a clear trace between data collected and conclusions drawn. Such traceability enhances the credibility of claims made during data analysis (Miles and Huberman 1994). In the *TelSoft* case, interviews, workshops, and meetings were recorded whenever feasible. The first sixteen SCG meetings were transcribed to support paper 4. A designated note taker created public meeting minutes for many of the PST and improvement team meetings. Other data sources included my reflective field notes, *TelSoft*'s process documentation database, and email messages between GSU and *TelSoft*. The detailed account of the problem solving cycle in Part III provides an overview of all data sources, how and when they were created, and how they related to interventions into software practices at *TelSoft*. Also, a complete list of documents in the case study database along with date created, primary author, and a brief description is provided in Appendix A of Part III. This extensive documentation of the problem solving cycle allows other researchers to recover the action research process as it unfolded (Checkland and Holwell 1998).

Control. Considering the nature of control in action research helps researchers evaluate project risks such as whether theory will be allowed to influence actions at the client site. Avison et al. (2001) describe control in terms of initiation, authority, and formalism. For the *TelSoft* case, *initiation* was client-driven which meant that *TelSoft*'s needs took priority over the need for research data collection. Because we were flexible regarding the actual research areas studied, this was not considered a problem. Since final *authority* on the project remained with the client, there was the risk that the suggested actions would be rejected by *TelSoft* managers as inappropriate. In our case, the research team respected the decisions of the managers, presented convincing arguments for research-oriented activities (e.g. REGPG assessment, sense-and-respond theoretical framing, recordings), and built trusting relationships over time. Therefore, our suggestions were carefully considered and well received. With respect to *formalism*, the memorandum of understanding included a clause that the project could be stopped at any time by either the client or research team. Having an agreement with the top level of the organization, CEO and VP of Software, was instrumental in maintaining the project even as key personnel changed throughout the project (e.g. Dr. Roy Johnson, original Division President, Division Director, and one of the original PST members).

Usefulness. In qualitative research in general, the applicability of the research findings to the field setting is considered a valuable indicator of quality (Miles and Huberman 1994). Given the goal of action research to deliver both to the scientific and practitioner communities (Rapoport 1970), the client's view of utility of the study becomes an important factor in determining the quality of action research.

Research Results The key research findings that were applied at *TelSoft* were from paper 1 (i.e. combining perceptions and processes during assessment) and paper 3 (i.e. implementing the SPR

model). Paper 2 and paper 4 were geared toward understanding practice after action had taken place; therefore, those results did not directly inform action at *TelSoft*.

In paper 1, the combined RE assessment was designed to prompt the RE assessment manager to consider both processes and perceptions. At *TelSoft*, the PST found this framework useful during the diagnosing phase as reported in (Napier, Mathiassen et al. 2006). The diagnostic report was validated by *TelSoft* managers as being accurate, and the resulting intervention strategy led to considerable improvements at *TelSoft*. I also used this framework when planning the final assessment as documented in chapter 5 of Part III. As the framework is tested in other settings, we will be able to judge its utility to other researchers and practitioners. A limitation of this work is that it does not provide a detailed description of the framework. This was due in part to the space constraints of the conference proceedings. When extending these ideas for a journal, we will consider adopting a design research approach (Hevner, March et al. 2004; Van Aken 2004) focused on creating more complete guidelines and recommendations.

In paper 3, the SPR principles and model describe how an organization can exploit knowledge from legacy processes during subsequent SPI initiatives. There were several ways that SPR helped *TelSoft* transition to more disciplined software process management. Implementing SPR allowed *TelSoft* to reduce its 75 legacy process to a more manageable 26. The PST created a list of valid software processes and began to actively manage them using the implementation and documentation statuses described in (Napier, Kim et al. under review). At the end of intervention cycle 2, the PST accepted responsibility for ensuring that these processes would become updated and meet the standards established by the process management team. Since that time, the PST has involved a variety of people throughout the organization to assist with SPR; for example, developers were asked to refine the coding guidelines for C++, Java, and REXX.

Practical results SPI success can be evaluated based upon a mixture of perceptions of SPI success as well as measures of organizational performance such as cost reduction, cycle time reduction, and customer satisfaction (Dyba 2005). Below, we summarize employee perceptions of SPI as well indicators of improved software practice at *TelSoft* in the seven improvement areas (summary appears in Table 13).

Overall, *TelSoft*'s management team was pleased with the SPI initiative as demonstrated in this email message from *TelSoft*'s Vice President of Software Development (dated 9/25/2006):

“[The collaboration] It has been a good education experience for most of the individuals in the software group, and by involving a large number of the software employees in the process improvement initiatives it has demonstrated to the entire group the importance of following a few key policies and processes to ensure that we have an appropriate level of control and repeatability to maintain a successful software business. We are seeing the benefits of the collaboration in better portfolio planning and coordination, improved customer relations, less internal strife over requirements management, fewer quality assurance (QA) cycles and increased transparency of our configuration management.”

Based upon the success of this first initiative, *TelSoft*'s management team funded an additional 12 month contract with the research team on enhancing project management skills.

During the learning phase, we developed an overall SPI assessment that included an REGPG assessment, employee and customer interviews, as well as an employee online questionnaire (see chapter 5 of Part III for details). The majority of employees agreed that the SPI initiative created either "some improvement" or "considerable improvement" in software practices (as shown in Figure 7). Broadly speaking, employees realized that process improvement was a legitimate activity that received significant management support as indicated by these remarks:

- "I think people are at least more in tune to the fact that process is important."
- "People think critically about our processes more now as a result of attention to these issues."

TelSoft made the most dramatic improvement in software configuration management (SCM) and quality assurance. With respect to SCM, the new software release process defined during intervention cycle 1 was consistently followed and allowed for early problem detection. In addition, *TelSoft* documented reliable procedures for building most of its software products which allowed them to rebuild the same version of software that its clients had. With respect to QA, the policy requiring the QA group to execute software builds was strictly followed and very positively perceived. Selected comments from employee questionnaire:

- "QA doing builds means they can trust the integrity of the builds"
- "I see much improvement in quality assurance and that entire process - more standardized than what we had done previously and with QA doing builds it has forced us to document all our build and deployment processes plus document release specifications."

TelSoft also made noticeable improvements in customer relationship management. The initiative emphasized the importance of maintaining a professional image. For instance, the customer relations team enhanced product packaging for all software releases and drafted a "Getting Started" brochure to be included with software packaging. In addition, *TelSoft* deliberately increased face-to-face time with major customers; as a consequence, these relationships improved. The software charter (i.e. reason for being, strategy, and policies) was communicated to customers via letter and, in some cases, in person. Selected comments from questionnaire:

- "Much less squawking from employees and customers."
- "Customer relations efforts - more focus on face/face and client communication channels; also presentation of our software has also improved - looks more professional now."

Although little to no change was perceived by employees for the remaining areas, there is still important evidence of improvement. With respect to requirements management, the REGPG assessment indicated that *TelSoft*'s overall requirements maturity increased from Initial (level 1) to Repeatable (level 2). In fact, *TelSoft* increased the percentage of best practices used in six of the eight requirements areas and improved all of its weak areas to average. Participants also agreed that *TelSoft* was more consistently documenting requirements on internal projects. With respect to software vision management and project portfolio management, the SCG developed and promoted the software strategy and division's reason for being; they developed a more systematic, critical evaluation of current projects and business opportunities; and they mapped

out release schedule for products in a more collaborative way. These activities provided a stronger foundation for continuous improvement at *TelSoft*.

Figure 7: Employee perception of overall SPI impact

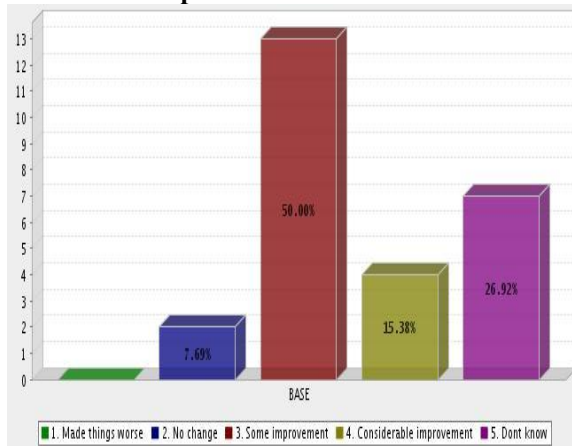


Table 13: Summary of Perceived Improvement

| <i>Improvement Area</i> | <i>Overall Assessment</i> |
|-----------------------------------|---------------------------|
| Software configuration management | Considerable improvement |
| Software quality assurance | Considerable improvement |
| Customer relations management | Some improvement |
| Requirements management | Little change |
| Software vision management | Little change |
| Project portfolio management | No change |
| End-user interaction | No change |

At the same time, this assessment revealed some limitations of the initiative. First, we could have improved communication between management and non-management. Despite interim status meetings, employees that did not participate on an SPI improvement team seemed unaware of the changes being made. This suggests a need to intentionally involve a broader set of individuals, particularly non-management employees, in the study. Second, there was still too much variation in the way that internal projects were managed which caused them to be over budget and *TelSoft* to losing propositions. After the initial diagnosis, *TelSoft* went through another round of layoffs, losing its dedicated business analysts. Through the improved practices, *TelSoft* became more consistent about having explicit and well-managed requirements for internal projects. However, the quality of those requirements was not always high – they were sometimes incomplete, did not consider what could go wrong, or did not involve inputs from experienced software personnel. *TelSoft*'s managers need to continue to monitor and take corrective action on these problems. Third, *TelSoft* developed more plans and processes than they had resources to implement. As one questionnaire respondent suggested: "Slow things down somewhat - we probably really need to fully implement the initiatives prior to moving on to another round. Or, I guess you could also say speed things up on the implementation."

Theory. Action research is distinguished from consulting by the use of theory to inform action and the application of theoretical frameworks to interpret findings (Baskerville and Wood-Harper 1996). At *TelSoft*, the overall SPI initiative was informed by SPI theory in general (McFeeley 1996; Mathiassen, Pries-Heje et al. 2002; Dyba 2005) and the sense-and-respond framework (Haeckel 1995; Haeckel 1999) in particular as documented in chapter 3 of Part III. In addition, each paper drew from a specific theoretical base and developed theoretical frameworks as detailed in chapter 4 above. In paper 1, we developed an RE assessment framework that combines perception-based and process-based data. In paper 2, we developed two theories underlying the debate on plan-driven versus agile development. In paper 3, we defined SPR, developed principles for conducting SPR, and presented an SPR model. In paper 4, we described the process of building ambidextrous capability by focusing on project portfolio management.

Transfer. This work is based upon a study within a single software company with a particular set of characteristics (e.g. low software process maturity, small setting, low organizational maturity, etc.). We have argued that using a single case is appropriate given the nature of action research, the similar characteristics that *TelSoft* shares with other small software organizations, and the benefits of being able to explore longitudinal data. A limitation of this choice is that we are unable to directly demonstrate that our conclusions will transfer in other settings. However, we have included rich descriptions of the settings, processes and actions to establish external validity, the domain to which findings can be generalized (Yin 2003). As further studies are conducted that use these frameworks and ideas, we will be able to evaluate the applicability of these findings for a variety of settings.

5.3 Implications

The key implications of this study are that improvement of software practices can benefit from:

- 1) Identifying dualities,
- 2) Appreciating the context,
- 3) Seeking ambidexterity at multiple levels, and
- 4) Re-conceptualizing ambidextrous software organizations.

Below, we discuss each of these implications from the standpoint of managers in charge of improving software practices as well as researchers developing theories of SPI.

1) Identify Dualities. By identifying dualities and designing interventions, we found creative alternatives to dominant paradigms and were able to integrate multiple perspectives. We have demonstrated how taking this approach allowed us to obtain richer insights at *TelSoft* for RE assessment, SPI action planning, and software process management (Napier, Mathiassen et al. 2006; Napier, Mathiassen et al. 2006; Napier, Kim et al. under review). Directly applying these results provides specific implications for SPI managers. For example, with respect to RE assessment, SPI managers could design an assessment plan that considers a mixture of perceptions and processes. For SPI action planning, managers could evaluate their diagnosis data from the standpoint of first the repeat-ability perspective and then the response-ability perspective to increase the variety and quality of the recommendations. When establishing software process management, managers should consider using the SPR model to exploit learning from legacy processes. Future research can build upon the theories and frameworks presented in this study by validating them in other settings.

In general, this research suggests that SPI managers and teams should intentionally look for dualities during each phase of the SPI process (McFeeley 1996). Once dualities have been identified, managers should consider how to embrace these tensions and integrate seemingly contradictory elements. These managers, therefore, need to become better at paradoxical thinking which considers *both* option A *and* option B instead of *either* option A *or* option B (Collins and Porras 1994; Smith and Tushman 2005). Future research could develop strategies for SPI managers who face these dualities.

2) Appreciate the Context. Organizational context refers to the environment in which the software firm operates as well as the systems, processes, and beliefs within the organization through which ideas for change have to proceed (Gibson and Birkinshaw 2004). Throughout this research, important aspects of *TelSoft's* organizational context influenced our approach to improving practices. For example, after learning that *TelSoft's* prior experience with SW-CMM had created legacy processes, we implemented SPR (Napier, Kim et al. under review); realizing that *TelSoft* valued being responsive to customers, we selected Haeckel's (1995; 1999) sense-and-respond framework to drive improvements; recognizing some skepticism among *TelSoft's* employees about the ability to change, we created improvement teams with employees from all levels of the organization and used a variety of methods to disseminate information about the initiative; and considering *TelSoft's* limited resources, we sought an alternative to structural

approaches to achieving ambidexterity (Napier, Mathiassen et al. under review). Ignoring these aspects of *TelSoft*'s context would have led us to apply generic solutions and blinded us to the need for SPR and the potential usefulness of contextual ambidexterity.

The initial diagnosis of context can provide critical information for SPI. First, the analysis of the initial context can dramatically influence the implementation plan for the overall improvement initiative. For organizations, like *TelSoft*, that are diagnosed as weak in performance management but stronger at social support, Gibson and Birkinshaw (2004) recommend to focus first on performance management; by contrast, organizations with weak social support are recommended to first work at increasing trust and support. Second, SPI managers could intervene to intentionally shape the organizational context.. For example, SPI managers could adopt the goal of increasing contextual ambidexterity by following a four-step process of diagnosing, visioning, implementing, and performing (Napier, Mathiassen et al. under review). Using this approach, SPI managers would explicitly measure the organizational context across each phase, develop improvement goals, and design effective interventions. At *TelSoft*, we used contextual ambidexterity retrospectively to analyze the SCG's actions with respect to project portfolio management, but we used sense-and-respond framework and general SPI theory to guide actions in the overall improvement initiative. An interesting possibility for future research would be to conduct an action research study in which contextual ambidexterity is the driver for change.

Another area for future research involves how organizational context is measured. Although Gibson and Birkinshaw's (2004) model of organizational context consisted of two constructs, future research can explore whether other aspects of organizational context are more salient. For example, we found that *TelSoft* was particularly impacted by historical events such as the prior SPI initiative and unsuccessful product innovation attempts.

3) Seek Ambidexterity at Multiple Levels. The software community has approached the idea of ambidexterity from primarily two levels: 1) creating ambidextrous projects that are both rigorous and agile (Boehm 2002; Lee, DeLone et al. 2006; Lee, DeLone et al. 2007) or 2) developing ambidextrous organizations that have separate sub-units focused on either discipline or agility (Vinekar, Slinkman et al. 2006). Ambidexterity can also be a characteristic of individuals within the organization. In addition, this research has looked at ambidexterity from the perspective of specific software practices. With project portfolio management practices, we designed the SCG to focus on both managing projects with its existing customer base and obtaining new customers.

Future research could develop a framework for understanding ambidexterity that takes these multiple levels under consideration. This is particularly true when studying the process of increasing ambidexterity. As SPI managers engage in action planning, they need theories that can guide them toward increasing ambidexterity within their organizations. Questions for future research that looks across levels include: What level of ambidexterity has the biggest impact on performance? What is the relative importance of ambidexterity at each level? How does ambidexterity at one level relate to ambidexterity at another level? Is there a preferred sequence for building ambidexterity at these levels? Is it possible to have an ambidextrous organization without having ambidextrous individuals?

4) Re-conceptualize Ambidextrous Software Organizations. Prior to this research, the term “ambidextrous software organization” has been defined in terms of an agile and traditional sub-unit with separate cultures and practices (Vinekar, Slinkman et al. 2006). However, we have argued that this structural approach to achieving ambidexterity is not feasible for all software organizations and have presented contextual ambidexterity as an alternative. Under certain circumstances, instead of accepting two separate cultures within software organizations, managers could focus on building a single culture that facilitates ambidexterity. Future research could provide more specific guidance for the conditions in which one form of ambidexterity is preferred over the other. At the same time, future research could consider the extent to which structural and contextual ambidexterity can be effectively integrated within a single organizations: In what ways can software organization combine structural and contextual ambidexterity? What is the impact of these various ambidextrous forms on organizational performance?

Research Summary. This work goes beyond the discipline-agility software debate to broaden our understanding of the dualities involved in improving software practice. We identified three new dualities in the areas of RE assessment, SPI action planning, and software process management; and we applied the existing duality of alignment-adaptability to project portfolio management and the entire SPI effort. We argued for the limitations of applying structural ambidexterity solutions within small software organizations; instead, we adopted an alternative view of ambidextrous software organizations based upon contextual ambidexterity. We demonstrated the feasibility of applying the contextual ambidexterity lens through a detailed case study showing the process of improving project portfolio management at *TelSoft*. Overall, we suggest that software organizations can be improved by creating a conducive, organizational context and by iteratively increasing the alignment and adaptability of vital software practices.

References

- Adler, P. S., B. Goldoftas, et al. (1999). "Flexibility versus efficiency? A case study of model changeovers in the Toyota production system." Organization Science **10**(1): 43-68.
- Adler, P. S., F. McGarry, et al. (2005). "Enabling process discipline: lessons from the journey to CMM level 5." MIS Quarterly Executive **4**(1): 215-226.
- Agile Alliance. (2001). "Manifesto for agile software development." Retrieved May, 2006, from <http://www.agilemanifesto.org/>.
- Argyris, C. (1985). Action Science. San Francisco, Jossey-Bass.
- Avison, D., R. Baskerville, et al. (2001). "Controlling action research projects." Information Technology & People **14**(1): 28-45.
- Avison, D., F. Lau, et al. (1999). "Action research." Communications of the ACM **42**(1): 94-97.
- Avison, D. and T. Wood-Harper (1990). Multiview: an exploration in information systems development. New York, McGraw-Hill.
- Baker, S. (2005). Formalizing agility: an agile organization's journey toward CMMI accreditation. Agile Conference.
- Barley, S. R. and G. Kunda (2001). "Bringing work back in." Organization Science **12**(1): 76-95.
- Baskerville, R. and T. Wood-Harper (1996). "A critical perspective on action research as a method for information systems research." Journal of Information Technology **11**: 235-246.
- Baskerville, R. and T. Wood-Harper (1998). "Diversity in information systems action research methods." European Journal of Information Systems **7**(2): 90-107.
- Baskerville, R. L. (1999). "Investigating information systems with action research." Communications of the AIS **2**(19): 1-32.
- Baum, J. A. C., S. X. Li, et al. (2000). "Making the next move: How experiential and vicarious learning shape the locations of chains' acquisitions." Administrative Science Quarterly **45**(4): 766-801.
- Beck, K. (1999). Extreme Programming Explained: Embrace Change. Reading, MA, Addison-Wesley.
- Beecham, S., T. Hall, et al. (2005). "Defining a requirements process improvement model." Software Quality Journal **13**(3): 247-279.
- Benner, M. J. and M. Tushman (2002). "Process management and technological innovation: A longitudinal study of the photography and paint industries." Administrative Science Quarterly **47**(4): 676-709.
- Benner, M. J. and M. L. Tushman (2003). "Exploitation, exploration, and process management: The productivity dilemma revisited." Academy of Management Journal **28**(2): 238-256.
- Birkinshaw, J. and C. Gibson (2004). "Building ambidexterity into an organization." Sloan Management Review **45**(4): 47-55.
- Boehm, B. W. (2002). "Get ready for agile methods, with care." Computer **35**(1): 64-69.
- Boehm, B. W. and R. Turner (2004). Balancing agility and discipline: a guide for the perplexed. Boston, Addison-Wesley.
- Checkland, P. and S. Holwell (1998). "Action research: its nature and validity." Systemic Practice and Action Research **11**(1): 9-21.
- Clark, K. B. and S. C. Wheelwright (1992). "Organizing and leading 'heavyweight' development Teams." California Management Review **34**(3): 9-28.

- CMMI Product Team (2002). CMMI for Systems Engineering/Software Engineering/Integrated Product and Process Development/Supplier Sourcing, Software Engineering Institute.
- Cockburn, A. (2000). Writing Effective Use Cases. Reading, MA, Addison-Wesley.
- Collins, J. C. and J. I. Porras (1994). Built to Last: Successful Habits of Visionary Companies. New York, HarperCollins Publishers.
- Cook, S. D. N. and J. S. Brown (1999). "Bridging epistemologies: The generative dance between organizational knowledge and organizational knowing." Organization Science **10**(4): 381-400.
- Cunningham, J. B. (1993). Action research and organizational development. Westport, CT, Praeger.
- Curtis, B., W. E. Hefley, et al. (2002). People Capability Maturity Model: Guidelines for Improving the Workforce. Boston, Addison-Wesley.
- Davison, R. M., M. G. Martinsons, et al. (2004). "Principles of canonical action research." Information Systems Journal **14**(1): 65-86.
- De Reyck, B., Y. Grushka-Cockayne, et al. (2005). "The impact of project portfolio management on information technology projects." International Journal of Project Management **23**(7): 524-537.
- Deming, W. E. (1986). Out of Crisis. Cambridge, MA, MIT Center of Advanced Engineering Study.
- Duncan, R. B. (1976). "The ambidextrous organization: Designing dual structures for innovation." Kilmann, RH Pondy, LR, and DP Slevin (eds.). The Management of Organization Design, I, New York: Elsevier North-Holland: 167-188.
- Dyba, T. (2005). "An empirical investigation of the key factors for success in software process improvement." IEEE Transactions on Software Engineering **31**(5): 410.
- El Emam, K. and A. Birk (2000). "Validating the ISO/IEC 15504 measure of software requirements analysis process capability." IEEE Transactions on Software Engineering **26**(6): 541.
- Floyd, S. and P. Lane (2000). "Strategizing throughout the organization: Managing role conflict in strategic renewal." Academy of Management Review **25**: 154-177.
- Fuggetta, A. (2001). Software process: A roadmap. Software Process Improvement. R. Hunter and R. H. Thayer. Los Alamitos, CA, IEEE Computer Society: 559-566.
- Ghoshal, S. and C. A. Bartlett (1994). "Linking organizational context and managerial action: The dimensions of quality of management." Strategic Management Journal **15**(Summer): 91-112.
- Gibson, C. and J. Birkinshaw (2004). "The antecedents, consequences, and mediating role of organizational amidexterity." Academy of Management Journal **47**(2): 209-226.
- Goldenson, D. R. and J. D. Herbsleb (1995). After the appraisal: a systematic survey of process improvement, its benefits, and factors that influence success. Pittsburgh, PA, Software Engineering Institute.
- Gregor, S. (2006). "The nature of theory in information systems." MIS Quarterly **30**(3): 611-642.
- Gupta, A., K. Smith, et al. (2006). "The interplay between exploration and exploitation." Academy of Management Journal **49**(4): 693-706.
- Haeckel, S. (1995). "Adaptive enterprise design: the sense-and-respond model." Planning Review **23**(3): 6-13, 42.
- Haeckel, S. (1999). Adaptive Enterprise: Creating and Leading Sense-and-Respond Organizations. Boston, MA, Harvard Business School Press.

- He, Z.-L. and P.-K. Wong (2004). "Exploration vs. exploitation: An empirical test of the ambidexterity hypothesis." Organization Science **15**(4): 481-494.
- Hevner, A. R., S. T. March, et al. (2004). "Design Science in Information Systems Research." MIS Quarterly **28**(1): 75-105.
- Highsmith, J. (2000). Adaptive Software Development: A Collaborative Approach to Managing Complex Systems. New York, NY, Dorset House Publishing.
- Highsmith, J. and A. Cockburn (2001). "Agile software development: The business of innovation." Computer **34**(9): 120-127.
- Hobday, M. (2000). "The project-based organisation: an ideal form for managing complex products and systems." Research Policy **29**(7-8): 871-893.
- Horvat, R. V., I. Rozman, et al. (2000). "Managing the complexity of SPI in small companies." Software Process: Improvement and Practice **5**(1): 45-54.
- Humphrey, W. S. (1989). Managing the Software Process. Boston, MA, Addison-Wesley.
- Iversen, J., L. Mathiassen, et al. (2004). "Managing risk in software process improvement: an action research approach." MIS Quarterly **28**(3): 395-433.
- Iversen, J., P. A. Nielsen, et al. (2002). Problem diagnosis in SPI. Improving Software Organizations: From Principles to Practice. L. Mathiassen, J. Pries-Heje and O. Ngwenyama. New York, Addison-Wesley.
- Jansen, J. J. P., F. A. J. van Den Bosch, et al. (2005). "Exploratory innovation, exploitative innovation, and ambidexterity: The impact of environmental and organizational antecedents." Schmalenbach Business Review **57**: 351-363.
- Kautz, K. H., H. W. Hansen, et al. (2000). Applying and adjusting a software process improvement model in practice: The use of the IDEAL model in a small software enterprise. International Conference on Software Engineering, Limerick, Ireland.
- Kock, N. (1997). "Negotiating mutually satisfying IS action research topics with organizations: An analysis of Rapoport's initiative dilemma." Journal of Workplace Learning **9**(7): 253-62.
- Krasner, H., J. Terrel, et al. (1992). "Lessons learned from a software process modeling system." Communications of the ACM **35**(9): 91-100.
- Kuvaja, P. and A. Bicego (1994). "BOOTSTRAP - a European Assessment Methodology." Software Quality Journal **3**(3): 117-127.
- Langley, A. (1999). "Strategies for theorizing from process data." Academy of Management Review **24**(4): 691-710.
- Lee, G., W. DeLone, et al. (2006). "Ambidextrous coping strategies in globally distributed software development projects." Communications of the ACM **49**(10): 35-40.
- Lee, G., W. DeLone, et al. (2007). "Ambidexterity and global IS project success: A theoretical model." 40th Annual Hawaii International Conference on System Sciences 44-54.
- Lincoln, Y. and E. Guba (1985). Naturalistic Inquiry. Newbury, CA, Sage.
- Lindgren, R., O. Henfridsson, et al. (2004). "Design principles for competence management systems: A synthesis of an action research study." MIS Quarterly **28**(3): 435-472.
- Lubatkin, M. H., Z. Simsek, et al. (2006). "Ambidexterity and performance in small-to medium-sized firms: The pivotal role of top management team behavioral integration." Journal of Management **32**(5): 646-672.
- Lyytinen, K. (1988). "Stakeholders, IS failures and soft system methodology: an assessment." Journal of Applied Systems Analysis **15**: 61-81.

- March, J. G. (1991). "Exploration and exploitation in organizational learning." Organization Science **2**(1): 71-87.
- Markowitz, H. (1952). "Portfolio Selection." The Journal of Finance **7**(1): 77-91.
- Mårtensson, P. and A. S. Lee (2004). "Dialogical action research at Omega Corporation." MIS Quarterly **28**(3): 507-536.
- Mason, J. (2002). Qualitative Researching. London, Sage.
- Mathiassen, L. (2002). "Collaborative practice research." Information Technology & People **15**(4): 321-345.
- Mathiassen, L., J. Pries-Heje, et al. (2002). Improving Software Organizations: From Principles to Practice. Boston, MA, Addison-Wesley.
- Mathiassen, L. and A. M. Vainio (2007). "Dynamic capabilities in small software firms: a sense-and-respond approach." IEEE Transactions on Engineering Management **54** (3): 522-538.
- McFarlan, F. W. (1981). "Portfolio approach to information systems." Harvard Business Review **59**(5): 142-150.
- McFeeley, B. (1996). IDEAL: A user's guide for software process improvement. Pittsburgh, PA, Software Engineering Institute.
- McKay, J. and P. Marshall (2001). "The dual imperatives of action research." Information Technology & People **14**(1): 46-59.
- Miles, M. B. and A. M. Huberman (1994). Qualitative Data Analysis: an expanded sourcebook. Thousand Oaks, Sage Publications.
- Mingers, J. (2001). "Combining IS research methods: Towards a pluralist methodology." Information Systems Research **12**(3): 240-259.
- Mingers, J. and A. Gill (1997). Multimethodology: The theory and practice of combining management science methodologies. Chichester, John Wiley & Sons.
- Napier, N. P., J. Kim, et al. (under review). "Software process reengineering: A model and its application to an industrial case study." IEEE Transactions on Software Engineering.
- Napier, N. P., L. Mathiassen, et al. (2006). Negotiating Response-ability and Repeat-ability in Requirements Engineering. International Conference on Information Systems, Milwaukee, Wisconsin.
- Napier, N. P., L. Mathiassen, et al. (2006). Perceptions and Processes in assessing software requirements practices. Proceedings of the Twelfth Americas Conference on Information Systems, Acapulco, Mexico.
- Napier, N. P., L. Mathiassen, et al. (under review). "Becoming ambidextrous: A contextualist inquiry into a small software firm." Organization Science.
- Nielsen, P. A. and J. Pries-Heje (2002). A framework for selecting an assessment strategy. Improving software organizations: from principles to practice. L. Mathiassen, J. Pries-Heje and O. Ngwenyama, Addison-Wesley.
- O'Reilly III, C. A. and M. Tushman (2004). "The ambidextrous organization." Harvard Business Review **82**(4): 74-81.
- Orlikowski, W. and J. J. Baroudi (1991). "Studying information technology in organizations: research approaches and assumptions." Information Systems Research **2**(1): 1-28.
- Overby, E., A. Bharadwaj, et al. (2006). "Enterprise agility and the enabling role of information technology." European Journal of Information Systems **15**(2): 120-131.
- Paulk, M. (2001). "Extreme programming from a CMM perspective." IEEE Software **18**(6): 19-26.

- Paulk, M., B. Curtis, et al. (1993). Capability Maturity Model for Software, Version 1.1. Pittsburgh, PA, Software Engineering Institute.
- Paulk, M., C. V. Weber, et al., Eds. (1995). The Capability maturity model: guidelines for improving the software process. SEI Series in Software Engineering. Boston, Addison-Wesley.
- Pettigrew, A. M. (1985). Contextualist Research: A Natural Way to Link Theory and Practice. Doing Research That is Useful for Theory and Practice. E. E. Lawler. San Francisco, Jossey-Bass.
- Pettigrew, A. M. (1987). "Context and action in the transformation of the firm." Journal of Management Studies **24**(6): 649-670.
- Pettigrew, A. M. (1990). "Longitudinal field research on change: theory and practice." Organization Science **1**(3): 267-292.
- Pouloudi, A. and E. Whitley (1997). "Stakeholder identification in inter-organizational systems: gaining insights for drug use management systems." European Journal of Information Systems **6**(1): 1-14.
- Ramesh, B., J. Pries-Heje, et al. (2002). Internet software engineering: a different class of processes. Annals of software engineering, Kluwer Academic Publishers. **14**: 169-195.
- Rapoport, R. (1970). "Three dilemmas in action research." Human Relations **23**(6): 499-513.
- Rising, L. and N. S. Janoff (2000). "The Scrum software development process for small teams." IEEE Software **17**(4): 26-32.
- Rout, T. P. (1995). "SPICE: A framework for software process assessment." Software Process: Improvement and Practice **1**(1): 57-66.
- Salo, O. and P. Abrahamsson (2005). Integrating agile software development and software process improvement: a longitudinal case study. International Symposium on Empirical Software Engineering.
- Schwaber, K. and M. Beedle (2001). Agile Software Development with Scrum. Upper Saddle River, Prentice Hall.
- Smith, W. K. and M. L. Tushman (2005). "Managing strategic contradictions: A top management model for managing innovation streams." Organization Science **16**(5): 522-536.
- Software Engineering Institute (2006). Improving Processes in Small Settings (IPSS): A White Paper. Pittsburgh, PA, Carnegie Mellon University.
- Sommerville, I. and J. Ransom (2005). "An empirical study of industrial requirements engineering process assessment and improvement." ACM Transactions on Software Engineering and Methodology **14**(1): 85-117.
- Sommerville, I. and P. Sawyer (1997). Requirements Engineering: A Good Practice Guide. New York, NY, John Wiley & Sons.
- Susman, G. and R. Evered (1978). "An assessment of the scientific merits of action research." Administrative Science Quarterly **23**(4): 582-603.
- The Standish Group International. (2004). "2004 Third Quarter Research Report." from URL http://standishgroup.com/sample_research/PDFpages/q3-spotlight.pdf.
- Turk, D., R. France, et al. (2005). "Assumptions underlying agile software-development processes." Journal of Database Management **16**(4): 62-87.
- Tushman, M. and C. A. O'Reilly III (1996). "Ambidextrous organizations: Managing evolutionary and revolutionary change." California Management Review **38**(4): 8-30.

- Van Aken, J. E. (2004). "Management research based on the paradigm of the design sciences: The quest for field-tested and grounded technological rules." Journal of Management Studies **41**(2): 219-246.
- Van de Ven, A. and M. Poole (1995). "Explaining development and change in organizations." Academy of Management Review **20**(3): 510-540.
- Vidgen, R. (1997). "Stakeholders, soft systems and technology: separation and mediation in the analysis of information system requirements." Information Systems Journal **7**(1): 21-46.
- Vinekar, V., C. W. Slinkman, et al. (2006). "Can agile and traditional systems development approaches coexist? An ambidextrous view." Information Systems Management **23**(3): 31-42.
- Weitzman, E. A. and M. B. Miles (1995). Computer Programs for Qualitative Data Analysis. Thousand Oaks, CA, Sage.
- Wenger, E. (1998). Communities of practice: Learning, meaning, and identity. Cambridge, Cambridge University Press.
- Yin, R. K. (2003). Case Study Research: Design and Methods. Thousand Oaks, Sage.
- Zbaracki, M. J. (1998). "The rhetoric and reality of total quality management." Administrative Science Quarterly **43**(3): 602-636.

Part II: Research Papers

Paper 1: Combining Perceptions and Processes

Title: Perceptions and Processes in Assessing Software Requirements Practices

This paper is coauthored by Nannette Napier, Lars Mathiassen, and
Roy D. Johnson

This version of the paper was accepted and presented at the
America's Conference on Information Systems (AMCIS),
Acapulco, Mexico, 2006

Abstract

Requirements engineering is a key discipline in analysis and design of business software. There are commonly accepted processes available for requirements engineering, but many organizations struggle to implement and follow these processes. A number of methods have therefore been developed to help assess and improve requirements practices. This exploratory study reports from a project at *TelSoft* in which we combined process assessments and stakeholder perceptions to arrive at recommendations for improving requirements practices. The paper presents the combined approach, experiences from using the approach at *TelSoft*, and the resulting insights and recommendations. On that basis, we offer a critical evaluation of the dominant process-driven approach and show how requirements assessment can benefit from the perceptions and active participation of key stakeholders.

Keywords

Requirements engineering assessment, process models, stakeholder perceptions, Requirements Engineering Good Practice Guide (REGPG).

Introduction

Requirements Engineering (RE) covers all aspects of the discovery, documentation, and maintenance of software requirements throughout the software development lifecycle (Kotonya and Sommerville, 1998). RE is a key discipline in analysis and design of business software. Companies looking to improve their RE practices may seek guidance from the Software Engineering Institute's Capability Maturity Model Integration (CMMI Product Team, 2002). This model defines two key process areas – Requirements Management and Requirements Development – directly related to requirements engineering and lists best practices in these areas. Despite the existence of these process descriptions and best practices, many organizations struggle to implement and follow these procedures. In fact, an expert panel consisting of both practitioners and academics agreed that the RE process is the most problematic of all software engineering activities (Beecham, Hall, Britton, Cottee and Rainer, 2005a). Furthermore, practicing software project managers ranked the problem of misunderstood software requirements as their second most important risk to be managed (Schmidt, Lytinen, Keil and Cule, 2001).

Companies seeking to improve their RE practices are recommended to assess these practices to identify strengths and weaknesses and help focus the improvement efforts (Curtis and Paulk, 1993; Humphrey, 1989). A number of methods have been developed to that end (e.g., Beecham, Hall and Rainer, 2005b; El Emam and Madhavji, 1995; Sommerville and Sawyer, 1997). While there are important variations between these assessment approaches, they all rely on the basic idea that current practices are best assessed and improved by benchmarking against best practices. This process-driven approach to assess RE practices has obvious advantages, but it ignores two important lessons from organizational learning. First, organizational stakeholders' perceptions of current and future practices are important sources for innovation and learning. Second, participatory approaches increase buy-in and thereby facilitate successful implementation of new practices.

This research is therefore designed to explore how assessments of RE practices can benefit from the perceptions and active participation of key stakeholders. To this end, we conducted a systematic assessment of RE practices in a small software firm, *TelSoft*, addressing the following research questions:

1. What different insights are gained from process- and perception-driven assessments of RE practices?
2. How can processes and perceptions be combined in assessment of RE practices?

Theoretical Background

In the following, we review existing process-driven approaches to assess RE practices and outline the theoretical basis for perception-driven approaches.

Current approaches to RE assessment

Researchers have used three main approaches to RE assessment: analyzing the RE-related data from generic software process assessments (e.g., SW-CMM or ISO/IEC 15504); applying a RE-specific version of the SW-CMM; and, measuring adherence to best practices based on a dedicated RE maturity model.

The first approach relies on general models for software process assessment. For example, El Emam and Birk (2000) used a subset of the assessment data collected from 44 organizations during the ISO/IEC 15504 trials (Simon, 1996) to examine whether the Software Requirements Analysis process capability is positively related to overall project performance. Damian et al. (2004) similarly studied the benefits of RE process improvement using SW-CMM mini-assessments.

The second approach relies on specific RE models. Beecham and colleagues have developed a RE model based upon the SW-CMM called R-CMM (Beecham et al., 2005b). Their approach is based on the Goal-Question-Metric paradigm (Basili and Rombach, 1988). They associate high-level RE goals with the different maturity levels from initial (level 1) to optimizing (level 5). An example of a high-level goal to achieve level 2 is “to implement a repeatable RE process” (Beecham et al., 2005b). Related to each goal is a set of assessment questions to ask about RE processes and their relation to best practices. Weaknesses pointed out in the analysis are then used to suggest RE improvement goals.

The third approach is uniquely focused on RE as suggested in the Requirements Engineering Good Practice Guide (REGPG) (Sommerville et al., 1997). The REGPG describes 66 RE practices within eight areas of RE – requirements documents, requirements elicitation, requirements analysis and negotiation, describing requirements, system modeling, requirements validation, requirements management, and requirements engineering for critical systems. Each normative practice is related to one of three levels of maturity: basic, intermediate, or advanced. The assessment rates how each practice is adopted within the organization: not used, discretionary based upon the project manager, normally used, or standardized throughout the organization. A score is then calculated to create an overall assessment of the organization’s RE maturity level. The REGPG has been used to assess ERP RE processes (Daneva, 2002; Daneva,

2003), to develop a formal assessment instrument (Niazi, 2005), and to suggest general success criteria for RE improvements (Kauppinen, Aaltio and Kujala, 2002; Kauppinen, Vartiainen, Kontio, Kujala and Sulonen, 2004). Sommerville and Ransom (2005) provide recommendations for adapting the model such as having domain-specific assignment of practices to maturity levels; creating domain-specific versions of the model; and, focusing on the business benefits of improving RE practice.

While there are important variations between these assessment approaches, they all analyze the gap between standardized RE processes and current practices. A process model drives data collection and analysis; specifies which practices should be adopted; and, outlines priorities to effectively increase RE maturity. Although all three approaches acknowledge the importance of tailoring assessments to organizational needs, they each assume that RE is best assessed and improved by benchmarking against best practices (Nielsen and Pries-Heje, 2002).

An alternative approach

An alternative approach to RE assessment would privilege perceived problems over prescribed processes (Nielsen et al., 2002). In this approach, stakeholder perceptions about strengths, weaknesses, and opportunities related to RE activities and artifacts drive data collection and analysis; stakeholders, rather than models, determine what is important to study by assigning priorities to problems; and, solutions are grounded in the specific context of the problematic situation.

Such a perception-based approach borrows from general stakeholder analysis (Lyytinen, 1988; Pouloudi and Whitley, 1997; Vidgen, 1997). Like interpretive research, stakeholder analysis considers organizational actors' subjective meanings as important knowledge sources; therefore, they emphasize the specific terms and perceptions of each stakeholder and avoid presenting a priori concepts (Orlikowski and Baroudi, 1991). Soft Systems Methodology (SSM) is an example of a qualitative, interpretive approach to study information systems issues based on stakeholder perceptions (Checkland and Scholes, 1999; Frederiksen and Mathiassen, 2005).

The process-driven approach to assess RE practices has obvious advantages: it provides the organization with new insights on RE; it makes comparisons across organizations feasible; it supports a structured and easy-to-adopt assessment approach; and, it leads to an immediate set of recommendations for improvement. However, organizational stakeholders' perceptions of current and future practices are also important sources for innovation and learning. Furthermore, process-driven approaches do not engage stakeholders in ways that increase buy-in and facilitate successful implementation of new practices. For these reasons, we recommend combining process-driven and perception-driven approaches. Methodological pluralism is appropriate for RE assessment because highly complex real-world problems call for multiple perspectives to understand their richness (Mingers, 2001; Mingers and Gill, 1997).

A combined approach

Our combined approach to RE assessment consists of three steps: initiating the assessment, executing multiple inquiry cycles, and making recommendations based upon the findings.

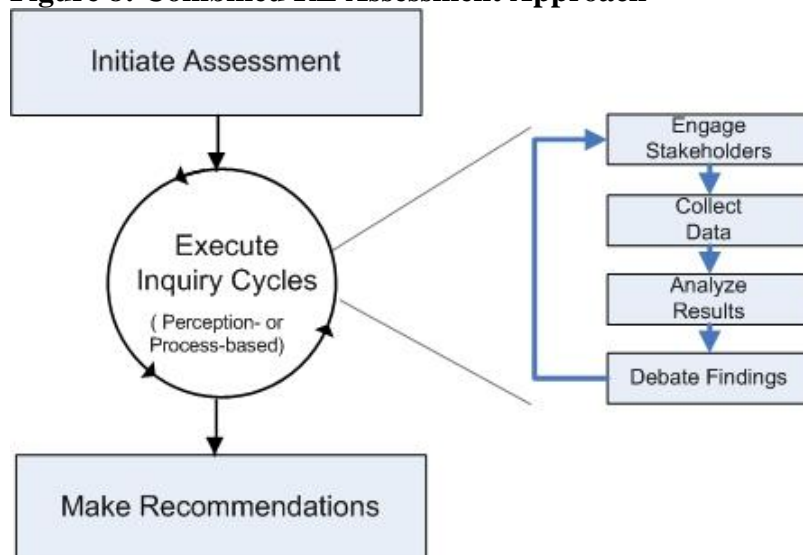
First, the RE assessment is initiated. Prior literature has identified several success factors for RE process improvement, including management support, motivation and commitment of other employees, and a systematic implementation strategy (Kauppinen et al., 2004). Therefore, it is

important to actively involve key stakeholders in the assessment and provide adequate structure when designing the assessment. The objective of this step is to establish commitment, select an assessment strategy, and agree on an overall plan for the inquiry cycles and the recommendation step. Three dimensions to consider when selecting an assessment strategy include required level of rigor, degree of reliance on a specific process model, and whether outside consultants should lead the assessment (Nielsen et al., 2002). The output of this step is commitments from key stakeholders to an RE assessment plan.

The next step is to understand the current state of RE practice through a series of inquiry cycles. Each inquiry cycle, whether perception-driven or process-driven, involves engaging stakeholders, collecting data, analyzing data, and debating findings. Perception-driven inquiry captures data about individual beliefs and experiences in the specific context of the problematic situation. Process-driven inquiry captures data on how current practices benchmark against pre-defined processes, best practice, and pre-defined questions. In all cases, information learned from each cycle feeds into the next inquiry cycle. The outcomes from this step include a prioritized list of problems as well as opportunities for improvement.

Finally, the knowledge learned from the inquiry cycles is used to make recommendations. A feasible approach to turning these insights into improved requirements practices is to align with the organizations priorities, traditions, and culture. It is also important to show business benefit to the proposed initiatives (Kauppinen et al., 2004; Sommerville et al., 2005). To ensure this, the recommendations should suggest an overall improvement strategy, establish project teams that focus on making visible, short-term investments in requirements practices, and consider the appropriate sequencing of improvement efforts (Humphrey, 1989).

Figure 8: Combined RE Assessment Approach



Research Method

We adopted a case study (Yin, 2003) based on action research (Baskerville, 1998; Rapoport, 1970; Susman and Evered, 1978). This allowed us to discover differences in insights from process- and perception-driven assessments and to explore practical ways to combine the two perspectives into a comprehensive RE assessment approach. In this section, we provide background information about the research site and describe the research approach in detail.

Research site

TelSoft was founded in 1971, with the mission to be a premier software services firm in the telecommunications and utility industries. The company has approximately 500 employees with 50 dedicated to software development. Many of the same employees that helped found the organization 35 years ago are still employed, bringing both a wealth of experience and old habits. One of the authors had previously worked at *TelSoft*, which allowed the research team immediate and deep engagement. It also provided a solid understanding of the context and acceptance of the R&D collaboration by *TelSoft* employees.

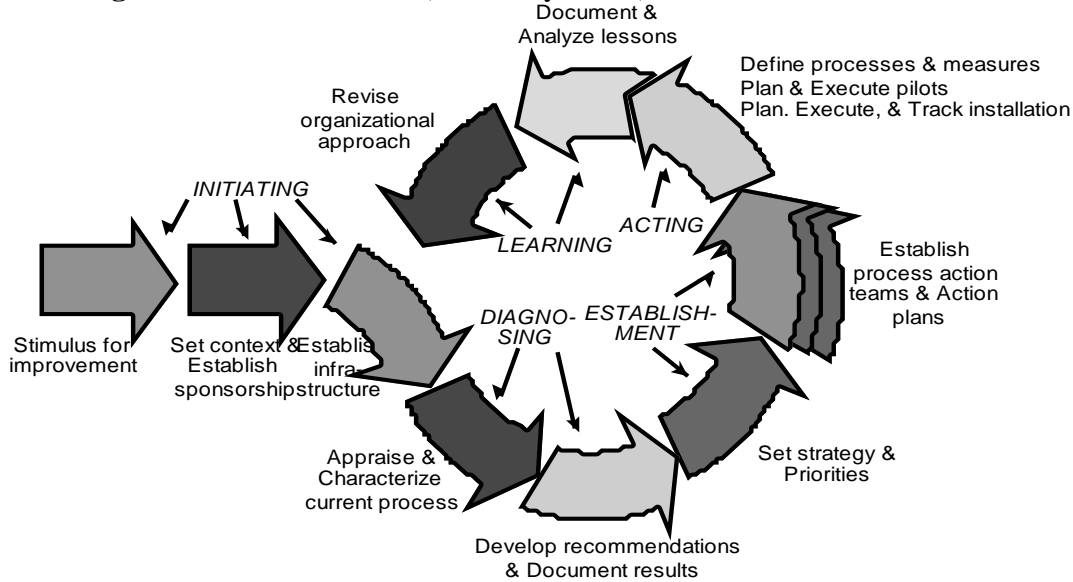
TelSoft emerged as an ideal site because the company was experiencing significant problems related to RE issues. For example, *TelSoft* depended on a few very large customers that constantly required software engineers to respond to requirements changes. Also, these customers had different requirements elicitation and documentation processes in place, and *TelSoft* was requested to adapt to each of these. Finally, the resulting software releases were often shipped with deviations from agreed upon requirements. *TelSoft* had previously been engaged in improving RE practices through a CMM-based initiative. While this effort resulted in documented new processes, these processes were not appropriate for the culture and business realities at *TelSoft*. Therefore, no sustainable changes had been implemented into RE practices.

Industry-research collaboration

To address these problems, a Collaborative Practice Research (Mathiassen, 2002) project was initiated between *TelSoft* and the authors. This research model focuses on understanding, supporting, and improving software practices; it relies on strong collaboration between practitioners and researchers; and, it seeks to develop relevant contributions based on rigorous research practices.

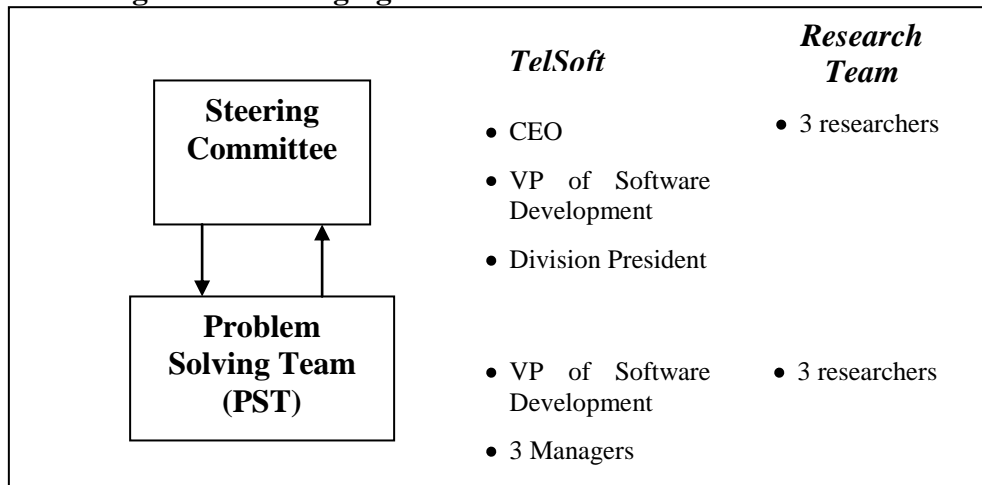
In seeking new approaches to problem solving in a business environment, Kock and Lau (2001) propose that action research is most appropriate. Specifically, we followed the recommendations of McKay & Marshall (2001) by implementing two interacting cycles of practical problem solving (leading to improvements at *TelSoft*) and research (leading to contributions to the literature). We implemented that by following the IDEAL model for improving software practices (McFeeley, 1996). This particular research article focus on information gathered during the “D” phase or “Diagnosing” (see Figure 9).

Figure 9: IDEAL Model (McFeeley, 1996)



The research project was managed by two teams: the Steering Committee and the Problem Solving Team. The Steering Committee was composed of the three researchers and *TelSoft*'s chief executive officer, division president, and vice president of software development. Meetings were held on a quarterly basis and used to set strategic direction for the improvement initiative. The Problem Solving Team (PST) was composed of the three researchers and *TelSoft*'s vice president of software development and three mid-level managers. The PST met monthly to manage operational aspects of the improvement initiative.

Figure 10: Managing Collaborative Practice Research



Assessment Experience and Results

From December 2004 to May 2005, we conducted an RE assessment at *TelSoft* using this combined approach. The effort involved 22 semi-structured interviews, two 3-hour workshops, a standardized assessment, and nearly a dozen meetings of the problem-solving and research

teams. In this section, we briefly describe how we collected, analyzed, and interpreted data on RE practices as well as arrived at key recommendations.

Step 1: Initiate assessment

The RE assessment was managed by the Problem Solving Team (PST). The goals of the assessment were to determine strengths and weaknesses of the existing RE practices and to identify improvement opportunities. Based upon these goals, the primary assessment strategy was perception-driven. The PST identified three stakeholder groups actively involved in creating and managing requirements: software development, internal customers, and external customers. Because the group valued the insights that could be achieved by comparing the company's processes against best practice, a process-driven component was also included in the assessment plan.

Step 2: Execute inquiry cycles

The resulting assessment plan contained three perception-driven inquiry cycles and one process-driven inquiry cycle. Key insights from each of these inquiry cycles are summarized in the following sections.

Inquiry Cycle 1: Software Development Perceptions

The software development group at *TelSoft* is responsible for interacting with clients to generate a software requirements specification, creating the GIS software based upon these software requirements, evaluating the impact of requirements changes, and ensuring the quality of the resulting software product. We interviewed nine representatives from the software development group: 2 project managers, 2 software engineers, 1 quality assurance analyst, 2 business analysts, and 2 mid-level managers. The interviews typically lasted one hour and were attended by at least two of the authors. The first author participated in all of the interviews, generated field notes, and maintained the case study database. An interview guide was created that asked about both objective and subjective data on requirements-related documentation and activities (see Table 14).

Table 14: Interview Guide

| <i>Requirements Documents</i> | <i>Requirements Activities</i> |
|---|---|
| <ul style="list-style-type: none"> • Which? • Inputs to you? • Contributions? • Output to whom? | <ul style="list-style-type: none"> • Which? • Interactions? • Collaboration? • Resources? |
| <ul style="list-style-type: none"> ✓ Strengths ✓ Weaknesses ✓ Opportunities | <ul style="list-style-type: none"> ✓ Strengths ✓ Weaknesses ✓ Opportunities |

Because this assessment was conducted as part of an improvement project, our analysis focused on the weaknesses identified. Participant's perceptions were analyzed for similar themes and documented into a list of 17 potential problem areas. Later, all members of the software development group participated in a three-hour workshop to evaluate this list. For each problem area, participants individually provided an assessment of criticality, feasibility, and priority.

These individual responses were then debated and again prioritized in break-out sessions during the workshop. A plenary session was then held in which all groups described their top issues.

Table 15 shows the RE-related problems that the software development group gave highest priority.

Table 15: Software Development Problem Areas

| <i>Problem Area</i> | <i>Description</i> |
|---|---|
| Quality Assurance Disintegration | Quality assurance department needs to be kept informed as detailed requirements evolve. |
| Change Management | Requirements changes are not addressed in a systematic fashion; documents are not kept updated and consistent. |
| Ad-hoc Review | Review of requirements is often performed in an ad-hoc fashion where reviewers are unprepared and critique is not systematically fed back into the requirements process. |
| Resource allocation | Quality assurance and core development have difficulties in prioritizing tasks and requests across projects. |
| Customer variation | There are considerable variations in requirements management and quality assurance practices across customers |
| Process Practice vs. | <i>TelSoft's</i> documented requirements management process is considerable different from actual practice; the ongoing maintenance and innovation of the described processes is not institutionalized. |
| Documentation Standards | Documentation standards vary; there are considerable variations in style and level of detail across authors; the most appropriate documentation form is not necessarily chosen to effectively target documentation users; some documentation standards do not fit current needs. |
| Outdated tools | Tools and methodologies for requirements management are not state-of-the-art; there are no procedures or responsibilities in place to facilitate improvements. |

Inquiry Cycle 2: Internal Customer Perceptions

In the second perception-driven cycle, we focused on the internal groups that interacted with the software development group in generating and managing software requirements. The software development group receives requirements from both the marketing organization and an internal production group that uses its GIS software. We interviewed 2 sales people, 3 project managers for the internal production group, and a mid-level manager. Once the interviews were completed, the authors again analyzed the interview data for common themes that suggested potential problem areas. We held a workshop for validating and prioritizing the 14 identified problem areas that involved the people interviewed as well as other users within the internal production group. Table 16 lists the RE-related problems given highest priority by internal customers.

Table 16: Internal Customer Problem Areas

| <i>Problem Area</i> | <i>Description</i> |
|--|--|
| Unsystematic early capture of requirements | <i>TelSoft's</i> Sales and Marketing representatives often capture client requirements in unsystematic, non-documented ways as basis for later interaction with other <i>TelSoft</i> stakeholders. |
| Changes not systematically communicated to internal users | Procedural and software changes are not systematically communicated to internal users |
| Varying contribution of requirements documentation | There are different opinions about the role and value of some requirements documentation. The intention is to create this document during the bid process to price the project. However, most clients spend little time specifying requirements upfront, and they tend to primarily present good, standard cases of data. That leads to inaccurate pricing. |
| Complex chain of requirements communication | There are several <i>TelSoft</i> stakeholders (e.g., Sales, Project management, business analysts, and software developers) involved in the requirements process. That leads to many interpretations and necessary translations, each introducing new sources of error. |

Inquiry Cycle 3: REGPG Assessment

Through these first two inquiry cycles, we learned of key concerns related to requirements practices from the perspective of *TelSoft* employees. However, we also wanted to evaluate *TelSoft's* practices against best practices to uncover additional vulnerabilities. The REGPG assessment (Sommerville et al., 1997) was chosen because prior empirical research showed it to be useful for RE process improvement (e.g., Kauppinen et al., 2002). Additionally, the authors had access to a REGPG assessment tool (Sommerville et al., 2005) that simplified data collection, provided process guidance, ensured accurate calculation of requirements maturity, and automated report generation.

The assessment was conducted during a two hour meeting with members of the PST. Participants were provided a written report containing a description of each of the 66 practices and expected benefits to including the practice. Early on, the group eliminated practices associated with the critical systems area as unnecessary for *TelSoft's* business. Each relevant practice was read aloud and categorized as being standardized, normalized, discretionary, or never followed. During discussion, the group created an additional category called "standardized but not checked" to indicate that *TelSoft's* documented processes met the spirit of the practice but there was no mechanism in place to ensure compliance. For the purposes of calculating RE maturity, this was coded as standardized in the REGPG assessment tool. For questions the group did not feel

prepared to answer, they solicited response from appropriate people after the meeting. After all of the practices had been evaluated, we assessed the usefulness of this assessment – what we learned, what possible actions could be taken, and how this compared to what we had discovered from the two workshops conducted. The REGPG assessment identified *TelSoft's* strengths as being in the areas of documenting, eliciting, and describing requirements. Areas for improvement were in analyzing, validating, and managing requirements. The company's overall RE maturity level was assessed at the lowest level: initial.

Table 17: Guideline Usage and Maturity Level

| | Basic | Intermediate | Advanced |
|--------------------|---------|--------------|----------|
| Guidelines Used | 19 | 9 | 0 |
| Weighted Score | 37 | 14 | 0 |
| Maximum Possible | 105 | 66 | 27 |
| Score % of Maximum | 35 | 21 | 0 |
| Level | Initial | | |

Inquiry Cycle 4: External Customer Perceptions

In the final inquiry cycle, we interviewed external customers who interacted with *TelSoft* to generate software requirements, request requirements changes, and perform user acceptance testing. The PST selected seven client representatives from three of *TelSoft's* long-time customers. A new interview guide was created that asked about requirements documentation, requirements management, and process innovation. In this cycle, there was no workshop used as a discussion forum. The customers praised the *TelSoft* personnel for understanding their business, responding promptly to customer requests, and adapting internal practices to client's needs; however, they identified areas for improvement as follows:

- *TelSoft* needs to increase the transparency and consistency of its configuration management, documentation, and test activities.
- *TelSoft* needs to improve its packaging procedures and related release notes.
- *TelSoft* needs to increase the frequency and consistency of their communication with the client.
- *TelSoft* should be better at making early estimates to help scope projects.

Step 3: Make Recommendations

An initial report was created by the PST and presented to the Steering Committee for approval. The problem areas from the combined RE assessment were categorized into seven improvement areas: software vision management, project portfolio management, software configuration management, customer relations management, requirements management, software quality assurance, and end-user interaction. The combined RE assessment revealed that *TelSoft* needed to develop its ability to sense customer needs, technological and market opportunities. They needed to be more proactive in their interactions with customers: sharing information about their

software development procedures to increase client confidence in the software product. Based upon this assessment, we recommended that TelSoft abandon a command-and-control approach and use governing principles and defined roles to become a more adaptive enterprise (Haeckel, 1995).

The improvement strategy would be addressed through a number of focused and dedicated project teams with clear success criteria and specified deliverables. The proposed project teams were to address software requirements management, software configuration management, software quality management, customer relations management, and software coordination issues. These project teams would be established, monitored, and coordinated through the PST. Once the Steering Committee approved the proposed project teams, a kick-off seminar would present the RE assessment results to all employees in the software development group to validate findings and create additional input from the employees on suitable improvement activities.

Discussion

This research contributes to our knowledge on how firms can assess RE practices to improve performance and better respond to customer and market dynamics. In the following sections, we discuss this contribution by relating the findings from *TelSoft* to the two research questions.

RQ1: Insights from Process- vs. Perception-driven

By comparing insights from the process-driven versus perception-driven inquiry cycles, we identified findings that were complementary, contradictory, or unique.

First, data from one inquiry type could support initial findings from the other. For example, the process-driven REGPG identified that *TelSoft* used only 2 of the 9 suggested practices in the requirements management area which could lead to development rework and systems that do not meet customer's expectations (Sommerville et al., 1997). The perception-driven assessment also identified weaknesses in managing requirements changes (*Cycle 1, Change Management*) and in ensuring that all stakeholders understand the current requirements and the relationship between them (*Cycle 2, Complex chain of requirements communication*). One of the REGPG guidelines advocates using a database to manage requirements, yet *TelSoft* suffered from unsophisticated requirements management tools (*Cycle 1, Outdated tools*).

Second, combining the two inquiry types could lead to contradictory results. *TelSoft* earned high marks with the process-driven REGPG for having defined a standard document structure with an optional glossary for specialized terms and a table of contents to help readers find information; the company also routinely held requirements review sessions. However, the perception-driven assessment indicated problems related to requirements documentation. For example, even though the format was standardized, it did not meet the needs of all stakeholders in the software development group (*Cycle 1, Documentation Standards*). Also, during the early requirements elicitation phases, sales and marketing representatives did not systematically document client requirements in sufficient detail for other stakeholders (*Cycle 2, Unsystematic early capture of requirements*).

Finally, one form of inquiry could provide insight into an area that the other did not even address. For example, the perception-driven inquiry highlighted problems in communicating requirements changes to stakeholders both internal and external to *TelSoft* (*Cycle 1, Quality*

Assurance Disintegration; Cycle 2, Changes not systematically communicated to internal users; Cycle 3, Increase communication with client). The perception-driven inquiry also revealed a lack of reflection and innovation of RE processes (*Cycle 1, Process vs. Practice; Cycle 1, Customer variation*) at *TelSoft* that was not captured during the REGPG assessment.

These examples illustrate the benefit of combining these two sources of knowledge to obtain a more comprehensive view of RE practices.

RQ2: Combined RE Assessment Approach

We have described a combined approach to RE assessment and illustrated its use in a case study at *TelSoft*, thereby addressing the second research question. The approach builds on existing process-driven assessments (Sommerville et al., 2005; Sommerville et al., 1997) and on approaches to organizational problem solving that is driven by stakeholder perception and involvement (Checkland et al., 1999). The resulting combined approach is illustrated in Figure 1.

In conclusion, this research illustrates how requirements assessment can benefit from the perceptions and active participation of key stakeholders as well as a process-driven approach such as REGPG. We advocate future research to explore how results from such a combined assessment can be used to improve RE practices within organizations.

References

1. Basili, V., and Rombach, H. (1988) The tame project: towards improvement-oriented software environments, *IEEE Transactions on Software Engineering*, 14, 6, 758-773.
2. Baskerville, R. (1998) Diversity in information systems action research methods, *European Journal of Information Systems*, 7, 2, 90.
3. Beecham, S., Hall, T., Britton, C., Cottee, M., and Rainer, A. (2005a) Using an expert panel to validate a requirement process improvement model, *The Journal of Systems and Software*, 76, 3, 251.
4. Beecham, S., Hall, T., and Rainer, A. (2005b) Defining a Requirements Process Improvement Model, *Software Quality Journal*, 13, 3, 247-279.
5. Checkland, P., and Scholes, J. (1999) *Soft systems methodology: a 30-year retrospective* John Wiley, Chichester.
6. CMMI Product Team (2002) *CMMI for Systems Engineering/Software Engineering/Integrated Product and Process Development/Supplier Sourcing*, CMU/SEI-2002-TR-011, Software Engineering Institute.
7. Curtis, B., and Paulk, M. (1993) Creating a software process improvement program, *Information and Software Technology*, 35, 6,7, 381.
8. 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, *Empirical Software Engineering*, 9, 1-2, 45-75.
9. Daneva, M. (2002) Using maturity assessments to understand the ERP requirements engineering process, *Requirements Engineering, 2002. Proceedings. IEEE Joint International Conference on*, 255-262.
10. Daneva, M. (2003) Lessons learnt from five years of experience in ERP requirements engineering, *Requirements Engineering Conference, 2003. Proceedings. 11th IEEE International*, 45-54.

11. 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, 541.
12. El Emam, K., and Madhavji, N.H. (1995) Measuring the success of requirements engineering processes, *IEEE International Symposium on Requirements Engineering*, 204-211.
13. Frederiksen, H.D., and Mathiassen, L. (2005) Information-Centric Assessment of Software Metrics Practices, *IEEE Transactions on Engineering Management*, 52, 3, 350-362.
14. Haeckel, S. (1995) Adaptive enterprise design: the sense-and-respond model, *Planning Review*, 23, 3, 6-13, 42.
15. Humphrey, W.S. (1989) *Managing the Software Process* Addison-Wesley, Boston, MA.
16. Kauppinen, M., Aaltio, T., and Kujala, S. (2002) Lessons Learned from applying the requirements engineering good practice guide for process improvement, *7th International Conference on Software Quality (ECSQ)*, Helsinki, Finland.
17. Kauppinen, M., Vartiainen, M., Kontio, J., Kujala, S., and Sulonen, R. (2004) Implementing requirements engineering processes throughout organizations: success factors and challenges, *Information and Software Technology*, 46, 14, 937.
18. Kock, N., and Lau, F. (2001) Information system action research: serving two demanding masters, *Information Technology & People*, 14, 1, 6-11.
19. Kotonya, G., and Sommerville, I. (1998) *Requirements engineering processes and techniques* John Wiley & Sons.
20. Lyytinen, K. (1988) Stakeholders, IS failures and soft system methodology: an assessment, *Journal of Applied Systems Analysis*, 15, 61-81.
21. Mathiassen, L. (2002) Collaborative practice research, *Information Technology & People*, 15, 4, 321-345.
22. McFeeley, B. (1996) IDEAL: A user's guide for software process improvement, CMU/SEI-96-HB-001, Software Engineering Institute.
23. McKay, J., and Marshall, P. (2001) The dual imperatives of action research, *Information Technology & People*, 14, 1, 46-59.
24. Mingers, J. (2001) Combining IS Research Methods: Towards a Pluralist Methodology, *Information Systems Research*, 12, 3, 240.
25. Mingers, J., and Gill, A. (1997) *Multimethodology: The theory and practice of combining management science methodologies* John Wiley & Sons, Chichester.
26. Niazi, M. (2005) An instrument for measuring the maturity of requirements engineering process, *6th International Conference on Product Focused Software Process Improvement*, Oulu, Finland.
27. Nielsen, P.A., and Pries-Heje, J. (2002) A Framework for selecting an assessment strategy, in L. Mathiassen, J. Pries-Heje and O. Ngwenyama (Eds.), *Improving software organizations: from principles to practice*, Addison-Wesley.
28. Orlikowski, W., and Baroudi, J.J. (1991) Studying information technology in organizations: research approaches and assumptions, *Information Systems Research*, 2, 1, 1-28.
29. Pouloudi, A., and Whitley, E. (1997) Stakeholder identification in inter-organizational systems: gaining insights for drug use management systems, *European Journal of Information Systems*, 6, 1, 1.
30. Rapoport, R. (1970) Three Dilemmas in Action Research, *Human Relations*, 23, 6, 499-513.
31. Schmidt, R., Lyytinen, K., Keil, M., and Cule, P. (2001) Identifying Software Project Risks: An International Delphi Study, *Journal of Management Information Systems*, 17, 4, 5-36.

32. Simon, J.-M. (1996) SPICE: Overview for software process improvement, *Journal of Systems Architecture*, 42, 8, 633.
33. Sommerville, I., and Ransom, J. (2005) An empirical study of industrial requirements engineering process assessment and improvement, *ACM Transactions on Software Engineering and Methodology*, 14, 1, 85-117.
34. Sommerville, I., and Sawyer, P. (1997) *Requirements Engineering: A Good Practice Guide* John Wiley & Sons, New York, NY.
35. Susman, G., and Evered, R. (1978) An assessment of the scientific merits of action research, *Administrative Science Quarterly*, 23, 4, 582-603.
36. Vidgen, R. (1997) Stakeholders, soft systems and technology: separation and mediation in the analysis of information system requirements, *Information Systems Journal*, 7, 1, 21-46.
37. Yin, R.K. (2003) *Case Study Research: Design and Methods*, (3rd ed.) Sage, Thousand Oaks.

Paper 2: Negotiating Repeat-ability and Response-ability

Title: Negotiating Repeat-ability and Response-ability in
Requirements Engineering

This paper is coauthored by Nannette Napier, Lars Mathiassen, and
Roy D. Johnson

This version of the paper was accepted and presented at the
International Conference on Information Systems (ICIS),
Award Best Paper in Track
Milwaukee, Wisconsin, 2006

Abstract

Requirements engineering (RE) practices are critical to success during development of business software. As managers assess RE practices, they apply specific perspectives that determine problems identified and recommendations for improvement. Two perspectives have recently dominated managerial thinking within the software industry, one rooted in software process improvement and the other rooted in agile software development. Underpinning these perspectives are two theories about what constitutes good software practice. In this paper, we explicate these theories in relation to RE and show how they differ in basic assumptions about the nature of requirements, requirements capture, requirements usage, change management, and approach to improvement. The repeat-ability theory holds that good requirements practices are plan-driven and follow generic best practices to arrive at an agreed-upon baseline of software requirements. Response-ability holds that good requirements practices are adaptive and involve close interaction between customers and developers to arrive at satisfactory software solutions. We use case study data from a software firm, TelSoft, to show how the theories lead to different interpretations about why current practices are problematic and how problems are resolved. Relating to the improvement strategy adopted at TelSoft, we demonstrate the superiority, for managers, of negotiating response-ability and repeat-ability concerns when improving RE practices. The paper concludes with a discussion of implications for research and practice.

Keywords

Requirements management, agile methods, software process improvement, CMM, case study

Introduction

Requirements Engineering (RE) involves eliciting, documenting, and maintaining software requirements throughout the software development lifecycle (Kotonya and Sommerville, 1998). Ineffective RE practices can have long-term consequences for software projects. For example, discovering requirements errors during the production phase is estimated to be 100 times more expensive to fix than if that same error is found during the analysis phase (Boehm, 1983). Acknowledging the significance of RE, software project managers have identified misunderstood requirements as the second most important risk to be managed (Schmidt, Lyytinen, Keil and Cule, 2001). Despite RE-specific process descriptions and best practices (Beecham, Hall and Rainer, 2005b; CMMI Product Team, 2002; Sommerville and Sawyer, 1997), RE remains one of the most challenging aspects of business software development (Beecham, Hall, Britton, Cottee and Rainer, 2005a). This is due in part to competitive business environments characterized by frequent requirements changes, rapid technological advances, and time-to-market pressures (Ramesh, Pries-Heje and Baskerville, 2002).

Software development managers looking to improve RE practices must first be able to identify problems with current RE practices and then determine the most appropriate tactics for resolving those problems. The perspective applied to the situation determines the problems identified and the resulting recommendations for improvement. Two perspectives which have strongly influenced software development are plan-driven versus agile development approaches (Boehm,

2002). Plan-driven approaches stress repeat-ability whereas agile approaches emphasize response-ability.

Plan-driven approaches, such as the Software Capability Maturity Model (SW-CMM), Bootstrap (Kuvaja and Bicego, 1994), or SPICE (Rout, 1995), emphasize documentation of project milestones, requirements, and designs; this approach is appropriate when the requirements are stable and known in advance (Boehm, 2002). The plan-driven approach assumes that improvement occurs by increasing organizational maturity through documented and repeatable processes (Humphrey, 1989). While some companies have benefited from implementing SW-CMM, there are also limitations with this approach to software process improvement: the scope of the assessment is limited by the model; it can be expensive to put into practice; and best practices may not fit closely the wants and needs of the organization (Iversen, Nielsen and Norbjerg, 2002). In the context of RE, one study found that SW-CMM-based approaches were able to improve technical RE problems, but not necessarily organizational RE problems (Beecham et al., 2005b).

Agile approaches, such as extreme programming (Beck, 1999), Crystal Methods (Cockburn, 2000), or Adaptive Software Development (Highsmith, 2000), emphasize people and prototypes over processes and documentation (Agile Alliance, 2001; Highsmith and Cockburn, 2001). Agile RE practices are less formal than plan-driven RE practices, but they still focus on understanding the customer's business requirements (Orr, 2004). Because requirements are expected to change, agile development occurs in short, iterative development cycles, and there is little attempt to predict future requirements. Agile methods also prescribe close collaboration between customers and the development organization to continually refine and prioritize requirements.

Although there are strong advocates of both the plan-driven and agile approaches, there have also been recent attempts to explore combining the two approaches. Boehm (2002) suggests that project characteristics such as developer skill set, customer availability, and requirements predictability be evaluated and used to pick the approach that best fits the situation. Furthermore, he suggests combining plan-driven and agile approaches for projects that have mixed characteristics. Some studies have examined how agile approaches can comply with the guidelines of the SW-CMM (Paulk, 2001) and its latest version the Capability Maturity Model Integration (CMMI) (Anderson, 2005; CMMI Product Team, 2002). Empirical case studies have also begun to appear that show how this combination can occur (Baker, 2005; Salo and Abrahamsson, 2005). However, the mixed messages about what approach to adopt can be a source of confusion for software managers. There is therefore a need to explicate the theoretical underpinning of the two approaches and to understand how they apply to RE practices.

Hence, we explore the repeat-ability and response-ability theories that underpin plan-driven and agile approaches, and we apply them to RE practices in a software firm, *TelSoft* (a pseudonym). We emphasize the two theories for RE from the viewpoint of their implications for action. The objective is to clarify the underlying assumptions of plan-driven and agile approaches in relation to RE and to explore what types of problems and recommendations each perspective reveals. To achieve this, we conducted a systematic assessment of RE practices in *TelSoft* and used the data to address the following research questions:

1. What assumptions distinguish repeat-ability from response-ability theories of RE?
2. How do repeat-ability and response-ability theories differ in assessing RE practice?

3. How do repeat-ability and response-ability theories apply to improving RE practice?

The argument is organized as follows: First, the repeat-ability and response-ability theories on RE are presented and contrasted in terms of their underlying assumptions. Next, background information is provided about *TelSoft* and the adopted research approach. Then, we evaluate the theories based on data from *TelSoft*. The paper concludes with recommendations for software managers and future research.

RE Theories

A manager trying to decide how to improve RE practices may hold one of two divergent theories about why current practices are problematic and how problems are resolved: repeat-ability and response-ability. Repeat-ability holds that good requirements practices are plan-driven and follow a set of generic best practices for how to arrive at an agreed-upon baseline of software requirements. Repeat-ability is an important principle within the SW-CMM (Paulk, Curtis, Chrissis and Weber, 1993). In fact, the first step in increasing organizational maturity involves moving from an initial level to a repeatable level by reducing variations in practices (Humphrey, 1989). In contrast, response-ability holds that good requirements practices are adaptive and involve close collaboration and interaction between customers and developers to help develop satisfactory software solutions. Response-ability is an important principle within agile development approaches (Beck, 1999; Boehm and Turner, 2004; Turk, France and Rumpel, 2005). In fact, one of the four basic principles of the Agile Manifesto is “Responding to change over following a plan” (Agile Alliance, 2001). Table 1 describes these two idealized perspectives in detail and explicates their underlying assumptions in the context of requirements engineering.

Table 1: Theories of RE – Underlying Assumptions

| <i>Assumption</i> | <i>Repeat-ability</i> | <i>Response-ability</i> |
|---------------------------|--|--|
| 1. Nature of requirements | <ul style="list-style-type: none"> ▪ Requirements represent software capabilities ▪ Requirements are explicated as texts in documents | <ul style="list-style-type: none"> ▪ Requirements are perceptions of software capabilities ▪ Requirements are tacitly embedded in social relationships |
| 2. Requirements capture | <ul style="list-style-type: none"> ▪ Requirements are derived through specification ▪ Interaction is formal | <ul style="list-style-type: none"> ▪ Requirements are discovered through negotiation ▪ Interaction is informal |
| 3. Requirements usage | <ul style="list-style-type: none"> ▪ Requirements are baselined and predate development ▪ Requirements are stored with traceability to source code | <ul style="list-style-type: none"> ▪ Requirements emerge through development ▪ Requirements are expressed through software solutions |
| 4. Change management | <ul style="list-style-type: none"> ▪ Requirements changes are exceptions and must be managed | <ul style="list-style-type: none"> ▪ Requirements changes are expected and must be embraced |
| 5. Improvement approach | <ul style="list-style-type: none"> ▪ The goal is to reduce process variance through best practices | <ul style="list-style-type: none"> ▪ The goal is to increase customer satisfaction through collaboration |

In the repeat-ability theory, requirements are textual representations of the desired software capabilities. Requirements knowledge is explicated as objects that are passed between requirements providers and requirements receivers. Requirements capture is a formal process that occurs before development work begins; it includes document review, discussion, and sign-off to indicate approval. Once sign-off has been obtained, a requirements baseline is established. Any changes to the requirements baseline must be documented and communicated to relevant stakeholders (Paulk et al., 1993). The role of quality assurance is to verify that the completed software matches the requirements specification. If RE practices are problematic, this approach looks for missing or inefficient processes. The overall improvement approach in the repeat-ability paradigm is to institute best practices and reduce process variance (Humphrey, 1989).

In the response-ability theory, requirements exist as shared understandings between stakeholders. Requirements knowledge is tacit, and the role of documentation is minimized. Customers play a critical role during software development as expressed in the principle “Customer collaboration over contract negotiation” (Agile Alliance, 2001). Customers provide immediate feedback on interim versions of the software and set priorities for the next iteration. Requirements capture happens informally as part of ongoing conversations with customers. This incremental approach allows requirements changes to be incorporated into the next version of the software. If RE practices are problematic, this approach looks for breakdowns in communication with customers or between developers. The overall improvement approach is to increase customer satisfaction by enhancing collaboration to quickly adapt to customer requests.

Research Methodology

A partnership between *TelSoft* and three researchers from a University Innovation Center (UIC) provided the basis for data collection. Overall, we adopted an action research approach (Baskerville, 1998; Rapoport, 1970; Susman and Evered, 1978) to diagnose RE practices, provide specific recommendations, and implement improvements. In this section, we provide background information about the research site and describe the research approach of this study in detail.

TelSoft

TelSoft was founded in 1971 with the mission to be the premier technical services firm in the telecommunications and utility industries. Approximately 50 people within *TelSoft*'s software development division work together to build and customize geographic information systems (GIS) software. *TelSoft*'s biggest strength is its people: experienced software engineers with deep knowledge of the GIS application, systems analysts with strong customer relationships, and managers willing to adapt quickly to customer requests. However, the company acknowledges recent issues with its RE practices. For example, internal stakeholders complain that insufficient information is collected during requirements elicitation, thereby delaying design and development activities. Increasingly, customers identify missing functionality during acceptance testing of the delivered software. Also, financial pressures require *TelSoft* to downsize its workforce, causing it to lose valuable customer and application expertise.

TelSoft's prior attempt at improvement was initiated in July 2000 guided by SW-CMM (Paulk et al., 1993). Despite high productivity rates and perceptions of progress, support for the SW-CMM initiative was withdrawn in August 2001 due primarily to financial pressures. *TelSoft* decided to

commit resource to imminent development rather than to process improvement. The most visible remains of the improvement effort were unused and out-dated process documentation combined with mistrust for rigorously following SW-CMM to improve RE practices.

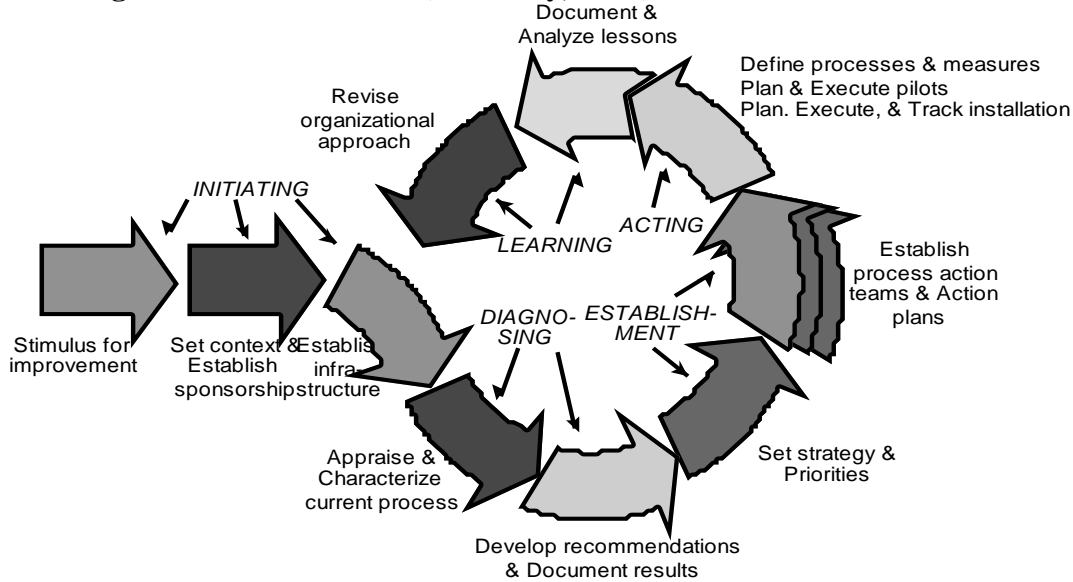
Industry-Research Collaboration

To address this problematic situation, a collaborative practice research (Mathiassen, 2002) project was initiated between *TelSoft* and the authors in October 2004. Collaborative practice research is a form of action research characterized by strong collaboration between practitioners and researcher. Galliers (1991) defines action research as an attempt to obtain practical results valued by the involved groups while adding to the body of knowledge in the discipline. Consistent with the dual problem solving cycle and research cycle (McKay and Marshall, 2001), the collaboration had two objectives: 1) improving the quality and productivity of software services at *TelSoft* through enhanced RE practices and 2) contributing to research in software requirements management. A memorandum of understanding detailing the project plan, initial tasks, and collaboration structure documented the agreement between *TelSoft* and UIC. The collaboration was designed to address the following tasks:

1. Model and assess *TelSoft's* existing practices and tools as they are applied to requirements elicitation, analysis, documentation, and management.
2. Describe key sources of requirements, the interests of the involved stakeholders, and the different ways in which new requirements are negotiated and used as the basis to define the scope of development projects.
3. Describe existing practices and tools used to continuously manage the scope of projects by tracing project activities and product functionality to the requirements of the project.
4. Identify strengths and weaknesses in current RE practices as well as opportunities for improvement. Generate new or changed process documentation to assist *TelSoft* future requirements management efforts.
5. Implement and assess selected improvements in RE practices.

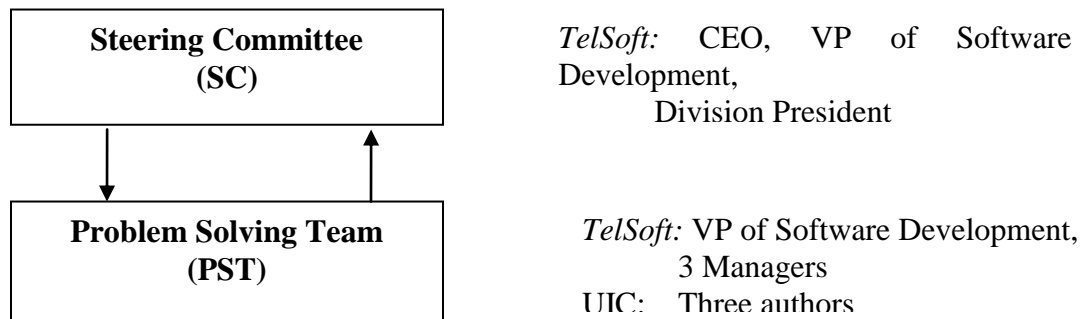
The IDEAL model was adopted from McFeeley (1996) to improve RE practices. This particular research article focus on information gathered during the “D” phase or “Diagnosing” (see Figure 1).

Figure 1: IDEAL Model (McFeeley, 1996)



The collaboration was managed by a Steering Committee (SC) composed of senior management from *TelSoft* and the three university researchers (see Figure 2). The SC meets 2-3 times per year as needed to oversee the project. More hands-on activities are completed by the Problem-Solving Team (PST) consisting of middle-level managers at *TelSoft* and the three researchers. The PST meets as needed to guide the collaboration and make decisions such as selecting participants for interviews and workshops.

Figure 2: Managing Collaborative Practice Research (Mathiassen 2002)



Data Collection

Data collection and documentation are essential for successful action research and qualitative research in general (Avison, Lau, Myers and Nielsen, 1999; Mason, 2002; Miles and Huberman, 1994). Because one of the authors had previously worked at *TelSoft*, the research team quickly earned acceptance by and confidence of the *TelSoft* employees. In December 2004, the research team initiated a diagnosis of RE practices by examining *TelSoft's* existing documentation of software development processes, procedures, and policies. This was followed by semi-structured interviews with 22 representatives from three major stakeholder groups: software development,

internal customers, and external customers (see Table 2: Summary of Interview Sources). In most cases, the interviews were recorded and conducted face-to-face with at least two researchers present; however, there were some interviews that were conducted via conference calls or with just the first author present. In all cases, the interviewers took extensive notes during the interview which were later reviewed, discussed, and analyzed. An interview guide was presented to participants to structure the interview process and ensure that we collected the desired information about RE practices. These interview guides were tailored to suit stakeholders internal and external to *TelSoft* (see Table 3: Interview Guide for Internal Stakeholders). Interviews were scheduled for one hour. While the interviews served as a primary data source, we used multiple sources of evidence to corroborate our findings (Mason, 2002; Miles et al., 1994). These sources included: field observation, field notes, minutes from PST meetings, the diagnostic report of RE practices at *TelSoft*, and unlimited access to all *TelSoft*'s process documentation.

Table 2: Summary of Interview Sources

| <i>Group Affiliation</i> | <i>Count</i> | <i>Role</i> |
|---|--------------|---|
| Internal Customers (Map Services, Sales) | 6 | 1 Liaison to Software Group 3 Project Managers 2 Sales Representatives |
| Software Development Group | 9 | 2 Development Managers 2 Project Managers 2 Software Engineers 2 Systems Analysts 1 Quality Assurance Analyst |
| External Customers (Far Telco, Local Telco, other) | 7 | 3 Managers, Far Telco 3 Managers, Local Telco 1 Engineer, other customer |

Table 3: Interview Guide for Internal Stakeholders

| <i>Requirements Documents</i> | <i>Requirements Activities</i> |
|--|---|
| <ul style="list-style-type: none"> • Which? • Inputs from whom? • Contributions? • Output to whom? | <ul style="list-style-type: none"> • Which? • Interactions? • Collaboration? • Resources? |
| <ul style="list-style-type: none"> • Strengths • Weaknesses • Opportunities | <ul style="list-style-type: none"> • Strengths • Weaknesses • Opportunities |

Data Analysis

As suggested by Miles and Huberman (1994, p. 56), data analysis was an ongoing process. After groups of interviews were conducted, the research team met to reflect upon what was learned and detect patterns emerging from the data. These ideas were discussed with the PST for feedback and verification and documented in field notes. Additionally, we created interim reports after completing interviews with each of the three stakeholder groups. We also conducted workshops with participants from the software development and internal customers groups to present the problems detected and to validate our assessment. In these 2-3 hour workshops, participants prioritized the identified problems in terms of criticality, feasibility, and priority. Feedback from these workshops and all interviews were accumulated into the comprehensive diagnostic report which was approved by both the PST and SC.

To answer our research questions, an additional level of analysis was conducted. We used an alternative templates strategy for analyzing the data (Langley, 1999); in this approach, different theories are independently applied to the same data to evaluate the explanatory power of the theories. This technique was previously used by Markus (1983) to compare three theories of resistance when studying systems implementation. Similarly, at *TelSoft*, we approached a complex managerial issue through alternative theoretical lenses of repeat-ability and response-ability. We applied each theory to the case data and assessed the usefulness of the theories for managerial practice.

The analytical process was guided by the fundamental principle of the hermeneutic circle (Klein and Myers, 1999); we alternated between focusing on each theory as a whole and on examining closely the underlying assumptions composing each theory as outlined in Table 1: Theories of RE – Underlying Assumptions. During the holistic analysis, the three researchers first adopted the repeat-ability lens. After reviewing selected data sources and reflecting upon their experiences at *TelSoft*, they identified key problems and recommendations that would occur within the repeat-ability paradigm. Once agreement had been reached, the three researchers then repeated their interpretation of the key problems and recommendations based upon the response-ability lens. This activity resulted in a rough, first version of what is presented in Table 4.

During the detailed analysis, evidence for each theoretical assumption was systematically gathered from the data. Several codes were developed for each of the five assumptions of repeat-ability and response-ability. For example, within the repeat-ability theory, two codes were created relating to the nature of requirements: (1) indicating that requirements are another representation of the software and (2) indicating requirements should be documented in textual format. Using Atlas.ti qualitative software, the first author then read through the entire set of data sources and applied the repeat-ability codes to all mentioning of problems related to requirements, their capture, their usage, change management, and approaches to improvement. The process was then done again using the codes from the response-ability theory.

Finally, all three researchers reconsidered the result of the holistic analysis in the light of the systematic coding of the data. This led to changes in and refinements of Table 4 and also to revision and improvement of the coding. These analysis activities were iterated until all three authors agreed that each of the two theories had contributed with a coherent and satisfactory explanation of the data from *TelSoft* (Langley, 1999).

Requirements Practices

TelSoft has two primary software products: Map Displayer and Engineering Support Tool (pseudonyms). The Map Displayer is relatively low-cost software that displays digitized maps, has global positioning capabilities, and supports limited drawing capabilities. Companies use Map Displayer to save on plotting and printing costs and to allow field workers access to up-to-date, accurate maps.

The Engineering Support Tool serves as an accounting system for utilities (e.g., location of poles, right of ways, cables, etc.). There is a great deal of configuration involved in setting up this particular software; therefore, it is expensive to license and to use. *TelSoft* has, as a consequence, only a handful of clients that use the Engineering Support Tool, and this client base is dominated by two long-standing, large customers whose requests largely dictate the product's innovation and growth.

There are two major groups within *TelSoft*: Software Development and the Map Services group. Software Development includes systems analysts, project managers, software engineers, quality assurance analysts, and their managers. Their job is to create new functionality requested by clients and maintain the existing software products. Map Services uses the Engineering Support Tool software to convert paper maps into digital format and to translate electronic maps from one format to another. Both of these groups communicate with *TelSoft*'s Sales group to learn about end user needs for either updated versions of the software or new formats for digitized maps.

In this next section, we describe RE practices at *TelSoft*. The data suggest that *TelSoft* practices vary greatly based upon the customer being served; therefore, this section is divided by customer type. First, we describe how Software Development and Map Services interact to generate requirements. Then, we describe the RE practices with two of *TelSoft*'s most established external customers. For each of these customers, we describe how requirements are captured, documented, stored, and changed.

Requirements Initiated by Internal Customers

The Map Services group is the primary internal customer of Software Development. Because this group is seen as part of the *TelSoft* family, the typical rules that apply to external customers regarding documenting and negotiating requirements are relaxed.

Requirements come from a variety of sources: end users looking for an easier way to do their jobs, Map Service's clients changing how digitizing should occur, or unanticipated data conditions found that the software now needs to handle. Requests for new software functionality are typically shared with Software Development via email messages or informal face-to-face conversations. Later, the resulting requirements are documented in bulleted format and logged in the defect tracking database. Because Map Services relies upon the software as a production tool, the chief concern of production managers is getting software that meets their requirements as quickly as possible with minimal documentation.

The relationship between the groups is strained in part because requirements are not fully understood and agreed upon before development work begins. Software Development gets frustrated and feels that Map Services does not do a good job of explicating their requirements

up front. Instead, they communicate what they think they want at a very high level and then, when software development implements it, they want something different. This leads to re-work and blown schedules.

From Map Services' perspective, Software Development does not deliver a quality product to them in a timely fashion which halts their ability to digitize maps and dramatically affects their bottom-line. Software Development prioritizes requests from external customers over the ones from internal customers. Not trusting that the stringent quality assurance guidelines were being followed, the Map Services manager dedicated a person on his staff just to test the quality of the work being done by Software Development. Because Software Development does not incur any costs for giving poor service or product to Map Services, there is little incentive for them to prioritize Map Services' needs over the needs of external customers.

Both Software Development and Map Services realize that there are missed opportunities for productivity and quality enhancement because the internal end users are not always aware of the capabilities of the Engineering Support Tool and Software Development is not knowledgeable about how the software is being used. This occurs even though there are a large number of end users from Map Services collocated with Software Development.

Requirements Initiated by External Customers

Software Development focuses primarily on two external customers that hold the largest number of licenses for its Map Displayer product and that have invested in enhancing the Engineering Support Tool. These companies drive changes to the software by specifying which functional and non-functional requirements they are willing to pay for and what the user-interface should look like. In an effort to keep these customers happy, *TelSoft* frequently responds with a "yes" when asked to make changes to their processes and products. Software Development has assigned a project manager to serve as the main customer liaison for each of these customers, Far Telco and Local Telco.

The project manager for Far Telco communicates with the customer primarily via email messages and internet-supported conference calls. Far Telco shares its high level needs and strategic direction with *TelSoft* at a yearly face-to-face planning session. More specific and detailed planning occurs for software releases which are scheduled approximately every 6-8 months. The client documents the business requirements for new functionality; then, communicates with the project manager to generate system level and functional requirements. These are documented formally in a functional specification that is written by *TelSoft* and must be approved before development work begins. The functional specification serves as the main communication means used by quality assurance analysts for testing and by software engineers for understanding what they should code. Once the code has been developed and integration tested, quality assurance analysts perform certification testing and document any deviations between the functional specification and the software product. If there are any changes to the requirements after the functional specification has been approved, a change control document is written to describe required change, perceived benefits, schedule impacts, and approval.

The project manager for Local Telco communicates with the client using a variety of means – email, phone, and face-to-face meetings – to understand requirements for new functionality.

Local Telco takes a much more hands-off approach to requirements elicitation. It emails high level requirements to *TelSoft* that includes bulleted lists or a few sentences; then, *TelSoft* interprets those into more detailed system level requirements and provides these through presentations or in documents for Local Telco's approval. Although *TelSoft* employees like having control over the changes that occur in the software, problems sometimes occur because Local Telco does not thoroughly review *TelSoft*'s specification of requirements. As a result, Local Telco is not always pleased with the delivered software.

Theoretical Interpretations

Given this background about the relationship between *TelSoft* and three of its primary customers, we now apply the repeat-ability and response-ability theories and compare and contrast the types of problems and recommendations each perspective brings to the data. For each theory, we revisit the data collected during assessment of RE practices at *TelSoft*, we interpret these data through the lens of each theory, and we present the result according to the five assumptions: nature of requirements, requirements capture, requirements usage, change management, and improvement approach.

Repeat-ability Perspective

Nature of Requirements

The repeat-ability theory assumes that requirements be explicated as texts in documents. At *TelSoft*, the existing requirements documents did not meet stakeholder needs. The software engineers commented that some sections of their technical requirements documents were no longer applicable. They also desired more detailed requirements documentation when working with Local Telco rather than relying on high-level documentation. They found the templates for the functional specification used for Far Telco to be sufficient, but there was great variation in the quality of this document depending on author:

“[Sometimes] we have somebody who’s writing the functional spec who doesn’t know the product and doesn’t know what kind of limitations we have because it is an existing product. When that knowledge isn’t there, it can make a product or a project more expensive, more complicated. There is a point also where they want to be able to do things that aren’t possible within the structure.” (*TelSoft* software engineer)

The Systems Analysts that write requirements documentation were also concerned that they had sufficient application knowledge:

“I have no access to the software for which I am writing requirements. Some I have never seen run. ... A major need is to have machine(s) set up and maintained ... so I can confirm current data structures and GUI. This should be dual use: for trouble report resolution, testing, documentation use; as well as for requirements. It should connect to realistic, preferably client provided, data sets which truly show their current models.” (*TelSoft* systems analyst)

Requirements Capture

The repeat-ability theory suggests formal interactions when capturing and approving requirements. Unfortunately, *TelSoft*'s Sales and Marketing representatives often capture client requirements in unsystematic, non-documented ways as the basis for later interaction with Software Development and Map Services. This leads to many interpretations and translations of customer requirements, each introducing potential new sources of error.

Requirements inspections can be a useful mechanism for clarifying ambiguous statements, documenting questions, and resolving issues. At *TelSoft*, review of requirements is often performed in ad-hoc fashion where reviewers are unprepared and the critique is not systematically fed back into the requirements process. The project manager for Local Telco expressed pressure to rush the requirements review and "hit the milestone dates regardless" because even a slip of a few days can upset the client. Several stakeholders noted that review meetings were ineffective when key experts had not read the proposed requirements documentation before the meeting. This can occur because of insufficient review time and overloaded human resources:

"If you have somebody who is working on three projects and has a deadline at the end of the week and somebody says 'I need you to review this functional spec in the next 48 hours', it doesn't happen. It just kind of falls through the cracks."
(*TelSoft* software engineer)

For some enhancements, requirements documentation is electronically distributed rather than discussed through face-to-face meetings. The quality of the comments received varies considerably indicating that this is not the most effective method for surfacing issues and building common understanding about requirements.

Requirements Usage

The repeat-ability theory stresses the value of establishing a requirements baseline before beginning development activities. Once approved by the customer, this requirements baseline serves as a contract between the customer and *TelSoft* regarding the capabilities of the delivered software:

"If the software is delivered and we missed a requirement the client can say 'Excuse me' (raps desk as if to point to a specific missed requirement). On the flip side, if client says 'Oh, but it doesn't do this.' We can say, 'Where does it say that?' " (*TelSoft* development manager)

Despite knowing the importance of an approved baseline, requirements sign-off at *TelSoft* happens inconsistently across customers and informally via email and phone conversations. In the interaction between Map Services and Software Development, obtaining of sign-off is not enforced. This causes problems when there are disagreements about delivered functionality.

The repeat-ability theory states that requirements should be stored with traceability to the source code. *TelSoft* experienced problems with both the repository chosen to store requirements and the ease of traceability. One software engineer expressed frustration with the current database used for storing requirements documentation:

“The problem with these technical documents is that once the project is done, nobody sees them again. They get lost in this huge Notes database so that all that time you spent on it ... is wasted. The document has no value anymore. If a bug gets called up on something, nobody knows where to go look for that documentation. If you do, it can take an inordinate amount of time to find it.”
(*TelSoft* software engineer)

Because the documents are difficult to find and not always kept up-to-date, software engineers rely on the code as the most credible source of requirements. The source code and requirements documentation can also get out of sync during the design process. *TelSoft*'s certification testing frequently detects discrepancies between the software and the requirements documentation. These discrepancies reflect design decisions that were discussed with the customer but not appropriately documented.

Change Management

In the repeat-ability theory, requirements changes are exceptions to the basic course of development and must be actively managed. Each requirements change must be documented with reference to the requirements baseline and communicated to all relevant stakeholders. *TelSoft* experienced problems in each of these areas.

Customer-initiated requirements changes are inconsistently documented. The project managers for external customers document changes on forms specified by the customer. These forms contain sufficient detail for *TelSoft* employees. With Map Services, change requests are usually described via phone call, face-to-face visit, or brief email. These discussions are then documented using a defect report.

Changes are not systematically communicated to key stakeholders, especially the quality assurance group. Rather than being told when changes occur, quality assurance analysts have to proactively check the requirements database for updates. This causes a delay in the quality assurance analysts' re-work of the associated test cases.

Improvement Approach

Within the repeat-ability paradigm, improvement focuses on reducing process variance by following best practices. Accordingly, processes should be defined; deviations from defined process should be minimized; and a mechanism for refining defined processes should be established.

TelSoft's current processes and templates do not explicitly support the management of requirements change. Also, the documented legacy processes are quite different from actual RE practices. Instead of repeating the same process over and over, *TelSoft*'s practices for documenting and changing requirements vary across customers. A common theme is that *TelSoft* allows external customers to dictate their internal processes. *TelSoft* resorts to ad-hoc practices when internal customers do not make those demands.

Finally, *TelSoft*'s RE practices are not assessed and continuously improved. For instance, there is no systematic process for tracking errors in requirements and software related to Map Services.

While software deficiencies are known, they are not tracked, root causes are not determined, and appropriate interventions are not enacted. There is also no mechanism for ongoing process management; therefore, documented RE processes are not evaluated with an eye toward innovation.

Response-ability Perspective

Nature of Requirements

In the response-ability theory, requirements exist as shared understandings between customers and software development. Since requirements are embedded in social relationships, tacit knowledge is lost when people with customer related capabilities and knowledge leave. At *TelSoft*, high employee turnover began to impact RE practices as senior-level employees voluntarily quit to pursue other opportunities. In fact, in the year since we completed our diagnosis, 7 of the 15 *TelSoft* employees interviewed are no longer with the company.

Requirements Capture

In the response-ability theory, requirements capture occurs informally and is seen as an ongoing communication with customers. Because requirements are discovered through negotiation, close, informal interactions with customers are essential during requirements capture. Here, we focus on specific problems with interactions during requirements discovery.

In the relationship between *TelSoft* and Far Telco, there are insufficient information technology tools in place to support requirements negotiation. For example, although the companies communicate frequently via conference calls, *TelSoft* does not have access to software that would support file sharing during these calls. Therefore, *TelSoft* is unable to see files created during the meeting that other participants were discussing. Also, Far Telco maintains its own database for storing high-level business requirements; however, *TelSoft* is not provided access to the most-up-to-date version of this database. Instead, Far Telco must manually push the requirements to *TelSoft*. These problems provide obstacles to requirements being effectively shared between *TelSoft* and Far Telco.

In the relationship between Local Telco and *TelSoft*, other communications obstacles are more salient. Local Telco does not trust *TelSoft* to deal with them fairly. Local Telco described *TelSoft* as “throwing code over the wall” without performing adequate testing. Because Local Telco doubted *TelSoft*’s integrity during requirements capture, one manager requested that *TelSoft* “roll back the covers” on processes, procedures, and tools.

TelSoft’s weakest relationship is with the users who actually work with their software products daily – even those that literally work around the corner from Software Development. *TelSoft* does not become involved with end users to identify and anticipate changes and to support training. This distant relationship means that *TelSoft* misses opportunities to understand customer needs for their products. For example, a manager at Far Telco described trying to manage and prioritize a list of 60 enhancement requests from the end user. She would appreciate more assistance from *TelSoft* in screening and prioritizing these potential requirements.

Requirements Usage

In the response-ability theory, requirements development is not done upfront and documented in requirements specifications. Requirements emerge throughout the development process. In this theory, spending too much time documenting requirements can be problematic:

“It’s always struck me that as much time as we spend writing these extremely detailed technical specifications, nailing down exactly how we’re going to do every single step of the implementation, that we’re basically stealing time from ourselves of actually getting the job done right in terms of testing it – integration testing and so on and so forth.” (*TelSoft* software engineer)

Key stakeholders also disagree about the value of other requirements documents. The Sales and Map Services groups use a specialized requirements template called the Source-to-Target Matrix for capturing requirements. The intention is to create this document during the bid process to price the project. However, most clients spent little time specifying requirements upfront, and they tend to primarily present their best case scenario and clean data sets. This leads to inaccurate estimates and pricing when the exceptions are encountered and dirty data sets are provided.

Change Management

In the response-ability theory, requirements changes are expected as a result of organizational dynamics and close collaboration and interaction between customers and developers. Requirements changes are therefore embraced as an important contribution to help develop satisfactory software solutions.

There is, however, a lot of formality built into the requirements change process, in particular in relation to Far Telco – in large part because Far Telco is a huge company having to integrate applications from several vendors. This level of formality causes problems for some Far Telco managers that would prefer to get changes quickly done without having to do the associated paperwork.

Improvement Approach

Within the response-ability paradigm, improvement focuses on increasing customer satisfaction through collaboration. *TelSoft*’s external customers feel that there is room for improving the amount of collaboration and the strength of the overall relationship. Local Telco representatives are the most dissatisfied with this relationship:

“We don’t have a partner relationship. A lot of times we kind of feel like there’s animosity from them toward us. I don’t know how big of a customer we are in their eyes, but I don’t feel treated like a valued customer.”(Local Telco manager)

Both customers desire more face-to-face time with *TelSoft*. Far Telco compares *TelSoft* with other vendors and notes that *TelSoft* lacks an onsite presence. They do not visit monthly, talk about future plans for the software, or provide ongoing training. This leaves *TelSoft* at a disadvantage when competitors use flashy sales presentations to impress upper management. There are even indications that Far Telco would be willing to fund some reasonable amount of travel to the site to have face-to-face interaction during RE.

Table 4: Problems and Recommendations

(#'s refer to assumptions in Table 1: Theories of RE – Underlying Assumptions)

| | <i>Repeat-ability</i> | <i>Response-ability</i> |
|----------|---|---|
| Problems | <ul style="list-style-type: none"> • Unsystematic early capture of requirements (1, 2) • Requirements documentation does not meet stakeholder needs (1, 3) • Requirements baselines not established and managed (2,3,4) • Requirements not systematically reviewed (3) • Requirements documentation not systematically updated (3, 4) • RE practices vary across customers (5) • RE process incompletely defined and different from practices (5) • RE practices not assessed and continuously improved (5) | <ul style="list-style-type: none"> • High dependency on people with customer related capabilities and knowledge (1, 2) • Customer sites are visited infrequently (1, 2) • Requirements and changes not effectively shared amongst stakeholders (1, 2, 3, 4) • Requirements documentation hinders interaction during development (2, 3, 4) • Lack of feedback from customers and quality assurance on software solutions (3, 5) • Lack of customer involvement in test (3, 5) • No systematic change management (4) • Lack of customer relationship management (5) |

| | <i>Repeat-ability</i> | <i>Response-ability</i> |
|-----------------|--|--|
| Recommendations | <ul style="list-style-type: none"> • Expand RE process to include systematic early capture of requirements • Revise requirements documentation standards so they meet the needs of all relevant stakeholders • Adopt two-phase funding to enforce establishment of requirements baseline • Develop systematic process for change management with traceability between requirements and source code • Enhance discipline of the requirements review process • Standardize, document, and enforce the RE process • Adopt continuous improvement mindset and establish systematic process management disciplines | <ul style="list-style-type: none"> • Increase availability and competence of people with customer related capabilities and knowledge • Establish activities to increase presence at customer sites • Establish ongoing communication of requirements amongst relevant stakeholders and make up-to-date documentation readily available • Document high-level requirements and establish systematic change management • Express detailed requirements directly as software solutions • Ensure systematic feedback from customers and quality assurance on interim software solutions • Improve test to reflect customer environments • Establish a customer relationship management program |

Recommendations for Action

The results of interpreting RE practices at *TelSoft* based on the two theories are summarized in Table 4. The table shows that both theories led to relevant, but quite different inventories of problems. The suggested recommendations for action are also quite different, though both inventories offer recommendations that potentially could improve RE practices. Because the theories provide potentially relevant, but different insights into RE at *TelSoft*, the question remains how to apply these recommendations to managerial decisions for improving RE practices at *TelSoft*. To explore this question, we consider how the actual assessment at *TelSoft* informed managerial decision-making on improving RE practices.

The comprehensive assessment report was created by the PST and presented to the SC for approval. The problem areas from the RE assessment were categorized into seven improvement areas: software vision management, project portfolio management, software configuration management, customer relations management, requirements management, software quality assurance, and end-user interaction. The PST found that *TelSoft* needed to better sense customer needs as well as technological and market opportunities. *TelSoft* also needed to be more proactive in its interactions with customers: sharing information about its software development procedures to increase client confidence in the software product. Finally, *TelSoft* needed to adopt a more disciplined approach to core activities related to RE. The PST hence recommended to the SC that *TelSoft* adopt an overall improvement strategy to become a more adaptive enterprise by

increasing its sense-and-respond capability (Haeckel, 1995; Haeckel, 1999). The improvement strategy should be implemented through a number of focused and dedicated projects with assigned resources, clear success criteria, and specified deliverables. The projects should be established, monitored, and coordinated through the PST. The SC approved the proposed improvement strategy, and a kick-off seminar was organized in which the RE assessment results and plans for improvement were presented to all employees in Software Development.

Management at *TelSoft* hence decided to adopt an improvement strategy that draws upon both theories. First, the strategy has a clear focus on enhanced interaction and collaboration between Software Development and internal and external customers; this is indicated by several improvement areas: customer relations management, requirements management, software quality assurance, and end-user interaction. *TelSoft* appreciated the importance of enhancing the relationships between software developers and internal and external customers, and on involving customers more actively in collaborative activities throughout the development process. Second, the improvement strategy has a clear emphasis on increasing discipline in key parts of the development process: software configuration management, requirements management, and quality assurance. In each of these areas, management at *TelSoft* saw a need to adopt more consistent processes and related tools. Finally, the strategy also focused on improving RE practices beyond the project level. All projects a *TelSoft* addressed issues related to the two primary software products: Map Displayer and Engineering Support Tool. Therefore, management found it important to improve coordination and consistency across projects.

In summary, the response-ability and the repeat-ability theory both provide important insights into problems and possible improvements of RE practices at *TelSoft*, and management's decision on a strategy for improvement draws upon both theories. The strategy is, however, not a simple merger of the two theories, but rather a negotiated compromise of the two theories for improvement. While *TelSoft* decided to improve the discipline in key RE activities, they had no desire to adopt statistical control and elaborate software metrics programs to help reduce variation across practices. Similarly, while *TelSoft* decided to improve the social relationships between developers and internal and external customers, they also insisted that it was important to have clear contractual arrangements with customers, to baseline requirements, and to systematically manage change request and the dynamics of their software configurations. Haeckel's approach to the adaptive enterprise (1995; 1999) was seen as an overall organizational approach that could help negotiate in detail such a compromise between the two theories.

Discussion

This research contributes to our knowledge of plan-driven versus agile approaches to software development in general and RE in particular by explicating the repeat-ability and response-ability theories and applying them to practices at *TelSoft*. Based on insights from the case, we argue that a negotiated compromise between the two theories provides the most useful approach to manage RE improvements. In this section, we elaborate on this contribution by relating the findings from *TelSoft* to the research questions and by discussing implications for research and practice.

Review of Research Questions

Our first research question focused on theory and asked about the key assumptions distinguishing repeat-ability and response-ability theories of RE. Drawing upon the literature on software process improvement and the literature on agile software development, we suggest that these theories differ based upon their assumptions about: nature of requirements, requirements capture, requirements usage, change management, and improvement approach. These findings are summarized in Table 4: Problems and Recommendations. There is an ongoing debate (e.g., Boehm, 2002; Boehm et al., 2004; Paulk, 2001) over the relationship between the two most influential contemporary paradigms for how to improve software practices, i.e. software process improvement and agile software development, and most issues remains unresolved. This is confusing and frustrating for managers who want to improve practices. The explication of the repeat-ability and response-ability theories provides clarification on main differences between the two paradigms, and it shows in particular how they apply to the key discipline of RE.

Our second research question focused on assessment and asked about differences in problem identification and resulting recommendation when diagnosing RE practices based on the two theories. Table 4 summarizes the findings from the two interpretations of RE practices at *TelSoft*. The two theories led to quite different inventories of problems and, as a consequence, also to quite different recommendations for improvement. In fact, there is little overlap between the two sets of findings. At the same time, both inventories of problems made sense to managers at *TelSoft*, and they were found to represent relevant and important issues related to RE practices. This application of the two theories suggests that they represent different and relevant perspectives on RE practices.

Our final research question focused on improvement and compared the resulting recommendations from applying the response-ability versus repeat-ability theories with the decisions made by management at *TelSoft*. Interestingly, management's chosen improvement strategy drew on insights from both theoretical perspectives and was tailored to the particular needs of *TelSoft*. When looking from Software Development towards internal and external customers, it was considered essential for the firm to maintain a highly responsive and flexible approach to deal proactively with both planned and emergent needs. The customers appreciated these practices, they saw them as expressions of a real interest in providing a high level of customer service, and they would like to enhance, rather than reduce these highly adaptive behaviors. Similarly, when looking at how developers, managers, and analysts worked within Software Development, it was quite clear, that practices were largely ad-hoc, established processes were not followed, and priorities were made and adjusted in-flight as a result of reactive responses to emerging demands. While there had been prior attempts to systematically follow SW-CMM (Paulk et al., 1993) to improve practices at *TelSoft*, these initiatives had failed. Also, while one project had experimented with agile software development, there were no systematic attempts or plans to adopt agile approaches. Instead, management decided to implement an improvement strategy which represented a negotiated compromise between the response-ability and repeat-ability theories, drawing upon the strengths of each without committing to extreme interpretations of either theory. This comparison between recommendations based on the two theories to the actual improvement strategy adopted at *TelSoft* suggests that the two theories represent complementary, rather than alternative perspectives on RE practices.

These responses to the three research questions are based on a particular approach to investigate RE practices at *TelSoft* with both strengths and limitations. Concerning reliability (Miles et al., 1994), we structured the investigation around three specific research questions, explicated our roles within *TelSoft*, explicated our theoretical constructs, used multiple sources of evidence, and used the fundamental principle of the hermeneutic circle (Klein et al., 1999) to converge towards a satisfactory interpretation. The reliability could, however, have been improved by instituting further checks of the coding scheme and its application. Concerning internal validity (Miles et al., 1994), we provided thick descriptions of the case and data, we linked data directly to the two presented theories and to each of the assumptions that characterize them, and we adopted systematic coding to relate the two theories to our data. The internal validity could be further improved by having key actors at *TelSoft* confirm the presentation and by considering rival explanations for how plan-driven and agile mindsets apply to the data from *TelSoft*. Finally, concerning action orientation (Miles et al., 1994), we present findings that are accessible to practitioners and researchers, the findings have proven useful to actors at *TelSoft*, and we have made the findings more useful for actors outside *TelSoft* by aggregating key viewpoints into two complementary theories of RE. The action orientation could be further improved by developing specific knowledge on how managers can negotiate an appropriate balance between repeat-ability and response-ability in other organizations.

Implications for Practice and Research

We began by considering a manager faced with problematic RE practices: what perspectives should this manager apply to assess current practices and make recommendations for improvement? Our research shows that applying either a repeat-ability or response-ability theory limits what a manager can know about RE practices. The two theories speak, to some extent, to different goals. For example, the response-ability theory emphasizes customer satisfaction whereas the repeat-ability focuses on reducing process variance. In most practical situations, neither of these goals can be ignored, and insights derived from the theories will therefore likely clash (e.g., role of documentation in RE practices) when managers prioritize how to actually improve RE practices. To get a more comprehensive understanding of RE situations in software firms, managers are therefore advised to apply both theories and negotiate how to best combine them to suit the particular context in which they operate.

Our research lends further support to efforts that seek to combine plan-driven and agile approaches (Boehm et al., 2004; Salo et al., 2005). The two theories explicate a common ground on which specific approaches can be evaluated, compared, and possibly combined with other approaches. Most attempts to compare and contrast the two paradigms do not apply theory as a basis for comparison or engage in theory-development to help us understand fundamental differences and identify new opportunities. While the literature on plan-driven development and process-focused improvement is clearly rooted in broader areas like Total Quality Management and statistical control, it is interesting to note that the agile software development literature does not explicitly draw upon theoretical insights on agility. The Agile Manifesto and related methods are largely an expression of a software-specific grassroots movement that resists traditional approaches to software development and emphasizes alternative values like: 1) individuals and interactions over processes and tools; 2) working software over comprehensive documentation; 3) customer collaboration over contract negotiation; and 4) responding to change over following a plan (Agile Alliance, 2001). Hence, we suggest that future research on combining plan-driven

and agile mindsets should apply theoretical lenses like repeat-ability and response-ability to investigate alternative approaches to business software development.

Such future research should build on the extensive literature on *organizational agility* (e.g., Dove, 2001; Gunneson, 1997; Haeckel, 1995; Haeckel, 1999) which is currently ignored by the software development discipline. Organizational agility requires “the ability to manage and apply knowledge effectively, so that an organization has the potential to thrive in a continuously changing and unpredictable business environment” (Dove, 2001, p. 9). Gunneson (1997) argues that agility is concerned with economies of scope, rather than economies of scale. The idea is to serve ever-smaller niche markets and individual customers without the high cost traditionally associated with customization. While the ability to respond to events in the environment in this way is the essential and distinguishing feature of the agile organization it is important to note that issues related to effective planning and appropriate process design are also emphasized (Dove, 2001; Haeckel, 1995; Haeckel, 1999); lean organizations are usually associated with the efficient use of resources, whereas agile organizations are related to effectively responding to a changing environment (e.g. through implementation of a response-ability theory) while at the same time being productive (e.g. through implementation of a repeat-ability theory).

As a case in point, the improvement of RE at *TelSoft* builds upon the principles of Haeckel’s adaptive enterprise design (1995; 1999). The intention is that such an approach will help create macro-level improvements within the organization as well as micro-level improvements within individual projects that can help *TelSoft* become more productive and respond more effectively to customers. Whether these attempts to improve RE practices will succeed remains to be seen. But they do set the stage for future research efforts that can help us develop alternative approaches to business software development. When market and technology conditions are relatively stable, one would expect an increased emphasis on repeat-ability on the macro-level and as these conditions change, one would expect increased emphasis on response-ability. Similarly, on the micro-level one would expect that the preference between the two theories would depend on the complexity and uncertainty of the development task at hand. The findings from this study could in this way guide future research efforts to investigate under which macro- and micro-level conditions different combinations of repeat-ability and response-ability would apply to development of business software.

Acknowledgements

This research was funded in part by *TelSoft*, Research Alliance, and a GAANN grant from the U.S. Department of Education. We thank the participants at *TelSoft* for their enthusiasm and openness during this ongoing industry-academia collaboration.

References

1. Agile Alliance "Manifesto for Agile Software Development", <http://www.agilemanifesto.org/>, 2001.
2. Anderson, D. "Stretching Agile to Fit CMMI Level 3 - the Story of Creating MSF for CMMI/Spl Reg/Process Improvement at Microsoft Corporation", in *Agile Conference*, July 24-29, 2005, pp. 193-201.
3. Avison, D., Lau, F., Myers, M., and Nielsen, P.A. "Action Research", *Communications of the ACM*, (42: 1), January 1999, pp. 94-97.

4. Baker, S. "Formalizing Agility: An Agile Organization's Journey toward CMMI Accreditation", in *Agile Conference*, July 24-29, 2005, pp. 185-192.
5. Baskerville, R. "Diversity in Information Systems Action Research Methods", *European Journal of Information Systems*, (7: 2), June 1998, pp. 90-107.
6. Beck, K. *Extreme Programming Explained: Embrace Change*, Addison-Wesley, Reading, MA, 1999.
7. Beecham, S., Hall, T., Britton, C., Cottee, M., and Rainer, A. "Using an Expert Panel to Validate a Requirement Process Improvement Model", *The Journal of Systems and Software*, (76: 3), June 2005a, pp. 251.
8. Beecham, S., Hall, T., and Rainer, A. "Defining a Requirements Process Improvement Model", *Software Quality Journal*, (13: 3), September 2005b, pp. 247-279.
9. Boehm, B.W. "The Economics of Software Maintenance", in *Proceedings of the Software Maintenance Workshop*, Washington, DC, 1983, pp. 9-37.
10. Boehm, B.W. "Get Ready for Agile Methods, with Care", *Computer*, (35: 1), January 2002, pp. 64-69.
11. Boehm, B.W., and Turner, R. *Balancing Agility and Discipline: A Guide for the Perplexed*, Addison-Wesley, Boston, 2004.
12. CMMI Product Team "CMMI for Systems Engineering/Software Engineering/Integrated Product and Process Development/Supplier Sourcing", CMU/SEI-2002-TR-011, Software Engineering Institute, 2002.
13. Cockburn, A. *Writing Effective Use Cases*, Addison-Wesley, Reading, MA, 2000.
14. Dove, R. *Response Ability: The Language, Structure, and Culture of the Agile Enterprise*, Wiley, New York, 2001.
15. Galliers, R. "Choosing Appropriate Information Systems Research Approaches: A Revised Taxonomy", in *Information Systems Research: Contemporary Approaches & Emergent Traditions*, H. Nissen, H. Klein and R. Hirschheim (Eds.), Elsevier, Amsterdam, The Netherlands, 1991.
16. Gunneson, A.O. *Transitioning to Agility – Creating the 21st Century Enterprise*, Addison-Wesley, Reading, MA, 1997.
17. Haeckel, S. "Adaptive Enterprise Design: The Sense-and-Respond Model", *Planning Review*, (23: 3)1995, pp. 6-13, 42.
18. Haeckel, S. *Adaptive Enterprise: Creating and Leading Sense-and-Respond Organizations*, Harvard Business School Press, Boston, MA, 1999.
19. Highsmith, J. *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*, Dorset House Publishing, New York, NY, 2000.
20. Highsmith, J., and Cockburn, A. "Agile Software Development: The Business of Innovation", *Computer*, (34: 9), September 2001, pp. 120-127.
21. Humphrey, W.S. *Managing the Software Process*, Addison-Wesley, Boston, MA, 1989.
22. Iversen, J., Nielsen, P.A., and Norbjerg, J. "Problem Diagnosis in SPI", in *Improving Software Organizations: From Principles to Practice*, L. Mathiassen, J. Pries-Heje and O. Ngwenyama (Eds.), Addison-Wesley, New York, 2002.
23. Klein, H., and Myers, M. "A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems", *MIS Quarterly*, (23: 1), March 1999, pp. 67-94.
24. Kotonya, G., and Sommerville, I. *Requirements Engineering Processes and Techniques*, John Wiley & Sons, 1998.

25. Kuvaja, P., and Bicego, A. "Bootstrap - a European Assessment Methodology", *Software Quality Journal*, (3: 3), September 1994, pp. 117-127.
26. Langley, A. "Strategies for Theorizing from Process Data", *Academy of Management Review*, (24: 4), October 1999, pp. 691-710.
27. Markus, M.L. "Power, Politics, and MIS Implementation", *Communications of the ACM*, (26: 6), June 1983, pp. 430-444.
28. Mason, J. *Qualitative Researching*, (2nd edition ed.), Sage, London, 2002.
29. Mathiassen, L. "Collaborative Practice Research", *Information Technology & People*, (15: 4)2002, pp. 321-345.
30. McFeeley, B. "Ideal: A User's Guide for Software Process Improvement", CMU/SEI-96-HB-001, Software Engineering Institute, Pittsburgh, PA, 1996.
31. McKay, J., and Marshall, P. "The Dual Imperatives of Action Research", *Information Technology & People*, (14: 1)2001, pp. 46-59.
32. Miles, M.B., and Huberman, A.M. *Qualitative Data Analysis: An Expanded Sourcebook*, Sage Publications, Thousand Oaks, 1994.
33. Orr, K. "Agile Requirements: Opportunity or Oxymoron?" *IEEE Software*, (21: 3), May-June 2004, pp. 71-73.
34. Paulk, M. "Extreme Programming from a CMM Perspective", *IEEE Software*, (18: 6), November-December 2001, pp. 19-26.
35. Paulk, M., Curtis, B., Chrissis, M.B., and Weber, C.V. "Capability Maturity Model for Software, Version 1.1", CMU/SEI-93-TR-24, Software Engineering Institute, Pittsburgh, PA, 1993.
36. Ramesh, B., Pries-Heje, J., and Baskerville, R. "Internet Software Engineering: A Different Class of Processes", in *Annals of Software Engineering*, Kluwer Academic Publishers, 2002, pp. 169-195.
37. Rapoport, R. "Three Dilemmas in Action Research", *Human Relations*, (23: 6)1970, pp. 499-513.
38. Rout, T.P. "Spice: A Framework for Software Process Assessment", *Software Process: Improvement and Practice*, (1: 1), August 1995, pp. 57-66.
39. Salo, O., and Abrahamsson, P. "Integrating Agile Software Development and Software Process Improvement: A Longitudinal Case Study", in *International Symposium on Empirical Software Engineering*, November 17, 2005, pp. 187-196.
40. Schmidt, R., Lyytinen, K., Keil, M., and Cule, P. "Identifying Software Project Risks: An International Delphi Study", *Journal of Management Information Systems*, (17: 4)2001, pp. 5-36.
41. Sommerville, I., and Sawyer, P. *Requirements Engineering: A Good Practice Guide*, John Wiley & Sons, New York, NY, 1997.
42. Susman, G., and Evered, R. "An Assessment of the Scientific Merits of Action Research", *Administrative Science Quarterly*, (23: 4)1978, pp. 582-603.
43. Turk, D., France, R., and Rumpe, B. "Assumptions Underlying Agile Software-Development Processes", *Journal of Database Management*, (16: 4), October-December 2005, pp. 62-87.

Paper 3: Managing Legacy and Current Processes

Title: Software Process Reengineering: A Model and Its Application
To an Industrial Case Study

This paper is coauthored by Nannette Napier, Jonathan Kim, and
Lars Mathiassen

This version of the paper was revised and submitted for
review at *Software Process: Improvement and Practice*

Abstract

Many software organizations engage in software process improvement (SPI), but software processes may not be fully implemented and process descriptions may become outdated. Moreover, some organizations suspend improvement efforts for a while before reengaging. As a result, SPI initiatives may need to reengineer legacy processes that are inconsistent with current software practices and policies. While the literature addresses how organizations can reengineer business processes and legacy systems, no guidance exists on reengineering software processes. Software Process Reengineering (SPR) is a transitional activity that helps organizations effectively reengage in SPI by defining criteria for making use of legacy processes; by assessing existing software processes against these criteria; by selecting processes to be removed, innovated, or created; and, by instituting a process management discipline to support continued improvement efforts. In this paper, we derive principles for SPR, use these principles to propose a model for reengineering software processes, and present an industrial case study to demonstrate the effectiveness of the model. In the presented case, SPR had several benefits: it leveraged earlier investments in legacy processes; it engaged key stakeholders in revitalizing improvement efforts; it created a shared understanding of the organization's software practices; and, it established a solid platform for continued SPI.

Keywords

Process implementation and change, reengineering, software management, software process

1. Introduction

Studies of software process improvement (SPI) have identified critical success factors such as continued commitment by management, involvement of respected technical staff, allocation of sufficient resources, and a clear statement of improvement goals [1], [2]. Barriers to SPI success can come from a number of sources including technical staff that consider it too time-consuming [2] and political pressures that focus more on obtaining a specific level than creating actual improvements [3]. As a consequence, many organizations struggle to advance in organizational maturity despite considerable investments in process-driven approaches. One study showed that 23% of organizations surveyed rated their SPI efforts as being marginally successful or not successful at all [4].

Process improvement models such as the Software Capability Maturity Model (SW-CMM) and CMMI present idealized scenarios of how organizations steadily advance in maturity through a series of lock-step phases [5], [6], [7]. By contrast, case studies of SPI reveal a slow process which may consist of active periods of progress and success interspersed with stagnating periods of disinterest and withdrawal of resources. In fact, one study found that after completing an initial SPI assessment, 42% of organizations soon diverted improvement resources to more pressing events and crises [8]. Given these shaky

beginnings, it is no wonder that moving from maturity level one to level two can take over two years [9]. When organizations reengage with SPI after having focused resources on other business issues, they do not begin with a clean slate; instead, they carry legacy software processes and associated documentation from previous SPI efforts. These processes may be inconsistent with current software practices and policies. This raises the question of how organizations can effectively manage legacy software processes as they reengage in SPI.

While the literature addresses how software organizations can manage legacy software systems [10], [11], [12], there is no guidance on how they can manage legacy processes. Similarly, there is advice for establishing a Software Engineering Process Group (SEPG) to manage software processes [13]; however, no specific direction is provided for transforming legacy processes as part of establishing a process management infrastructure. When organizations reengage in SPI, one choice would be to simply ignore legacy software processes and start creating new software processes. However, such an approach does not allow the organization to leverage the investments made in existing process capabilities, it requires that all processes are designed from scratch, and it easily reinforces general mistrust in the value of SPI. We propose an alternative solution which reuses knowledge embedded in legacy processes and institutionalizes a process management discipline as a platform for continued SPI efforts. This approach requires knowledge on how to reengineer software processes, including criteria for evaluating and selecting relevant processes, and practical ways to integrate legacy processes into new practices.

In this paper, we review related work to identify principles for software process reengineering (SPR). We then use these principles to propose a model for SPR that enables software organizations to reengage in SPI by leveraging previous investments in process capabilities. In Section 2, we define SPR in the context of SPI and, more specifically, software process management. In Section 3, we derive SPR principles based on existing knowledge on business process change [14], [15], [16] and reengineering of legacy systems [11], [17]. In Section 4, we then propose a model for SPR, define its individual elements, and detail the steps involved. In Section 5, we demonstrate the effectiveness of the model by presenting an industrial case study. Section 6 presents conclusions and future research directions.

2. Background

2.1 Software Process

A *software process* can be defined as “the coherent set of policies, organizational structures, technologies, procedures, and artifacts that are needed to conceive, develop, deploy, and maintain a software product” [18, p. 560]. In the SW-CMM and CMMI [6], [7], software processes are key to increasing organizational maturity: mature software organizations define processes and tailor them to specific projects; they establish an infrastructure for managing software processes; and they use quantitative measures to support continuous development of software processes [5], [6], [7]. Organizational maturity is indicated by satisfying key process areas associated with five levels: initial (1), repeatable (2), defined (3), managed (4), and optimizing (5).

In this paper, we focus on the documented software process descriptions, shortened to software that processes, which an organization creates during SPI. These documents software processes can take many forms including detailed, textual process descriptions, work flow diagrams, templates, standards, and guidelines. We further distinguish between legacy processes and managed processes. Legacy processes are software processes that have become out-dated because changes have not been carefully managed over time. The documented legacy processes have become inconsistent with the organization's current policies and practices. Like legacy systems, legacy processes often contain important business knowledge about successful operation of the software organization. When these processes are not carefully maintained, they can suffer many of the same problems as legacy systems: difficulty in modifying, out-of-date, and no longer useful [17], [19]. Just as we are learning it is important to evolve legacy systems over time, we must carefully consider why, when, and how to evolve legacy processes so they become aligned with continued SPI efforts [20], [21], [22].

By contrast, managed processes are software processes that have a well-defined state, represent current organizational policies, and are explicitly monitored and controlled. Managed processes are in line to be approved and implemented into engineering practices. Ideally, an organization would have a software process repository that contains only managed processes and no legacy processes. To ensure that software processes are defined, documented, measured and controlled [23], [24], organizations need to practice software process management.

2.2 Software Process Management

SW-CMM and CMMI [6], [7] both provide guidelines for instituting a process management discipline. The SW-CMM proposes five process areas related to software process management: organization process focus (level three), organization process definition (level three), training program (level three), quantitative process management (level four), and process change management (level five). CMMI [7] describes similar process areas related to process management and offers both a continuous and a staged view for approaching SPI. The staged view prescribes an order that organizations should follow for SPI which is consistent with SW-CMM. The continuous view encourages organizations to customize their focus on process areas based on their current weaknesses, overall strategy, and SPI goals.

Two of these process areas are particularly important for SPR: process definition and process change management. Process definition (level 3) advises organizations to develop, maintain, and explicitly manage standard process assets such as policies, procedures, templates, or standards [7]. Process definition improves visibility into engineering and management practices for all stakeholders and is a prerequisite to process automation and quantitative process management [24]. Process change management (level 5) emphasizes continuously improving processes used within the organization to increase quality and productivity [5]. These two process areas work together to prevent legacy processes and ensure that managed processes exist. This implies that organizations that have reached

level three are less likely to have legacy processes because the documented standards are actively maintained and quality assurance verifies that organizational performance is in line with standards. Those organizations that have reached level five have the added protection of being good at process change management. As they discover better ways of doing executing a process, they have procedures in place to fold those innovations into standard processes. This discipline keeps standards up-to-date with the company's local best practices.

If organizations adopt SW-CMM or the staged view with CMMI, they will develop a portfolio of software processes (e.g. organizational process definition) before they have completely instituted proper process management (e.g. process change management). Hence, they risk creating a situation that allows legacy processes to accumulate. If organizations adopt a continuous view, they can choose to institute a process management discipline at an earlier stage, thereby reducing this risk. There is, however, no awareness in the literature that such a risk exists and should be addressed. Therefore, it remains to be seen how widespread such practices will become.

2.3 Software Process Reengineering

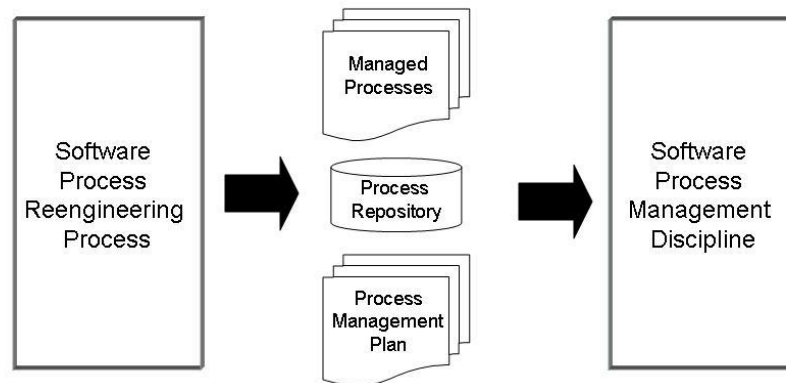
The term SPR has previously been associated with defining processes to reengineer software:

“Software process reengineering should result in a self-improving software process for updating and renewing software on an ongoing basis... The reengineered software process should include the activities involved in creating, selecting, and integrating reusable software components into new applications” [10, p. 72-73]

We agree with Ahrens et al. [10] that developing effective processes for reusing and reconfiguring software components is an important research area. However, SPR involves more than just software components used to build applications. It also involves people, management strategies, and organizational infrastructures. In general, reengineering involves the systematic analysis and modification of a system to allow transforming it into a new format [25]. For example, business process reengineering (BPR) transforms organizations into new forms by reconfiguring people, technology, and processes in a more rational way to better support business strategies and objectives [26].

Similarly, for organizations reengaging in SPI, SPR transforms their legacy processes into managed processes and institutes a process management discipline. Organizations with a substantial portfolio of legacy processes must attend to these legacy processes through SPR before engaging in continued SPI efforts. SPR is hence a one-time activity to get SPI back on track by transforming legacy processes to managed processes, by generating a process repository, and by developing a process management plan. These resulting deliverables subsequently become the foundation for moving SPI forward based on a strong process management discipline as shown in Figure 1. In this paper, we describe a model for conducting SPR to generate the three deliverables, and we demonstrate how this

Figure 1: Relationship between Software Process Reengineering (SPR) and Software Process Management



model was used at *TelSoft* to establish a software process management discipline and bring the organization's SPI efforts back on track.

In summary, we define SPR as follows:

SPR defines criteria for transforming legacy processes; assesses existing software processes against these criteria; and selects which processes should be removed, innovated, or implemented. SPR establishes on that basis a repository of managed software processes and institutes a process management discipline to support continued improvement efforts.

In the context of mature software organizations, the need for SPR is not apparent: organizations that reach level five would not develop a backlog of legacy processes and need to engage in SPR. However, less than 45% of organizations investing in SPI report even reaching level two [27]. As organizations struggle to find effective paths towards increased maturity, they may linger between levels without successfully advancing to level three. As a result, these organizations will start accumulating legacy processes that increasingly become misaligned with current practices and policies.

This was the experience at *TelSoft*, a US based provider of software solutions for the telecommunication industry. *TelSoft* restarted SPI after a three year hiatus. They had previously developed several processes with extensive documentation. These processes had, however, not been maintained so they were no longer consistent with current software practices and policies. Faced with these legacy processes, we engaged in developing principles and a model for SPR and applied them to reengineer legacy software processes at *TelSoft*.

3. Principles for Software Process Reengineering

Fuggetta [18] states that "software processes are processes too", reminding SPI practitioners and researchers to learn from other communities concerned with managing

processes. Following this suggestion, we consult the broader literature on business process change to derive principles for SPR. Moreover, reengineering principles have been applied with success to evolve legacy software systems. Therefore, we also review this literature to inform our approach to SPR.

3.1 Business Process Change

In general, business process change [14] refers to strategic initiatives designed to improve organizational performance and product quality by redesigning and innovating business processes. Business process change initiatives such as BPR [28], [29], process innovation [30], and business process management [31], [32] are based on total quality management (TQM) [33], [34]. TQM has been proposed in the context of system development [35]. From TQM perspective, process management is one of the key components for quality-oriented organizational system [36]. Within the software engineering industry, SPI has been a dominant form of business process change. SPI can be affected by management infrastructure factors such as support of top management support and participation of stakeholders which are proposed from TQM literature [37]. SPR represents another form of business process change designed specifically to bring legacy process under process management control.

Consider Organizational Context

Business process change begins when senior management articulates a new vision for operations as well as an approach for transforming business processes [14], [38]; such guidance is frequently represented in vision statements, goals, and policies. When considering which actions will lead to the desired state, management cannot simply generically apply industry best practices to the situation. Instead, the change initiative must consider important elements of the business environment such as the organizational culture, existing policies, industry regulations, and norms and values [14].

Internal and external stakeholders serve as primary, first-hand sources for understanding the business environment, and they have a rich base of knowledge for action planning [39]. Therefore, the organization should leverage stakeholder knowledge about established work practices as well as possible process revisions and designs. An added benefit of having internal stakeholders involved is that it is likely to breed enthusiasm about the change initiative and counter any cynicism that could negatively impact change efforts [2], [40]. Involving the organization's external stakeholders can help enhance customer satisfaction during business process change [14], [41]. Therefore elicitation of various viewpoints on processes from diverse users is needed to create merged, consistent process models [42].

As business processes frequently cross organizational boundaries, it is also important to consider inter-organizational relationships when redesigning business processes [14]; for example, business partners need to be made aware of and agree to changes to process interfaces that will impact their work practices. Curtis et al. [43] further emphasize that making key processes visible improves coordination.

This way of situating change initiatives in the organizational context has already been recognized in the software engineering community through, for example, the Goal-Question-Metric (GQM) approach to measurement; GQM requires managers to tailor goals to the organizational context in question [44]. These insights suggest that SPR should be guided by the following principle:

Principle 1: SPR should consider the organizational context by identifying goals and policies for SPI and incorporating viewpoints of internal and external stakeholders.

Consider Change Practices

Before changing existing processes, initiatives should thoroughly assess current change practices [30], [38]. Generally, the organization's history with change initiatives indicates its ability to handle future initiatives. By definition, organizations engaged in SPR have experienced prior difficulties with SPI and have immature process management discipline. Organizations engaging in SPR should therefore learn from previous failures and be prepared to adopt new approaches to address the risk of failing again. Data should be gathered about successes and failures in previous change initiatives, strengths and weaknesses in current process documentation, and, about potential process revisions and redesigns. Organizations are advised to document these data so they can be shared across the organization and support action planning [45]. When considering which business processes should be modified, managers should challenge existing assumptions and practices [14], [30]. Use of consultants or change agents from outside of the organization can facilitate that process.

These insights suggest the following SPR principle:

Principle 2: SPR should consider the organization's change practices by critically reviewing previous SPI initiatives and results and by taking measures to avoid previous failures.

Leverage IT

Early BPR proponents perceived IT as an enabler of the innovative redesign of core business processes [30], [46]. Investments in IT infrastructure can facilitate relatively quick changes to business processes while outdated or inflexible IT infrastructures can constrain or inhibit process change [30]. An appropriate level of IT infrastructure is hence needed [47]. First, IT can enable process innovations by providing new capabilities for collecting, storing, and sharing data relevant for process execution [14]. Second, IT can facilitate the creation, sharing, and communication of process knowledge. For example, much research has been conducted around alternative software process models and tools to support improvement efforts [24], [48], [49].

Although IT itself does not have power to change processes, using it for the management of process knowledge is critical to the success of SPR. Process knowledge needs to be shared and communicated on a platform that provides easy access and management of software processes. Tools like groupware and web portals can reduce the cost of analysis

and enable many different stakeholders to participate in SPR [50]. The same approach has been applied to make reusable software components such as web services available at portals [51]. IT can in this way support a more collaborative approach to SPR and help make change happen. As organizations engage in SPR, they should therefore consider how to establish and leverage an appropriate IT infrastructure:

Principle 3: SPR should leverage IT capabilities to establish a platform for effective storage, communication, and usage of managed processes.

3.2 Legacy Systems Reengineering

Reengineering principles have already been adopted within software organizations in relation to legacy systems. Legacy systems are aging business software that are increasingly difficult to modify and evolve [17]. Legacy systems are often critical to businesses and contain embedded requirements and business knowledge. Such systems are problematic because they are difficult to evolve, are expensive to maintain, and use obsolete technology [19]. Software organizations face similar challenges related to legacy processes, and we therefore consider how principles for dealing with legacy systems apply to SPR.

Apply Multiple Strategies

Multiple strategies are offered when reengineering legacy systems. First, the *redevelopment* strategy, also called Big Bang or Cold Turkey [17], advocates complete replacement of the legacy system, very much in line with BPR advocates. This strategy is most appropriate when the business environment requires significant changes from existing systems; however, this approach is resource intensive and does not reuse existing knowledge. In addition, in a rapidly changing environment, the new system can become obsolete before development is completed. Second, the *migration* strategy moves an existing system to a new platform while retaining key functionalities of the legacy system and causing as little disruption to the operational and business environment as possible [11]. Finally, the *wrapping* strategy ensures reusability of existing code by refactoring legacy systems into modularized components with a well-defined interface [52]. Wrapping is considered a practical solution as it involves the lowest costs and the fewest risks. However, compared with redevelopment and migration, the wrapping strategy also has a minimal impact on improving legacy systems.

These insights suggest that a contingency approach should be taken when selecting the best strategy for reengineering legacy systems; that is, different approaches are appropriate based upon the business context and the specific legacy system under consideration. When organizations engage in SPR, they should therefore consider a wide range of options. In one extreme, radical approaches to SPR discard legacy processes and replace them with new ones; in the other extreme, incremental approaches identify and implement improvements in existing processes. Accordingly, legacy processes that are no longer considered useful with regard to current engineering practices and conditions may be redeveloped [17]; legacy processes that are potentially useful, but require reconfiguration and change may be migrated [11], [19]; and, legacy processes that have potentially useful

components or that are currently inappropriately documented may be wrapped [52]. Hence the following principle for SPR:

Principle 4: SPR should rely on multiple strategies that are contingently applied based upon current process portfolios and engineering practices.

Adopt Iteration

Another theme that emerges from reengineering of legacy systems is iteration. Engineering is generally an iterative rather than a purely linear process [53]; iterative approaches have been actively promoted within the software engineering discipline for more than three decades [54], [55]; and several SPI approaches such as the IDEAL model recognize the iterative nature of SPI (e.g., plan-do-act-check) [38], [56], [57]. Iteration allows learning to take place which can feed changes in later development cycles [54], [55]. In addition, iterative SPR practices can reduce resistance from employees and help process engineers better learn the targeted processes [56].

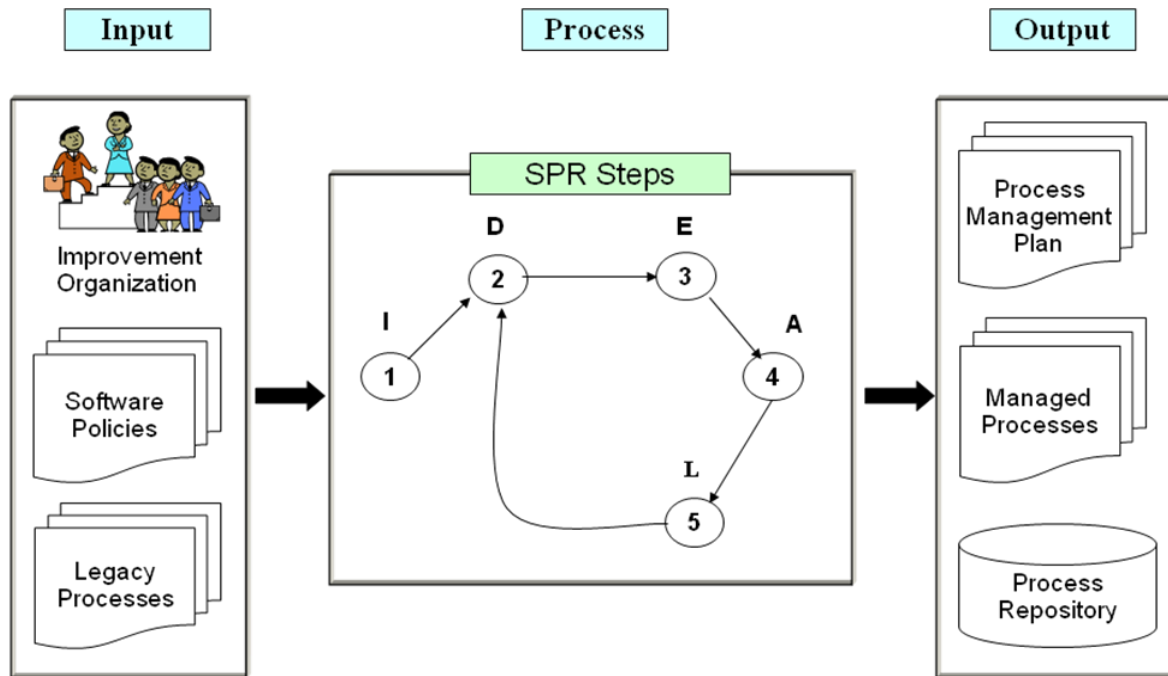
Bianchi et al. [58] explicitly describe an iterative reengineering strategy in which engineers select a small number of legacy components and apply iterative reengineering processes to these components. Engineers subsequently repeat this reengineering approach to other sets of components. The goal is hence to improve the quality of software systems continuously while guaranteeing coexistence among the various components. In the same vein, SPR should iteratively select a subset of legacy processes, assess the usefulness of those processes, and decide upon an appropriate plan for action. This process should continue until all legacy processes have been discarded or transformed into managed processes.

Principle 5: SPR should iteratively turn legacy processes into managed processes to enhance learning, facilitate change, and establish a baseline for continued SPI.

4. SPR Model

We have applied the five principles above to construct a model for conducting SPR (see Fig. 2 Figure 2). The SPR model takes three inputs: an improvement organization, software policies, and legacy processes. The iterative SPR steps subsequently produce three deliverables or outputs: managed processes, process repository, and process management plan. Each of these components is briefly discussed below.

Figure 2: Software Process Reengineering (SPR) Model



SPR Components

Improvement Organization

There is no single, best way to organize SPR; the most feasible organization depends on how improvement is generally organized within the organization and on the specific portfolio and status of legacy processes (SPR Principle 2). The general improvement organization provides leadership and context for SPR. Effective organizational structures discussed in the SPI literature, such as the SEPG [13], process action teams [7], and the experience factory [59], can help organize SPR as part of SPI. It is important to remember success in any SPI effort requires continued commitment by management, allocation of sufficient resources, and a clear statement of improvement goals [2]. Also, it is advisable SPR efforts involve well-respected software engineers and managers that give legitimacy to the project [2] and can express viewpoints from various stakeholder groups (SPR Principle 1). There will likely be several teams established as part of SPR, each with different and complementary responsibilities, e.g., to take stock of legacy processes (SPR Principle 2), to assess current process management practices, or, to design and implement a new process repository (SPR Principle 3).

Software Policies

In general, policies are guiding principles identified by senior management to guide decision-making and drive day-to-day operations [7]. In particular, software policies explicate the organization’s governing principles for successful software development. Using governing principles rather than a command and control paradigm helps an

organization to become more effective and adaptive in the dynamic contexts that characterize the software industry [60], [61]. Employees are provided with key principles that are enforceable and that state what to do and what not to do, without unnecessary details about how to do it. This empowers them to respond effectively and quickly in the best interest of the organization [60], [61], [62]. Before beginning SPR, the organization's current software policies should therefore be explicated to define the basis for SPR and to establish a strong foundation for continued SPI efforts (SPR Principle 1). We later show how software policies can be used to prioritize which legacy processes to discard now, revise immediately, or redesign later.

Legacy Processes

Legacy processes are documented software processes that exist at the beginning of SPR. Legacy processes should be critically examined to determine their status (SPR Principle 2). We specifically recommend that SPR systematically characterize all legacy processes on the basis of two key characteristics: documentation status and implementation status (SPR Principle 4). Documentation status indicates how well the process is described to support software practice and comply with standards for process documentation. Implementation status indicates the extent to which the organization's day-to-day practices align with the process. The combination of documentation status and implementation status is used to guide prioritizing activities during SPR.

SPR Steps

The SPR process requires coordinated efforts of many people within the organization. Legacy processes are transformed to managed processes by iteratively characterizing and modifying their status (SPR Principle 5). SPR considers a range of approaches when turning legacy processes into managed processes (SPR Principle 4). Each step provides additional clarity on opportunities and challenges related to bringing all legacy processes under management control. The steps are based upon the generic process improvement model IDEAL [38] (see section B for details).

Managed Processes

Managed processes are software processes under management control: they have been assigned a non-obsolete documentation and implementation status; they are available from the process repository; and they are addressed through continuous process management. Periodically, the current implementation status of each managed process is evaluated against a desired level of implementation. The documentation status is also reassessed to determine if changes are needed. Any changes to managed processes must follow the improvement organization's defined policy for change management.

Process Repository

The process repository is an IT-based resource that facilitates effective storage, communication, and usage of all managed processes (SPR Principle 3). The technological platform used could include company website, intranet, and internal documentation management system. It serves as an effective communication medium for key stakeholders regarding relevant software processes (SPR Principle 1). For example, a software process

describing the software testing procedures would be of interest to both internal and external stakeholders; however, the details of tracking software defects would be important only for internal stakeholders. The repository should allow each stakeholder group easy access to relevant software processes (SPR Principle 3).

Process Management Plan

The process management plan describes the activities and mechanisms that the improvement organization will adopt for continuous software process management after SPR completion. The process management plan should be based on a realistic and sustainable approach for implementing a process management discipline (SPR Principles 2 and 5). For example, a process management group could be established and given responsibility for activities such as approving software processes, monitoring compliance with approved software processes, deciding whether new processes should be created, deciding on the standards for process descriptions, and prioritizing work done on innovating and improving software process management. The process management plan should also describe how to maintain an up-to-date and easy-to-access process repository (SPR Principle 3) and be sensitive to the needs of both internal and external stakeholders (SPR Principle 1).

4.2 SPR Steps

In this section, we specify steps for conducting SPR using the IDEAL model [38] as framework. The IDEAL model consists of five generic steps used for implementing SPI: Initiating the project, Diagnosing current practice, Establishing an action plan, Acting out that plan, and Learning from these actions. These steps are served to create and embed relevant knowledge for SPI (Ravichandran and Rai).

Initiating

The objectives of the initiating phase are to understand the need for SPR, determine readiness to proceed, and create an overall plan and supportive infrastructure for the project [38]. Specifically, we advise the following activities:

I.1 Assess need for SPR. Organizations that previously invested in documenting software processes through SPI are candidates for SPR; however, not all companies that have started and stopped SPI will find it beneficial or cost-effective to engage in SPR. After all, the problems driving SPR – large body of legacy processes, lack of software process management discipline, and inadequate process repository – could also be solved by using an approach that starts over from scratch. SPR is appropriate when the organization attaches value to the knowledge embedded within the legacy processes despite the need for cleanup. A decision to proceed with SPR recognizes existing problems and assumes there is important knowledge that should not just be thrown away.

I.2 Determine readiness for SPR. Having recognized that there is a problem, the organization must determine whether they are ready to proceed with SPR. First, the organization should reflect upon its prior successes and failures in process implementation and try to draw upon lessons learned to enhance their future success rate (SPR Principle 2).

Second, the organization should heed important lessons from the SPI literature on implementation success: secure sufficient management commitment and allocation of resources [63], [64]. Finally, the organization should ensure that the inputs to SPR are known. Software policies should have been explicated prior to beginning SPR, or they should be explicated from the very start. These software policies ensure that the SPR effort is well aligned with the organization's SPI strategy (SPR Principle 1). At this stage, it is also important to create a list of all legacy processes in preparation for subsequent evaluations.

I.3 Establish appropriate improvement organization. Once the organization has demonstrated commitment to the effort, an appropriate improvement organization should be created to execute and facilitate SPR. This would involve: a dedicated SPR project and its relation to the overall improvement organization, e.g. in the form of the SEPG and the steering committee for SPI [6], [13]. The SPR project should be provided with adequate resources, be staffed with respected and influential employees, and represent varied stakeholder perspectives (SPR Principle 1). As part of establishing an appropriate improvement organization, plans for SPR should be detailed and expectations and responsibilities should be explicated.

These initiating activities ensure that the three inputs to the SPR process – legacy processes, software policies, and improvement organization – are in place.

Diagnosing

The key objectives of the diagnosing phase are to understand current practices and to establish a baseline for further improvement [38]. We suggest the following key diagnosing activities in SPR:

D.1 Characterize legacy processes. The improvement organization should systematically characterize the documentation status, current implementation status, and desired implementation status for each legacy processes (SPR Principles 2 and 4).

Documentation status: Legacy processes are evaluated with respect to conformance with documentation standards; consistency with software policies and overall strategic direction; and clarity of process description. A described process should also represent best practice within the organization. Using these criteria, the improvement organization may use the following scale to characterize documentation status for each legacy process:

- **Obsolete** – The legacy process is no longer appropriate and should be deleted. Technological and organizational changes can cause a legacy process to become obsolete. A legacy process may be labeled obsolete if it is inconsistent with current software policies, provides wrong level of detail to be valuable, suggests ideas that are no longer considered best practice, or relies on technologies that are no longer relevant to the company (e.g. coding guidelines for a programming language no longer in use).
- **Needs revision** – The legacy process needs revision to be useful for practice. These revisions could range from minor changes, such as ensuring conformance with

documentation standards, to major ones, such as ensuring alignment with current software policies.

- Needs approval – The legacy process is ready to be reviewed for approval. This means that the legacy process meets standards for conformance, consistency, clarity, and best practices.
- Approved – The legacy process has been reviewed by the appropriate group within the improvement organization and is ready to be implemented.

Implementation status: Decisions on which legacy processes to reengineer should be based on realistic assumptions about their implementation (SPR Principle 2). Based on [65], the improvement organization may use the following scale to characterize implementation status for each legacy process:

- Not used (<20%) – The legacy process is either used rarely within the organization or used by only a small subset of the organization.
- Discretionary (<60%) – The legacy process is used at the discretion of the project manager and may not be applicable for all projects.
- Normally used (<90%) – The legacy process is used consistently by almost all projects within the organization; however, there are a few known compliance issues that need to be addressed.
- Standardized (>=90%) – The legacy process is institutionalized within the organization's culture and daily practices and adapted to the needs of each new project.

D.2 *Assess process repository.* The existing process repository platform should be evaluated based on its usefulness, ease in locating related process documents, and suitability for both internal and external stakeholders (SPR Principles 1 and 3). There are a number of documented techniques that can be applied to evaluate process repositories [66], [67], [68].

D.3 *Diagnose process management.* Existing process management practices should be evaluated. Various strategies for process assessment can be applied [69], [70]. Appreciative inquiry focuses on identifying the strengths of the organization and on positive change [71]. Problem-based approaches focus on identifying and solving problems seen as hindering process management [72]. Finally, model-driven approaches compare current practices against best practices with discrepancies indicating areas where improvement is needed [72]. Generic best practices for process management are available in [5], [7].

When these three diagnostic activities have been completed, the improvement organization will have taken the first steps to bring the legacy processes under management control, identified the strengths and weaknesses of its process repository, and assessed current process management practices. These insights should be communicated to stakeholders outside of the SPR team for confirmation and debate.

Establishing

The establishing phase uses diagnostic information to create a strategic action plan for SPR which contains both short-term and long-term goals [38]. The action plan should address improvements of managed process documents, the process repository, and software process management practices:

E.1 Assign action status. The SPR team should identify the appropriate actions on each processes based on its specified documentation and implementation statuses. Given limited resources, a major portion of creating the SPR action plan involves prioritizing and scheduling which legacy processes to be innovated. Decisions on which processes to innovate should be based on realistic assumptions about their implementation (SPR Principle 2). A comparison between current and desired implementation status can help prioritize. The SPR team may gain most by focusing on processes with the biggest gap between desired (e.g. standardized) and current (e.g. not used) implementation status. Alternative prioritization schemes could use factors such as available resources, dependency with other processes, degree of changes needed, number of stakeholders that need access to the legacy process, and degree to which this process aligns with strategic priorities.

Action status: A variety of actions should be considered – from radical replacement to minor revisions (SPR Principle 4). Accordingly, each process should be assigned one of the following action statuses:

- Discard – These processes should be moved to an archive database or deleted. Legacy processes with documentation status of “obsolete” will most likely be discarded.
- Redesign later – These processes need modification; however, they are given a low priority at this time.
- Redesign now – These processes are considered important to the organization but need modification to more closely reflect desired practices. These legacy processes have a documentation status of “needs revision” and will be immediately addressed by the improvement organization.
- Submit for approval – These processes have a documentation status of “needs approval” and should be scheduled for review as soon as the process management infrastructure has been firmly established.

E.2 Redesign process repository. The SPR action plan should also suggest innovations for the process repository based on the diagnosis of the existing platform. The suggested changes will depend heavily on the results of the diagnosis. However, in general, the improvement team should ensure that the repository: meets the needs of both internal and external stakeholders; provides straight-forward, easy access to relevant documents; applies configuration management to ensure only the most up-to-date document gets updated; and, provides capabilities to archive documents that are no longer useful without deleting them (SPR Principle 3).

E.3 Outline process management plan. Finally, the action plan should address topics such as determining standards for process documents, auditing new processes to ensure that they meet these standards, issuing approval for documents, identifying processes that need

revision, and carrying out ongoing management of the process repository. These activities should be included into the process management plan to create a sustainable basis for continuous process management after SPR (SPR Principle 5); otherwise, the organization will again find itself growing more legacy processes over time.

As suggestions are being made during the establishing phase, it is important to consider strategies for mitigating possible resistance to change [73], [74]. A detailed implementation plan should include milestones, involvement of key stakeholders, and mechanisms for measuring and tracking progress. In keeping with the iteration principle (SPR Principle 5), the action plan should strive for small iterations of successful change.

Acting

During the acting phase, the strategic action plan is executed, deploying changes throughout the organization [38]. With SPR, this involves the following activities:

A.1 Reengineer legacy processes. SPR should be concerned with the documents as well as their impact on implementation efforts. Processes should be reengineered according to the action status assigned during the establishing phase. If action is needed, resources are assigned to make these changes (e.g. remove document from repository, bring document inline with standards, or modify to reflect desired best practices). Depending upon the scope of the change, this may involve considerable interaction and discussion among many members of the organization. Software processes that have been submitted for approval are reviewed by the appropriate process management team based upon conformance, consistency, clarity, and desirability for best practice. If the document is approved, this review should further consider how to ensure a smooth transition to the newly documented processes. Advice on implementing process change can be found in [4], [56], [64], [75]. At a minimum, employees should be made aware of the changes and told where to find the newly approved documents in the process repository.

A.2 Develop process repository. Following the proposed redesign, a new process repository is developed and tested for compliance with relevant stakeholder needs.

A.3 Pilot process management. A process management group should be identified to pilot the mechanisms outlined in the process management plan. Lessons learned from this experience can lead to refinements of the plan for continuous process management.

Implementing the SPR action plan is a highly iterative process in which solutions must be tested and modified (SPR Principle 5). Compared with other phases, it requires substantial amount of time and resources as many stakeholders have to work together to help turn new solutions into organizational practices.

Leveraging

The leveraging phase is a time of critical reflection in which lessons learned during earlier phases are used to inform future SPI cycles [38]. With SPR, this involves two activities:

L.1 *Evaluate achievements.* The SPR team should collect data from the effort, analyze them, and suggest important lessons learned. In particular, it should evaluate whether the intended objectives were met.

L.2 *Determine whether to exit.* The SPR team should decide whether to exit from the IDEAL cycle or whether additional cycles are required to meet project objectives. SPR is complete when all legacy processes have either been approved or discarded, the process repository has been revised to meet relevant stakeholder needs, and the process management plan has been approved and piloted. If the criteria for ending SPR are not met, a new SPR cycle can be started from any of the previous phases. If the criteria for ending SPR are met, the organization is ready to focus on software process management and continued SPI efforts. Table 1 summarizes the impact of the SPR principles on each of the SPR steps.

Table 1: Impact of SPR Principles on SPR Model

| SPR principle | Implications for SPR Steps |
|--|--|
| 1. SPR should consider the organizational context by identifying goals and policies for SPI and incorporating viewpoints of internal and external stakeholders. | <ul style="list-style-type: none"> ▪ The organization’s current software policies should be explicated and used as drivers for SPR. (IDEAL) ▪ The improvement organization should contain representatives from various stakeholder groups. (I) ▪ Consider the viewpoints of internal and external stakeholders during the transformation of legacy processes to managed processes. (DEA) ▪ When creating the process repository and process management plan, ensure that they meet the needs of both internal and external stakeholders. (DEA) |
| 2. SPR should consider the organization’s change practices by critically reviewing previous SPI initiatives and results and by taking measures to avoid previous failures. | <ul style="list-style-type: none"> ▪ Review successes and failures in past process implementations to enhance the success rate. (I) ▪ Legacy processes should be critically examined to determine their current usefulness and implementation status. (D) ▪ Decisions on which legacy processes to innovate and implement should be based on realistic assumptions about their implementation. (EA) ▪ Process Management Plan should be based on a realistic and sustainable approach to implement a process management discipline. (DEA) |
| 3. SPR should leverage IT capabilities to establish a platform for effective storage, communication, and usage of managed processes. | <ul style="list-style-type: none"> ▪ The process repository should facilitate effective storage, communication, and usage of all managed processes. (DEAL) ▪ The repository should allow stakeholders easy access to apply relevant software processes. (DEAL) ▪ The process management plan should maintain an up-to-date and easy-to-access process repository. (DEAL) |

| SPR principle | Implications for SPR Steps |
|---|---|
| 4. SPR should rely on multiple strategies (redevelopment, migration, and wrapping) that are contingently applied based upon current process portfolios and engineering practices. | <ul style="list-style-type: none"> ▪ SPR should systematically characterize all legacy processes. (DEA) ▪ SPR should consider a range of actions when turning legacy processes into managed processes. (EA) |
| 5. SPR should iteratively turn legacy processes into managed processes to enhance learning and to develop a sustainable baseline for continued SPI | <ul style="list-style-type: none"> ▪ The SPR model follows the IDEAL [38] iterative improvement model. (I) ▪ Legacy processes are transformed to managed processes by iteratively characterizing and modifying their status. Each step provides additional clarity on the opportunities and challenges related to bringing all software process under management control. (DEA) ▪ The process management plan should create a sustainable basis for continued SPI. (DEA) |

5. Industrial Experience

We proceed to describe how our collaboration with *TelSoft* raised awareness of the need for SPR as well as provided an environment for applying the proposed SPR model to industrial practices.

5.1 SPI History and Context

TelSoft has roughly 50 employees dedicated to software development. Over the last 35 years, *TelSoft* has evolved from being an engineering services firm primarily performing computer-aided drafting to becoming a software solutions provider that customizes geographic information systems for telecommunications and utilities industries. In this section, we present *TelSoft*'s two major SPI initiatives which set the stage for SPR (as summarized in Table 2).

First SPI Initiative

Wanting a definitive measure of its software engineering proficiency, *TelSoft*'s management set a goal of reaching level three on the SW-CMM. To that end, in July 2000 *TelSoft* established an SEPG [13] consisting of a project manager, three standing committee members, and rotating representatives from each of the four major groups within software development. The SEPG informally assessed *TelSoft* at SW-CMM level one.

The group met to consider how processes could be improved. They began to vigorously develop new software processes and document them through detailed guidelines and associated templates and checklists. During the following year, the group created over 75 documents covering areas such as project planning, requirements management, release

planning, software coding standards, and quality assurance. Despite high productivity rates and perceptions of progress in SPI, support for the SW-CMM initiative was withdrawn in August 2001 due primarily to financial pressures. *TelSoft* decided to commit its resources to imminent development rather than SPI.

Second SPI Initiative

Three years after the SEPG was disbanded, *TelSoft* engaged in collaboration with a group of researchers (including the three authors) from a nearby University Innovation Center (UIC). Our relationship to *TelSoft* was organized as a focused R&D collaboration [76] with the dual purpose [77] of revitalizing SPI efforts at *TelSoft* and at the same time contributing knowledge to the scientific community.

The overall improvement initiative was managed by two standing groups: the Software Coordination Group (SCG) and the Problem Solving Team (PST). The SCG consisted of *TelSoft*'s President, Vice President of Software Development, Software Development Manager, and Product Marketing Manager. The SCG met monthly to set strategic direction for *TelSoft*'s software products, monitor SPI initiatives, and manage the portfolio of software projects. These meetings were planned and facilitated by UIC researchers. The PST consisted of three highly regarded *TelSoft* engineers and managers and three UIC researchers. The PST held responsibility for prioritizing improvement initiatives and establishing improvement projects to focus on specific software processes.

After completing a thorough diagnosis of software practices (described in [78]), the PST identified seven improvement areas: software vision management, project portfolio management, software configuration management, customer relations, requirements management, software quality assurance, and end-user interaction. The PST instituted two action cycles to address these improvement areas. The first action cycle consisted of five improvement projects focused on software coordination processes, quality assurance, requirements management, configuration management, and customer relations. These projects revised some legacy processes while generating additional software processes. It was during these interactions that the PST became aware of two problems with process management. First, the legacy processes varied greatly from actual software practices. This mismatch occurred, in part, because *TelSoft*'s software development group allowed client demands rather than internal guidelines to drive their actions. Second, no procedures existed for managing software processes. The PST decided to tackle these problems during the second action cycle.

Table 2: SPI at *TelSoft*

| | First SPI Initiative (July 2000 – August 2001) | Second SPI Initiative (October 2004 – December 2006) |
|------------|---|---|
| Goal | Achieve SW-CMM Level 3 to comply with customer requirements | Solve perceived problems in software development |
| Leadership | Internal employees. Limited support from external | Internal employees. Ongoing facilitation through |

| | | |
|--------------|--|---|
| | consultant (2 day training on SW-CMM). | collaboration with UIC. |
| Organization | SEPG: <ul style="list-style-type: none"> ▪ 1 full-time employee as team leader ▪ 3 standing team members ▪ 4 team members that rotated out every 3 months | PST, SCG, and focused improvement projects. |
| Approach | Initiatives organized as one big project. Each initiative mainly driven by individuals. | Initiatives organized into two action cycles. Each initiative driven by a team. |
| Sponsorship | Supported by Vice President | Supported by President and CEO |

5.2 Application of SPR Model at TelSoft

In this section, we detail how the improvement organization worked together to execute SPR at *TelSoft* during the second action cycle. The section concludes with specific lessons learned.

Initiating

I.1 *Assess need for SPR.* *TelSoft* was a candidate for SPR because it had a large repository of legacy processes and no procedures in place for software process management. While some legacy processes created during the first SPI initiative were clearly obsolete, other legacy processes were actively used by the software development group or needed modification to become useful. The PST valued the knowledge contained within many of the legacy processes; therefore, rather than throw away the legacy processes, the PST decided to reengineer them.

I.2 *Determine readiness for SPR.* There were three indicators that *TelSoft* was ready to tackle SPR: its reflective stance on prior SPI initiatives, demonstration of senior management commitment to SPR, and adoption of software policies to guide reengineering.

- Steps were taken to try to overcome weaknesses from the first SPI initiative. Rather than focusing on achieving a specific SW-CMM level, the initiative was driven by problems perceived to be important by key organizational stakeholders. The improvement organization included a broad range of employees and used experienced outside facilitators throughout the change process.
- *TelSoft's* upper management had committed to collaborate with the UIC for a two-year period to effectively reengage in SPI. They had witnessed some success during the first action cycle and were, therefore, enthusiastic about continuing. Furthermore, they realized that as the SPI initiative continued, they would be adding more software processes to the repository, potentially increasing the

problem of legacy processes. To address this problem, they decided to implement systematic software process management.

- During the first action cycle, TelSoft had created software policies. These policies were brief and enforceable, stating desired practices that the SPI program should develop (see Table 3). The policies had been suggested by the improvement teams, consolidated by the PST, debated by software development employees, and approved by the SCG. Recognizing the dynamic nature of policies and priorities, the SCG was reviewing the policies quarterly to assess whether modifications were required.

I.3 *Establish appropriate SPI organization.* The PST established the SPR team and gave it five months to place legacy processes under management control, revise the existing process repository, and create a process management plan. Members of this cross-functional team included the manager of the first SPI effort and the developer targeted to be responsible for the new process management process.

Table 3: Software Policies at TelSoft

| <i>Area</i> | <i>Policy</i> |
|-------------------------------|---|
| 1. Professional Standards | <i>TelSoft</i> will strive to operate based on the highest professional standards and processes. |
| 2. Customer Knowledge | <i>TelSoft</i> will strive to understand and incorporate its customers' business knowledge in our products. |
| 3. Relationship Management | <i>TelSoft</i> will maintain a proactive professional relationship to its customers. |
| 4. Two-phase Funding | <i>TelSoft</i> will manage each development project with a two-phase approach that separates requirement and development activities. |
| 5. Requirements First | <i>TelSoft</i> will only engage resources to start design and construction when <i>TelSoft</i> has a baseline of identifiable and agreed upon requirements. |
| 6. Change Request | <i>TelSoft</i> will only engage resources to address requirement change requests that are documented, agreed upon and applied to the requirements baseline. |
| 7. Communicate Status | <i>TelSoft</i> will communicate status to its customers of all active projects on a regular basis. |
| 8. Quality Assurance Approval | <i>TelSoft</i> will only deliver official releases of software to a client with the written approval of Quality Assurance. |
| 9. Release Documentation | Each release of <i>TelSoft</i> software will include documentation of all changes and new features since the previous release. |

Team members were asked to work on SPR for no more than 4 hours every two weeks, signaling a preference for pragmatic decision making over comprehensive consideration of all options. Like all other improvement teams, the SPR team reported to the PST.

The initiating steps concluded at *TelSoft* with the three inputs to SPR process firmly in place: 75 legacy processes, 9 software policies, and an improvement organization to guide SPR consisting of the SPR team, the PST, and the SCG.

Diagnosing

D.1 *Characterize legacy processes.* Given many legacy processes but limited resources, the SPR team selected an iterative approach to SPR. They would first characterize all the legacy processes according to relevant attributes; they would then use those attributes to select the legacy processes that would get reengineered first. Therefore, the SPR team captured the following relevant attributes for each legacy process: documentation status, current implementation status, desired implementation status, desired visibility for customers, and associated software policies.

Reaching agreement on these attributes for each legacy process was not a straightforward, simplistic process. The SPR team tried various approaches before falling into a method that worked. At first, the SPR team asked a *TelSoft* employee who was also on the team to do the assessment with minor assistance of two UIC researchers. Although it proved fairly easy to reach agreement on the legacy processes that were obsolete, this group lacked the authority and knowledge required to assign current and desired implementation status.

The second attempt at assigning attributes was designed to get more input from other members of the SPR team. Each week, all members of the SPR team were assigned 4-6 legacy processes to assess; they could also add specific suggestions on how to improve the documents. The responses were collected and any disagreements were discussed at the SPR team meeting. This approach had the benefit of allowing a more careful review of the processes and getting specific suggestions from a variety of stakeholders. However, it was time intensive and the SPR team did not have a big picture view of *TelSoft's* software development process.

To solve the challenges of lack of authority, tendency toward detail and thoroughness, too much pressure on one person, and having the right people involved, the PST finally decided they were better suited to make the assessments. Each member of the PST assessed all processes independently. During a series of three meetings of about two hours each, the PST then discussed and negotiated the assessment of all legacy processes. The presence of the Vice President of Software Development and the Software Development Manager made it easier to deal with the strategic questions of desired implementation.

D2. *Assess process repository.* *TelSoft's* process repository was assessed from the viewpoint of two key stakeholders: the internal *TelSoft* employees and the external customers. The existing repository was a convenient choice for *TelSoft* employees: it was

fully integrated with the system they used for email, scheduling meetings, and sharing documents. The main problems were due to the volume of documents that existed in the database and the haphazard way in which documents were organized. Employees complained that relevant processes were difficult to find.

While employees suffered from information overload, external customers had the opposite problem. They had no access to the process repository and had limited insight into software development practices at *TelSoft*. This lack of information coupled with some performance problems, led them to reduced confidence in the organization. The SPR team found that giving customers access to key software processes would help *TelSoft* present a more professional image for both current and future customers.

D.3 Diagnose process management practice. A detailed diagnosis had been conducted prior to the first action cycle revealing the following problems with process management practice:

- At *TelSoft*, there was no systematic process management group in place to approve documents or manage the process repository. Any person within the software development group had the authority to create process documents. These documents were typically reviewed by members of the *TelSoft* management group for informal approval before being placed in the LotusNotes repository.
- There were no written standards for process documents.
- Changes to software processes were not centrally managed. Once documents were placed within the repository, the document's original author could make changes to the document without notifying anyone.
- Several written processes had little impact on engineering practices. Many software processes were neither read nor enforced. More likely, it was the case that documents were written and then largely forgotten unless the management team insisted upon conforming and monitored compliance.

Through these diagnosing activities, the PST and SPR team began to appreciate the problems with legacy processes, process repository, and process management practice.

Establishing

E.1 Assign action status for managed processes. The PST assigned action status to each process. The obsolete processes were immediately discarded. The processes with "needs approval" status were assigned "Submit for approval" status and held until the process management process had been defined. For the 19 managed processes with "needs revision" status, the PST decided to reengineer the processes iteratively. In the first wave of modifications, they assigned an action status of "redesign now" to the processes they felt should be visible to customers; all other documents were assigned an action status of "redesign later." The second wave of modifications would focus on those processes where the current and desired implementation statuses were not aligned (see Table 4)

Table 4: "Redesign Later" Processes with Misaligned Implementation Status

| Current Implementation Status | Desired Implementation Status | Count |
|--------------------------------------|--------------------------------------|--------------|
| Discretionary | Normally Used | 1 |
| Normally used | Standardized | 7 |
| Not Used | Discretionary | 1 |
| Not Used | Normally used | 2 |
| Not Used | Standardized | 1 |
| | Total | 12 |

E.2 *Redesign process repository.* The main improvement for internal stakeholders was to reduce the number of obsolete processes cluttering the existing repository. To increase external stakeholders' visibility into *TelSoft's* processes, the SPR team decided to redesign the company's website to fully describe the software policies, show selected software processes which support these software policies, and described the SPI effort.

E.3 *Outline process management plan.* The SPR team created standards for templates and processes. These standards would be used to assess whether processes could be marked as "approved". A process management plan was created that involved: making process management a responsibility of the existing quality assurance group; adding a process monitoring and control activity to the monthly PST meetings; maintaining the documentation, implementation, and action status; and yearly assessment of how well policies were being implemented.

Acting

A.1 *Reengineer legacy processes.* The processes that were assigned status of "redesign now" were modified and reviewed for conformance with standards before being approved.

A.2 *Develop process repository.* The website underwent several iterations to arrive at a design which was easy to navigate and provided succinct and relevant information to external stakeholders. The new updates were deployed on schedule by the October 2006 deadline.

A.3 *Pilot process management plan.* The process management plan went through several rounds of internal review and debate before being approved by the PST. This activity ended with (1) a pilot meeting of the PST in which the process management monitoring and control was executed, (2) a transfer of responsibility for daily management of processes to the quality assurance group, and (3) a workshop to announce the new process management processes to the entire software development group

Leveraging

L.1 *Evaluate achievements.* *TelSoft's* SPR effort was designed to eliminate legacy processes, update the process repository, and improve their process management discipline. As a result of this process, 26 of the 75 legacy processes were considered useful for retaining (see Table 5 for summary of managed processes).

Table 5: Summary of Management Processes at *TelSoft*

| Documentation Status | Current Implementation Status | | | | Total |
|----------------------|-------------------------------|---------------|---------------|--------------|-------|
| | Not used | Discretionary | Normally used | Standardized | |
| Needs revision | 4 | 6 | 7 | 2 | 19 |
| Needs approval | 0 | 3 | 0 | 3 | 6 |
| Approved | 0 | 0 | 0 | 1 | 1 |
| Totals | 4 | 9 | 7 | 6 | 26 |

Specific lessons learned during this experience include:

1. SPR should consist of team members with sufficient authority and process knowledge to evaluate documentation and implementation status. These statuses, particularly the desired implementation status, drive SPR and should represent a commitment from *TelSoft* upper management team to assign the required resources.
2. SPR should take advantage of frequent feedback from improvement teams and software engineers in general. The SPR team at *TelSoft* had difficulties early on that were resolved only when the PST actively asked questions and involved key stakeholders.
3. SPR should use agreed-upon policies to prioritize action planning. *TelSoft* had agreed to policies prior to SPR; however, they had not yet prioritized those policies. As it became clear that they could not revise all legacy processes at once, *TelSoft* used the policy mapping to help determine which documents they should focus on first.
4. Publicizing policies and key processes demonstrated to *TelSoft* customers that a systematic development approach is being followed; they created positive expectations to *TelSoft*'s focus on client relationships; and, they reinforced *TelSoft*'s commitment to long-term, continuous improvement of its software practices.
5. Developing and piloting the plan for software process management made the PST realize what is required to sustain and institutionalize a process discipline at *TelSoft*.

L.2 *Determine whether to exit.* The PST decided to exit from SPR as the process repository had been sufficiently revised to meet stakeholder needs. The quality assurance team had practiced checking processes against standards. The PST had created a baseline of the documentation, implementation, and action statuses for all software processes. They were committed to reviewing this status on a monthly basis.

6. Conclusions and Future work

SPI has become one of the major approaches to improve performances within the software industry. While there are many success stories presented in the literature, SPI is not without complications. Software organizations involved in SPI might decide to focus resources on other business issues, or they might develop a portfolio of processes without having a proper process management discipline in place. As a result, these organizations will increasingly face legacy software processes that are inconsistent with current software practices and policies. This

research has addressed this challenge by developing a systematic and practical model for transforming legacy software processes to managed processes. The presented SPR model uses software policies to guide the reengineering effort. The feasibility of the model is demonstrated based upon an industrial case study of a small software organization, *TelSoft*. The model had several key benefits: it engaged key stakeholders in *TelSoft*'s improvement efforts; it effectively communicated the organization's software practices; and, it created a solid platform for institutionalizing a process management discipline. As other software organizations engage in SPR, their situation will be different from the one at *TelSoft*. It is therefore important they carefully consider the context for SPR (SPR Principle 1) to help adapt the proposed model to their specific needs. Future research is needed to investigate the suitability of the model within other software organizations as well as to analyze its long-term effectiveness.

The presented research has also provided conceptual clarity regarding the problem of legacy software processes and the need for software process reengineering. A key point is that organizations' history with SPI impacts their ability to move forward. This is especially true for those that follow SPI approaches with a heavy focus on generic, documented processes that are tailored to individual projects. When these software organizations fail to institute proper process management practices or when they decide to reinvest in SPI, they may likely be confronted with a considerable portfolio of legacy processes. Future research needs to further appreciate this problem and reconsider how software organizations can effectively develop and implement process management solutions.

Finally, this research integrates lessons from business process change, SPI, and legacy software systems to provide principles (as described in Section 3) for SPR. Practitioners can use these principles as basis for adapting the proposed SPR model to their particular context and needs. In addition, future research can further explore how such broader knowledge from related disciplines can be used to further develop knowledge and practices within SPI.

Acknowledgment

The authors wish to thank the employees and management at *TelSoft* and [name withheld during review process]. This work was supported in part by grants from [name withheld during review process], *TelSoft*, and United States Department of Education.

References

- Adler, P. S., B. Goldoftas, et al. (1999). "Flexibility versus Efficiency? A Case Study of Model Changeovers in the Toyota Production System." *Organization Science* **10**(1): 43-68.
- Adler, P. S., F. McGarry, et al. (2005). "Enabling process discipline: lessons from the journey to CMM level 5." *MIS Quarterly Executive* **4**(1): 215-226.
- Agile Alliance. (2001). "Manifesto for Agile Software Development." Retrieved May, 2006, from <http://www.agilemanifesto.org/>.
- Argyris, C. (1985). *Action Science*. San Francisco, Jossey-Bass.
- Avison, D., R. Baskerville, et al. (2001). "Controlling action research projects." *Information Technology & People* **14**(1): 28-45.
- Avison, D., F. Lau, et al. (1999). "Action Research." *Communications of the ACM* **42**(1): 94-97.
- Avison, D. and T. Wood-Harper (1990). *Multiview: an exploration in information systems development*. New York, McGraw-Hill.
- Baker, S. (2005). Formalizing agility: an agile organization's journey toward CMMI

- accreditation. Agile Conference.
- Barley, S. R. and G. Kunda (2001). "Bringing Work Back In." Organization Science **12**(1): 76-95.
- Baskerville, R. and T. Wood-Harper (1996). "A Critical Perspective on Action Research as a Method for Information Systems Research." Journal of Information Technology **11**: 235-246.
- Baskerville, R. and T. Wood-Harper (1998). "Diversity in information systems action research methods." European Journal of Information Systems **7**(2): 90-107.
- Baskerville, R. L. (1999). "Investigating information systems with action research." Communications of the AIS **2**(3es).
- Baum, J. A. C., S. X. Li, et al. (2000). "Making the Next Move: How Experiential and Vicarious Learning Shape the Locations of Chains' Acquisitions." Administrative Science Quarterly **45**(4): 766-801.
- Beck, K. (1999). Extreme Programming Explained: Embrace Change. Reading, MA, Addison-Wesley.
- Beecham, S., T. Hall, et al. (2005). "Defining a Requirements Process Improvement Model." Software Quality Journal **13**(3): 247-279.
- Benner, M. J. and M. Tushman (2002). "Process Management and Technological Innovation: A Longitudinal Study of the Photography and Paint Industries." Administrative Science Quarterly **47**(4): 676-709.
- Benner, M. J. and M. L. Tushman (2003). "Exploitation, Exploration, and Process management: The productivity Dilemma Revisited." Academy of Management Journal **28**(2): 238-256.
- Birkinshaw, J. and C. Gibson (2004). "Building ambidexterity into an organization." Sloan Management Review **45**(4): 47-55.
- Boehm, B. W. (2002). "Get ready for agile methods, with care." Computer **35**(1): 64-69.
- Boehm, B. W. and R. Turner (2004). Balancing agility and discipline: a guide for the perplexed. Boston, Addison-Wesley.
- Checkland, P. and S. Holwell (1998). "Action research: its nature and validity." Systemic Practice and Action Research **11**(1): 9-21.
- Clark, K. B. and S. C. Wheelwright (1992). "Organizing and Leading 'Heavyweight' Development Teams." California Management Review **34**(3): 9-28.
- CMMI Product Team (2002). CMMI for Systems Engineering/Software Engineering/Integrated Product and Process Development/Supplier Sourcing, Software Engineering Institute.
- Cockburn, A. (2000). Writing Effective Use Cases. Reading, MA, Addison-Wesley.
- Collins, J. C. and J. I. Porras (1994). Built to Last: Successful Habits of Visionary Companies. New York, HarperCollins Publishers.
- Cook, S. D. N. and J. S. Brown (1999). "Bridging epistemologies: the generative dance between organizational knowledge and organizational knowing." Organization Science **10**(4): 381-400.
- Cunningham, J. B. (1993). Action research and organizational development. Westport, CT, Praeger.
- Curtis, B., W. E. Hefley, et al. (2002). People Capability Maturity Model: Guidelines for Improving the Workforce. Boston, Addison-Wesley.
- Davison, R. M., M. G. Martinsons, et al. (2004). "Principles of canonical action research." Information Systems Journal **14**(1): 65-86.
- De Reyck, B., Y. Grushka-Cockayne, et al. (2005). "The impact of project portfolio management on information technology projects." International Journal of Project Management **23**(7): 524-537.
- Deming, W. E. (1986). Out of Crisis. Cambridge, MA, MIT Center of Advanced Engineering Study.
- Duncan, R. B. (1976). "The Ambidextrous Organization: Designing Dual Structures for Innovation." Kilmann, RH Pondy, LR, and DP Slevin (eds.). The Management of Organization Design, I, New York: Elsevier North-Holland: 167-188.
- Dyba, T. (2005). "An Empirical Investigation of the Key Factors for Success in Software Process Improvement." IEEE Transactions on Software Engineering **31**(5): 410.

- El Emam, K. and A. Birk (2000). "Validating the ISO/IEC 15504 measure of software requirements analysis process capability." IEEE Transactions on Software Engineering **26**(6): 541.
- Floyd, S. and P. Lane (2000). "Strategizing throughout the organization: Managing role conflict in strategic renewal." Academy of Management Review **25**: 154-177.
- Fuggetta, A. (2001). Software Process: A Roadmap. R. Hunter and R. H. Thayer. Los Alamitos, CA, IEEE Computer Society: 559-566.
- Ghoshal, S. and C. A. Bartlett (1994). "Linking organizational context and managerial action: the dimensions of quality of management." Strategic Management Journal **15**(Summer): 91-112.
- Gibson, C. and J. Birkinshaw (2004). "The Antecedents, consequences, and mediating role of organizational amidexterity." Academy of Management Journal **47**(2): 209-226.
- Goldenson, D. R. and J. D. Herbsleb (1995). After the appraisal: a systematic survey of process improvement, its benefits, and factors that influence success. Pittsburgh, PA, Software Engineering Institute.
- Gregor, S. (2006). "The Nature of Theory in Information Systems." MIS Quarterly **30**(3): 611-642.
- Gupta, A., K. Smith, et al. (2006). "The Interplay between Exploration and Exploitation." Academy of Management Journal **49**(4): 693-706.
- Haecel, S. (1995). "Adaptive enterprise design: the sense-and-respond model." Planning Review **23**(3): 6-13, 42.
- Haecel, S. (1999). Adaptive Enterprise: Creating and Leading Sense-and-Respond Organizations. Boston, MA, Harvard Business School Press.
- He, Z.-L. and P.-K. Wong (2004). "Exploration vs. Exploitation: An empirical test of the ambidexterity hypothesis." Organization Science **15**(4): 481-494.
- Hevner, A. R., S. T. March, et al. (2004). "Design Science in Information Systems Research." MIS Quarterly **28**(1): 75-105.
- Highsmith, J. (2000). Adaptive Software Development: A Collaborative Approach to Managing Complex Systems. New York, NY, Dorset House Publishing.
- Highsmith, J. and A. Cockburn (2001). "Agile Software Development: The Business of Innovation." Computer **34**(9): 120-127.
- Hobday, M. (2000). "The project-based organisation: an ideal form for managing complex products and systems." Research Policy **29**(7-8): 871-893.
- Horvat, R. V., I. Rozman, et al. (2000). "Managing the complexity of SPI in small companies." Software Process: Improvement and Practice **5**(1): 45-54.
- Humphrey, W. S. (1989). Managing the Software Process. Boston, MA, Addison-Wesley.
- Iversen, J., L. Mathiassen, et al. (2004). "Managing risk in software process improvement: an action research approach." MIS Quarterly **28**(3): 395-433.
- Iversen, J., P. A. Nielsen, et al. (2002). Problem Diagnosis in SPI. Improving Software Organizations: From Principles to Practice. L. Mathiassen, J. Pries-Heje and O. Ngwenyama. New York, Addison-Wesley.
- Jansen, J. J. P., F. A. J. van Den Bosch, et al. (2005). "Exploratory innovation, exploitative innovation, and ambidexterity: The impact of environmental and organizational antecedents." Schmalenbach Business Review **57**: 351-363.
- Kautz, K. H., H. W. Hansen, et al. (2000). Applying and adjusting a software process improvement model in practice: The use of the IDEAL model in a small software enterprise. International Conference on Software Engineering, Limerick, Ireland.
- Kock, N. (1997). "Negotiating mutually satisfying IS action research topics with organizations: an analysis of Rapoport's initiative dilemma." Journal of Workplace Learning **9**(7): 253-62.
- Krasner, H., J. Terrel, et al. (1992). "Lessons learned from a software process modeling system." Communications of the ACM **35**(9): 91-100.
- Kuvaja, P. and A. Bicego (1994). "BOOTSTRAP - a European Assessment Methodology." Software Quality Journal **3**(3): 117-127.

- Langley, A. (1999). "Strategies for theorizing from process data." Academy of Management Review **24**(4): 691-710.
- Lee, G., W. DeLone, et al. (2006). "Ambidextrous coping strategies in globally distributed software development projects." Communications of the ACM **49**(10): 35-40.
- Lee, G., W. DeLone, et al. (2007). "Ambidexterity and Global IS Project Success: A Theoretical Model." System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on: 44-44.
- Lincoln, Y. and E. Guba (1985). Naturalistic Inquiry. Newbury, CA, Sage.
- Lindgren, R., O. Henfridsson, et al. (2004). "Design Principles for Competence Management Systems: A Synthesis of an Action Research Study." MIS Quarterly **28**(3): 435-472.
- Lubatkin, M. H., Z. Simsek, et al. (2006). "Ambidexterity and Performance in Small-to Medium-Sized Firms: The Pivotal Role of Top Management Team Behavioral Integration." Journal of Management **32**(5): 646-672.
- Lyytinen, K. (1988). "Stakeholders, IS failures and soft system methodology: an assessment." Journal of Applied Systems Analysis **15**: 61-81.
- March, J. G. (1991). "Exploration and Exploitation in Organizational Learning." Organization Science **2**(1): 71-87.
- Markowitz, H. (1952). "Portfolio Selection." The Journal of Finance **7**(1): 77-91.
- Mårtensson, P. and A. S. Lee (2004). "Dialogical Action Research at Omega Corporation." MIS Quarterly **28**(3): 507-536.
- Mason, J. (2002). Qualitative Researching. London, Sage.
- Mathiassen, L. (2002). "Collaborative practice research." Information Technology & People **15**(4): 321-345.
- Mathiassen, L., J. Pries-Heje, et al. (2002). Improving Software Organizations: From Principles to Practice. Boston, MA, Addison-Wesley.
- Mathiassen, L. and A. M. Vainio (2007). "Dynamic capabilities in small software firms: a sense-and-respond approach." IEEE Transactions on Engineering Management **54**.
- McFarlan, F. W. (1981). "Portfolio Approach to Information Systems." Harvard Business Review **59**(5): 142-150.
- McFeeley, B. (1996). IDEAL: A user's guide for software process improvement. Pittsburgh, PA, Software Engineering Institute.
- McKay, J. and P. Marshall (2001). "The dual imperatives of action research." Information Technology & People **14**(1): 46-59.
- Miles, M. B. and A. M. Huberman (1994). Qualitative Data Analysis: an expanded sourcebook. Thousand Oaks, Sage Publications.
- Mingers, J. (2001). "Combining IS Research Methods: Towards a Pluralist Methodology." Information Systems Research **12**(3): 240-259.
- Mingers, J. and A. Gill (1997). Multimethodology: The theory and practice of combining management science methodologies. Chichester, John Wiley & Sons.
- Napier, N. P., J. Kim, et al. (under review). "Software Process Reengineering: A Model and Its application to an industrial case study." IEEE Transactions on Software Engineering.
- Napier, N. P., L. Mathiassen, et al. (2006). Negotiating Response-ability and Repeat-ability in Requirements Engineering. International Conference on Information Systems, Milwaukee, Wisconsin.
- Napier, N. P., L. Mathiassen, et al. (2006). Perceptions and Processes in assessing software requirements practices. Proceedings of the Twelfth Americas Conference on Information Systems, Acapulco, Mexico.
- Napier, N. P., L. Mathiassen, et al. (2006). Perceptions and Processes in assessing software requirements practices. Proceedings of the Twelfth Americas Conference on Information Systems, Acapulco, Mexico.
- Napier, N. P., L. Mathiassen, et al. (under review). "Becoming Ambidexterous: A Contextualist Inquiry into a Small Software Firm." Organization Science.
- Nielsen, P. A. and J. Pries-Heje (2002). A Framework for selecting an assessment strategy. Improving software organizations: from principles to practice. L. Mathiassen, J. Pries-

- Heje and O. Ngwenyama, Addison-Wesley.
- O'Reilly III, C. A. and M. Tushman (2004). "The Ambidextrous Organization." Harvard Business Review **82**(4): 74-81.
- Orlikowski, W. and J. J. Baroudi (1991). "Studying information technology in organizations: research approaches and assumptions." Information Systems Research **2**(1): 1-28.
- Overby, E., A. Bharadwaj, et al. (2006). "Enterprise agility and the enabling role of information technology." European Journal of Information Systems **15**(2): 120-131.
- Paulk, M. (2001). "Extreme Programming from a CMM perspective." IEEE Software **18**(6): 19-26.
- Paulk, M., B. Curtis, et al. (1993). Capability Maturity Model for Software (Version 1.1), Carnegie Mellon University.
- Paulk, M., B. Curtis, et al. (1993). Capability Maturity Model for Software, Version 1.1. Pittsburgh, PA, Software Engineering Institute.
- Paulk, M., C. V. Weber, et al., Eds. (1995). The Capability maturity model: guidelines for improving the software process. SEI Series in Software Engineering. Boston, Addison-Wesley.
- Pettigrew, A. M. (1985). Contextualist Research: A Natural Way to Link Theory and Practice. Doing Research That is Useful for Theory and Practice. E. E. Lawler. San Francisco, Jossey-Bass.
- Pettigrew, A. M. (1987). "Context and Action in the Transformation of the Firm." Journal of Management Studies **24**(6): 649-670.
- Pettigrew, A. M. (1990). "Longitudinal field research on change: theory and practice." Organization Science **1**(3): 267-.
- Pouloudi, A. and E. Whitley (1997). "Stakeholder identification in inter-organizational systems: gaining insights for drug use management systems." European Journal of Information Systems **6**(1): 1.
- Ramesh, B., J. Pries-Heje, et al. (2002). Internet software engineering: a different class of processes. Annals of software engineering, Kluwer Academic Publishers. **14**: 169-195.
- Rapoport, R. (1970). "Three Dilemmas in Action Research." Human Relations **23**(6): 499-513.
- Ravichandran, T. and A. Rai "Quality management in systems development: An organizational system perspective." MIS Quarterly **24**(3): 381.
- Rising, L. and N. S. Janoff (2000). "The Scrum software development process for small teams." Software, IEEE **17**(4): 26-32.
- Rout, T. P. (1995). "SPICE: A Framework for Software Process Assessment." Software Process: Improvement and Practice **1**(1): 57-66.
- Salo, O. and P. Abrahamsson (2005). Integrating agile software development and software process improvement: a longitudinal case study. International Symposium on Empirical Software Engineering.
- Schwaber, K. and M. Beedle (2001). Agile Software Development with Scrum. Upper Saddle River, Prentice Hall.
- Smith, W. K. and M. L. Tushman (2005). "Managing Strategic Contradictions: A Top Management Model for Managing Innovation Streams." Organization Science **16**(5): 522-536.
- Software Engineering Institute (2006). Improving Processes in Small Settings (IPSS): A White Paper. Pittsburgh, PA, Carnegie Mellon University.
- Sommerville, I. and J. Ransom (2005). "An empirical study of industrial requirements engineering process assessment and improvement." ACM Transactions on Software Engineering and Methodology **14**(1): 85-117.
- Sommerville, I. and P. Sawyer (1997). Requirements Engineering: A Good Practice Guide. New York, NY, John Wiley & Sons.
- Susman, G. and R. Evered (1978). "An assessment of the scientific merits of action research." Administrative Science Quarterly **23**(4): 582-603.
- The Standish Group International. (2004). "2004 Third Quarter Research Report." from URL http://standishgroup.com/sample_research/PDFpages/q3-spotlight.pdf.

- Turk, D., R. France, et al. (2005). "Assumptions underlying agile software-development processes." Journal of Database Management **16**(4): 62-87.
- Tushman, M. and C. A. O'Reilly III (1996). "Ambidextrous organizations: Managing Evolutionary and revolutionary change." California Management Review **38**(4): 8-30.
- Van Aken, J. E. (2004). "Management research based on the paradigm of the design sciences: The quest for field-tested and grounded technological rules." Journal of Management Studies **41**(2): 219-246.
- Van de Ven, A. and M. Poole (1995). "Explaining development and change in organizations." Academy of Management Review **20**(3): 510-540.
- Vidgen, R. (1997). "Stakeholders, soft systems and technology: separation and mediation in the analysis of information system requirements." Information Systems Journal **7**(1): 21-46.
- Vinekar, V., C. W. Slinkman, et al. (2006). "Can agile and traditional systems development approaches coexist? An ambidextrous view." Information Systems Management **23**(3): 31-42.
- Weitzman, E. A. and M. B. Miles (1995). Computer Programs for Qualitative Data Analysis. Thousand Oaks, CA, Sage.
- Wenger, E. (1998). Communities of practice: Learning, meaning, and identity. Cambridge, Cambridge University Press.
- Yin, R. K. (2003). Case Study Research: Design and Methods. Thousand Oaks, Sage.
- Zbaracki, M. J. (1998). "The rhetoric and reality of total quality management." Administrative Science Quarterly **43**(3): 602-636.

Paper 4: Becoming Ambidextrous

Title: Becoming Ambidextrous: A Contextualist Inquiry
into a Small Software Firm

This paper is coauthored by Nannette Napier, Lars Mathiassen, and
Dan Robey

This version of the paper is submitted to *Organization Science*,
Special Issue on Ambidextrous Organizations,

Abstract

Ambidextrous organizations are argued to achieve high performance by simultaneously aligning their activities with existing customers while adapting to emerging market opportunities. Distinctions have been made in prior literature between structural ambidexterity, which separates alignment and adaptability into distinct activities, and contextual ambidexterity, which integrates both alignment and adaptability into the organization's systems, processes, and beliefs. For small firms with limited resources, contextual ambidexterity is an attractive proposition because it limits the complexity of formal organization structure. However, there is limited actionable advice on how managers can shape the organizational context to develop ambidextrous capability. On this backdrop, we report a two-year action research study of one small software firm's attempt to innovate project portfolio management. Drawing upon Pettigrew's guidance for contextualist inquiry, we show how changing degrees of alignment and adaptability interacted with the performance management and social support context over time. Based on these experiences, we propose a model for becoming ambidextrous through the processes of diagnosing, visioning, intervening, and practicing.

Introduction

To improve organizational performance, managers must often balance concerns which at times may seem contradictory. For instance, managers must decide where to invest resources to enhance performance and whether such investments should focus on aligning with existing customers in mature markets or on adapting to new customers in emerging markets. To reap the benefits of both alignment and adaptability, organizations have been advised to strive for ambidexterity – the paradoxical ability to pursue simultaneously contradictory capabilities such as exploration-exploitation (Tushman and O'Reilly III 1996), alignment-adaptability (Gibson and Birkinshaw 2004), and flexibility-efficiency (Adler, Goldoftas et al. 1999). Ambidextrous organizations compete by optimizing efficiency, cost, and incremental innovation while also exhibiting flexibility, speed, and radical innovation (Tushman and O'Reilly III 1996). Moreover, studies have begun to provide empirical support for a positive relationship between ambidexterity and organizational performance (Gibson and Birkinshaw 2004; He and Wong 2004).

Despite the anticipated benefits, achieving ambidexterity is by no means straightforward. Each of the contradictory capabilities requires different and often incongruent systems, processes, and beliefs, thereby creating conflicts and dilemmas that are challenging to resolve (Tushman and O'Reilly III 1996; Floyd and Lane 2000; Gibson and Birkinshaw 2004). How, then, can managers design ambidextrous organizations? Two general approaches have been suggested: structural and contextual ambidexterity. With structural ambidexterity, managers create separate business units within the organization which specialize in one required capability, and the top management team bears responsibility for coordinating contributions of the two units to achieve ambidexterity at the organizational level (Gibson and Birkinshaw 2004). With contextual ambidexterity, the responsibility for achieving ambidexterity is shared by members within a single business unit. To create a high performing business unit, the top management team is advised to create an organizational context which facilitates both alignment and adaptability through appropriate performance management and social support (Gibson and Birkinshaw 2004).

While many firms could benefit from being ambidextrous, they may lack the resources or stability required for creating dual structures as advised by structural ambidexterity. For small firms that operate in dynamic environments, the concept of contextual ambidexterity therefore seems most feasible (Lubatkin, Simsek et al. 2006). For these firms, the challenge then becomes one of increasing alignment and adaptability practices while simultaneously shaping the organizational context to support these practices. Although prior research on contextual ambidexterity has demonstrated that an organizational context with appropriate performance management and social support facilitates alignment and adaptability (Gibson and Birkinshaw 2004), the practical questions related to becoming ambidextrous have not been addressed. Specifically, how can organizations develop and engage in ambidextrous practices and create and sustain organizational contexts that facilitate such practices? What challenges will managers face during such transformation processes and how can they be addressed? How long does it take to become ambidextrous, and are there specific shortcuts which enable this process to go more quickly?

Our focus is therefore on contextual ambidexterity and our objective is to explore how organizations can develop managerial practices and organizational contexts as they strive to become ambidextrous. The research is framed as a two-year contextualist inquiry (Pettigrew 1985, 1987) based on action research (Susman and Evered 1978; McKay and Marshall 2001; Mathiassen 2002) into practices at *TelSoft*, a small software firm with a well-established customer base and a need to innovate its processes and products. Adopting action research principles allows us to get deep and first-hand insight into how contextual ambidexterity was approached and developed over time. Pettigrew's contextualist inquiry helps us to conceptualize and explore how content, context, and processes interacted and shaped each other over the two-year period. Our focus is on project portfolio management, i.e., the systematic management of the company's projects in order to decide which projects should be added or removed as well as the relative priority of projects within that portfolio (Markowitz 1952; McFarlan 1981; De Reyck, Grushka-Cockayne et al. 2005). In software firms that are project-based organizations, project portfolio management is a core management activity requiring ongoing assessment of existing projects and new business opportunities (Clark and Wheelwright 1992; Hobday 2000).

TelSoft is representative of small software firms. It is oriented toward known customers in a niche market; it has high reliance on committed employees who perform many roles within the organization; and it has few resources devoted to innovation (Horvat, Rozman et al. 2000). Although not considered a market leader, *TelSoft* has a reliable customer base consisting of two large customers that drive innovation to their core software products and several hundred smaller customers that use *TelSoft*'s standardized geographic mapping software. *TelSoft* management acknowledges that the company's biggest strength is its people: experienced software engineers with deep knowledge of its products, systems analysts with strong customer relationships, and managers willing to adapt quickly to customer requests. Due to recent financial pressures, *TelSoft* was forced to downsize its workforce, causing it to lose valuable customer and technical expertise, and also requiring that employees adopt additional roles and responsibilities. Struggling to survive in a competitive environment, *TelSoft* frequently neglected innovation and adaptation, and instead emphasized known customers, products, and services.

Software firms like *TelSoft* represent an ideal setting for studying contextual ambidexterity for three main reasons. First, software firms operate in competitive business environments characterized by frequent customer changes, rapid technological advances, and time-to-market pressures (Ramesh, Pries-Heje et al. 2002; Mathiassen and Vainio 2007). They must adapt quickly to such environmental changes to ensure customer satisfaction and technology acceptance. Second, software firms have a track record of poor performance: less than half of software development projects result in a quality software product that is delivered on time and within budget (The Standish Group International 2004). Consequently, software managers need to ensure that employees are working toward the common goal of developing software that meets or exceeds stakeholder requirements. Third, software firms face the need to integrate seemingly opposing development synergies. On the one hand, software innovation strategies need to emphasize the predictable “repeat-ability” of development processes while, on the other hand, strategies need to emphasize agility and “response-ability” (Boehm 2002; Napier, Mathiassen et al. 2006). While in the past there have been staunch advocates for one strategy over the other, recently there has been a renewed interest in how software firms can achieve the benefits of both approaches simultaneously (Holmberg and Mathiassen 2001; Boehm and Turner 2004; Salo and Abrahamsson 2005; Lee, DeLone et al. 2006; Napier, Mathiassen et al. 2006; Vinekar, Slinkman et al. 2006; Lee, DeLone et al. 2007). The integration of opposing capabilities would, in effect, require software firms to become ambidextrous.

This paper uses the *TelSoft* case as a basis for developing insights on how organizations develop managerial practices and organizational contexts as they strive to become ambidextrous. In the next section, we review the literature on contextual ambidexterity, and we introduce contextual inquiry as the analytical lens adopted in this study. The third section describes the research approach used to study *TelSoft*. The fourth section offers a detailed account of how *TelSoft* changed its project portfolio management capabilities. The fifth section discusses key insights from examining the changes in process, context, and content. The final section concludes the paper with suggestions for future research and practical guidance for managers.

Theoretical Background

Contextual Ambidexterity

Contextual ambidexterity requires simultaneous success at both alignment – the capacity of employees within the business unit to work toward common goal, and adaptability – the capacity of the business unit to change quickly in response to dynamic market conditions (Gibson and Birkinshaw 2004). With contextual ambidexterity, responsibility is shared among individual employees within a specific business unit. This perspective recognizes that the day-to-day activities of individual employees shape and reflect ambidexterity. Therefore, the top management team is charged with creating an organizational context that facilitates ambidextrous practices.

Following Ghoshal and Bartlett (1994), Gibson and Birkinshaw (2004) identify two salient aspects of the organizational context that can be manipulated to increase alignment and adaptability: performance management and social support. The performance management context represents systems, processes, and beliefs related to meeting performance objectives set by the organization’s management (Gibson and Birkinshaw 2004). Discipline is an attribute that encourages people to voluntarily meet those objectives whereas stretch is an attribute that

encourages people to strive for even more ambitious goals (Ghoshal and Bartlett 1994). The social support context represents systems, processes, and beliefs associated with member relationships (Gibson and Birkinshaw 2004). Trust is an attribute of the organizational context that encourages people to rely on one another whereas support is an attribute that empowers people to lend assistance to others (Ghoshal and Bartlett 1994).

With respect to software project portfolio management, discipline can be exhibited by consistently completing projects that meet stakeholder requirements on time and within budget. Stretch encourages project teams to focus and work hard to achieve goals that will add value to customers or open opportunities for new business. However, where stretch is not balanced with discipline, project-based organizations can experience problems. Designers and engineers can fall into the trap of adding unnecessary functionality (i.e., feature creep), and project managers can allow the scope of projects to expand to the point that projects are no longer profitable (i.e., scope creep). Beyond individual projects, discipline can be exhibited by ensuring that the existing project portfolio is well managed, resources are appropriately distributed, and underperforming projects are brought back on track or terminated. On this level, stretch is focused on exploring new technology or market options and making decisions to alter the existing project portfolio more strongly towards innovation. Again, the challenge for management is to balance discipline and stretch.

Successful project portfolio management also requires strong social support. For instance, it is well established that software projects depend heavily on the level of trust between designers and managers on the one hand and between customers and future users on the other (Sabherwal 1999). Weinberg suggests that the essence of managing software teams is to create an environment in which designers and engineers become empowered (Weinberg 1986). Best practices have evolved in software firms that require managers to lend expert assistance across project boundaries, e.g., quality assurance through peer-to-peer reviews (Weinberg and Freedman 1982).

Managers in organizations with low alignment and adaptability may seek actionable advice on shaping the organizational context to become ambidextrous. However, thus far research has mainly investigated the antecedents of ambidexterity and the impact of ambidexterity on performance without considering in detail how ambidexterity is developed (Gibson and Birkinshaw 2004; He and Wong 2004). Researchers typically use interviews and surveys to generate snapshot measures of ambidexterity and performance. These studies do not provide insights into how ambidexterity develops within an organization over time or what work activities and practices are entailed (Barley and Kunda 2001). By contrast, collecting and analyzing longitudinal, qualitative data can provide insights into how and why people in organizations act and interact over time (Langley 1999).

The process of building contextual ambidexterity is described as “complex, causally ambiguous, widely dispersed, and quite time-consuming to develop” (Gibson and Birkinshaw 2004, p. 209-210). Through their reports of case studies with multinational organizations, Birkinshaw and Gibson (2004) provide some general lessons on where and how organizations can start developing ambidextrous capabilities: diagnose the organizational context; change key aspects of the context; ensure communication about ambidexterity throughout the organization; consider

contextual and structural ambidexterity; and empower employees throughout the organization to participate. While these lessons serve as a starting point for understanding how to develop ambidexterity, much more is needed to understand how context and managerial practices interact over time and shape each other as organizations strive to become ambidextrous. As we found no empirical studies that describe these processes, we decided to investigate the process of becoming ambidextrous at *TelSoft*.

Contextual Inquiry

We adopt Pettigrew's (1985; 1987) contextualist inquiry framework to investigate the process of becoming ambidextrous. Contextualist inquiry is concerned with understanding how transformation efforts unfold in particular organizational settings focusing on the interactions between content, context, and process (see Figure 1). *Content* refers to the areas being transformed; in this case we focus on managerial practices at *TelSoft* specifically related to project portfolio management. *Context* refers to the environment in which the organization operates as well as the systems, processes, and beliefs within the organization through which ideas for change have to proceed. Focusing here on contextual ambidexterity, we are particularly interested in how the performance management and social support elements of the context shape and are shaped by the process of becoming ambidextrous. Finally, *process* refers to the actions and interactions between various interested parties as they attempt to transform practices. In our case, we focus on the actions and interactions related to building alignment and adaptability within *TelSoft*.

Contextualist inquiry provides a general framing of the study that is well aligned with our focus on building contextual ambidexterity. In addition to the conceptual distinctions between content, context, and process, contextual inquiry combines a process orientation with multiple levels of analysis (Pettigrew 1985, 1987). Within the process orientation, the emphasis is on the interconnectedness of phenomena in historical, present, and future time. In our case, we focus on how past events at *TelSoft* shaped its attempts to build ambidextrous capability and how these events created a basis for moving forward. At different levels of analysis, contextual inquiry draws attention to individuals, groups, the organization at large, and the organization's environment. At *TelSoft* we focus on how individuals engage in project portfolio management, we study how groups of managers interact to become ambidextrous, and we also focus on the wider context of the organization and its interactions with existing and potential customers.

Research Context and Methods

Research Context

TelSoft, a privately held company founded in 1971, customizes geographic information systems (GIS) software for the telecommunications and utility industries. A permanent business unit with approximately 50 members was the focus of our study. For most of its history, *TelSoft*'s client base was dominated by two long-standing, large customers referred to by managers as the "bookends" which kept the company from falling. Advances to software products were driven by change requests from these existing customers. Despite awareness of technological changes in the marketplace, *TelSoft* invested very little in upgrading its software. For instance, even as Microsoft products became the standard for developing Windows-based software applications, software engineers at *TelSoft* used an obsolete technology no longer supported by its vendor.

Although the underlying technological standard of *TelSoft*'s main GIS product was gradually being replaced, *TelSoft* had no plans to comply with new standards.

Prior attempts at radical innovation had gone poorly for *TelSoft*. In the late 1990s, *TelSoft* sensed that the introduction of spatial databases could revolutionize their GIS products. After years of investment, however, the company's CEO chose to terminate the project due to missed deadlines, inadequate functionality, and limited market success. From that point on, management was wary of developing new practices and pursuing new markets and was ordered by the CEO to halt all "speculative development" until further notice.

Action Research

At the time our study began in 2004, *TelSoft* was experiencing severe issues with their main customers: software releases were frequently shipped late, ran over budget, and contained deviations from agreed upon requirements. These issues prompted the management team to focus on innovation, and thus began a two-year action research project initiated in October 2004 by mutual agreement between *TelSoft* and the University Innovation Center (UIC). UIC is a multi-disciplinary research unit within the business school which collaborates closely with industry partners to study end-to-end business process innovation. The first two authors are part of the research group at UIC. The first author had previously been employed at *TelSoft*.

McKay and Marshall (2001) conceptualize action research as containing two concurrent learning cycles. The problem solving cycle addresses the practical concerns of the industry partner while the research cycle addresses the quest for scientific knowledge by the researchers. The challenge for action researchers is to simultaneously navigate both inquiry cycles as well as their interdependencies while attending to potential ethical, initiative, and goal dilemmas (Rapoport 1970). Action research can generate rich data using a mixture of research methods such as participant observation, interviews, document analysis, and surveys; thus supporting research that is both rigorous and relevant. Such characteristics make action research an excellent candidate for studying longitudinal organizational change processes (Pettigrew 1990). There are many forms of action research (Baskerville and Wood-Harper 1998), including canonical action research (Susman and Evered 1978; Davison, Martinsons et al. 2004), action science (Argyris 1985), and soft systems methodology (Checkland 1981, 1990).

This study is based upon collaborative practice research (Mathiassen 2002), a particular form of action research that is characterized by strong collaboration between practitioners and researchers to effect change. The dual goal of the research was 1) to improve software practices at *TelSoft*, and 2) to contribute to scientific knowledge on ambidextrous innovation, in the particular context of software firms. As shown in Figure 2, the research was executed in collaboration between *TelSoft* employees and the UIC research team and organized into a steering committee (SC), a problem solving team (PST), and temporary innovation project teams. The SC involved senior management from *TelSoft* and met two or three times per year as needed to oversee the project. The PST, which consisted of middle-level managers at *TelSoft* and the researchers, was responsible for diagnosing current practices, identifying and prioritizing innovations, and establishing projects to focus on specific innovation areas. In this study, we describe and analyze project portfolio management, the focus of one of the dedicated innovation projects at *TelSoft*. The goals of this project were to formulate, revise, and communicate

TelSoft's innovation strategy; set priorities for software projects; and develop new practices for allocating resources across projects, customers, and products. Consistent with the iterative learning approach typically found in action research studies (Susman and Evered 1978; Davison, Martinsons et al. 2004), this innovation project followed four phases: diagnosing, visioning, intervening, and practicing.

Data Collection

Our data collection occurred through all four phases and used multiple sources of qualitative data as summarized in Table 1. In the *diagnosing* phase, we began by understanding the current problems and practices that required change at *TelSoft*. The primary data sources for this phase were semi-structured interviews with 22 representatives from three major stakeholder groups: software development, internal customers, and external customers. The purpose of the interviews was to gather perceptions of strengths, weaknesses, and opportunities for innovation at *TelSoft*. The first author was the primary interviewer and was frequently joined by one or two other members of the UIC research team. Where possible, these interviews were recorded and later transcribed. In all cases, field notes were taken for later analysis. In addition, we held workshops with employees to confirm our diagnoses, resulting in a comprehensive report prepared by the PST and presented to top management. This report was subsequently used in our data analysis.

The purpose of the *visioning* phase was to create new ways to manage project portfolios at *TelSoft*. Over the course of three meetings, members of the PST established a formal software coordination group (SCG). The group would meet monthly and follow a fixed agenda covering current projects, business opportunities, improvement initiatives, and strategy. These meetings were facilitated by two of the authors. The SCG consisted of four *TelSoft* employees: Division President, Vice President (VP) of Software, Development Manager, and Product Manager as shown in Figure 3. Key data sources during this phase included recordings of the planning meetings, meeting notes, the resulting project plan, and the first two meetings of the SCG.

During the *intervening* phase, we enacted the vision by facilitating several SCG meetings, which were recorded and transcribed. SCG members prepared documents in advance of the meetings and these became important data sources. For the current project review, the Development Manager prepared a spreadsheet listing cost, schedule, and quality assessments for each project. For the review of new opportunities, the Product Manager provided a prioritized list of possible business opportunities, business cases, and maintained a list of future product releases.

During the *practicing* phase, the emerging approaches to project portfolio management became integral parts of the way of operating at *TelSoft*. This phase focused on practicing project portfolio management, evaluating the initiative's impact, and reflecting on what had been learned from this experience. The SCG meetings continued to be a major data source, but we also conducted semi-structured interviews with ten selected employees and customers.

Data Analysis

This iterative nature of action research, in particular, assures that data collection and data analysis are intertwined. Thus, data analysis proceeded across project phases and informed activity in subsequent phases. For example, the research team met during the diagnosing phase to detect patterns emerging from the interview data and to reflect upon what was learned. We

created interim reports and held status meetings with members of the software development group. To address the question of ambidexterity, we coded data reflecting the concepts of performance management, social support, alignment, and adaptability (Gibson and Birkinshaw (2004). These codes are summarized in Table 2. Following a strategy of temporal bracketing (Langley 1999), the data were divided into the phases of diagnosing, visioning, intervening, and practicing. We then analyzed coded data within each phase and extracted the organizational practices that facilitated and balanced alignment and adaptability. Once data for all phases were analyzed, we conducted an analysis across phases to show the mechanisms that caused ambidexterity to increase or decrease.

Results

In this section, we describe *TelSoft*'s process of becoming ambidextrous while innovating project portfolio management. Ambidextrous project portfolio management involves balancing alignment (monitoring existing projects) with adaptability (identifying new projects) by effectively allocating resources across both existing and future projects. In the Diagnosing Section, we assess the degree of alignment and adaptability that existed at *TelSoft*. The three following sections explain how the action research project transformed project portfolio management at *TelSoft*. Following Pettigrew's contextualist approach, we identify aspects of the process, context, and content for each phase of the transformation as summarized in Table 3.

Diagnosis

Context. *TelSoft*'s systems, processes, and beliefs did not support people working in a disciplined fashion to meet or exceed business objectives. Instead, each project manager had considerable autonomy in executing projects and managing the budget. As a result, project outcomes varied considerably depending upon the project manager and resources used. For instance, the *TelSoft* project manager for one of the major clients frequently prioritized producing a high quality product over controlling the triple constraint of successful projects: cost, scope, and time. As a result, his software development projects at *TelSoft* frequently missed deadlines and exceeded the budget. This practice continued, in part, because there were no rewards for either project failure or success. Employees we talked to said that there were few incentives for meeting or exceeding project objectives. Long-time project managers faced no threat of being replaced, and non-management employees had limited opportunities for promotions or increased responsibilities. Incentives were not given to acknowledge exemplary performance, resulting in low employee morale among employees who had not received a raise in three years.

Two other important issues contributed to poor performance management. First, *TelSoft* did not facilitate or encourage employee development. Task assignments were made to use existing expertise rather than to provide opportunities for professional development. Second, there was no systematic process for allocating scarce talent across projects to ensure the company's profitability. *TelSoft*'s Product Manager identified a limited pool of four qualified engineers, who had to be spread across three projects. Rather than allocating resources to maximize profit, *TelSoft* privileged requests from major clients over requests from internal customers, which jeopardized the productivity of the company as a whole.

TelSoft's social support context emphasized the roles that external customers and the Division President played in selecting innovation projects. Existing customers were a major impetus for

process and product innovation at *TelSoft*. In July 2000, *TelSoft* was prompted into process innovation by a major client's requirement for outside certification of its software capability by achieving level 2 on the Software Capability Maturity Model (CMM) (Paulk, Curtis et al. 1993; Paulk, Weber et al. 1995). However, after only one year of engaging in software process improvement, all resources associated with this initiative were abruptly reassigned when the client removed the certification requirement. Subsequently, no organized activity focused on improving management of individual projects or the project portfolio. Although the major customers appreciated *TelSoft*'s responsiveness to their requests, they also wanted *TelSoft* to be more proactive in investing in its products. One customer commented:

“*TelSoft* has a tendency to wait until their major clients tell them they want something before they do something that may make their software better. *TelSoft* should have been working on things on their own for the core product and we shouldn't have to ask for them and pay for them.” (*Client Liaison*, interview)

The Division President was another significant actor setting the direction for product innovations. The VP of Software claimed that the Division President operated based upon hunches, reacting to events emotionally or intuitively. As a result, company-sponsored product innovations were often not aligned well with the market and were, therefore, unsuccessful. In the light of these failed innovation attempts, *TelSoft*'s employees were hesitant to move forward and take risks. The CEO's resulting halt on “speculative development” effectively eliminated enthusiasm around innovation. These failures also made several employees skeptical as the action research study began:

“I did have some skepticism about it initially. I was involved in CMM initially and that was a total flop. It was all about defining the process – not how to implement or follow them. Then all that stuff got forgotten. It wasn't easy to get me fired up about this.” (*Development Manager*, interview)

Despite this drawback, trust and support among the management team members was high. The VP of Software had worked with several of his direct reports for over 15 years and a friendly, comfortable relationship existed. When cost overruns and blown schedules occurred, the VP's displeasure was tempered by a belief that the managers were committed to doing the best job that they could under difficult circumstances.

Content. *TelSoft*'s capability for alignment at this point was fairly positive. Employees rallied behind some project managers to ensure the completion of assigned work, although the strength of alignment varied across project managers. *TelSoft* continued to select projects reactively and lacked a shared vision of a long-term product strategy or optimal project portfolio. In this way, *TelSoft* lacked adaptability. *TelSoft* employees focused on known products and services and were reluctant to invest in changes. There were no systems in place for assessing processes and products and improving them. Although *TelSoft* quickly responded to the needs stated by its customers, it had a dismal track record when it came to responding to the market at large.

Visioning Phase

By June 2005, a new Division President had arrived and was ready to make additional changes. With the UIC's diagnostic report, the SC committed to working with the UIC for the next 18 months to change software practices. Although a number of innovation areas and projects were identified, we focus here on the creation of the software coordination group (SCG) as a mechanism for project portfolio management.

Process. After a series of planning meetings with members of the PST, the research team and VP of Software submitted a detailed plan to the proposed members of the SCG in November 2005. A kick-off meeting was held to ensure that each member understood his role in the group and to allow refinements to the initial agenda covering current projects, business opportunities, improvement initiative, and strategy.

Three important events occurred during the visioning phase. First, the SCG clarified the company's mission, targeted markets, and operating policies. Following the sense-and-respond model (Haeckel 1995, 1999), the SCG collaborated with the CEO to create a "reason for being" statement. The group also articulated its software strategy, which named the organization's main customers, products, and development approach. Nine specific policies contained in the software strategy were contributed by members from all levels of the organization and comprised succinct statements of practices that *TelSoft* members would perform in support of the business objectives. Policies included, for example, requiring approval of the quality assurance department before delivering official releases; and managing each development project with a two-phase funding approach that separated requirements and development activities. After discussion, the SCG reached consensus on the reason for being, software strategy, and policies which collectively became known as *TelSoft*'s software charter.

Second, the SCG agreed to the importance of key performance indicators (KPIs) for assessing current projects. The VP of Software reinstated a practice of all project managers creating weekly status reports. The Development Manager assumed responsibility for collecting the information and distributing it to team members before the SCG meetings began.

Third, the SCG began the practice of reviewing business opportunities. The Product Manager prepared a cost-benefit analysis template for justifying investments. During the first two meetings, he used this template to present two business opportunities for product innovation. The proposed innovations were for enhancements to *TelSoft*'s existing product line and already had the broad support of managers in the room.

Context. The visioning phase saw some improvements to performance management, specifically in the desire to become more disciplined about monitoring and tracking the company's performance objectives. The SCG was committed to the idea of using status information about current software development projects to facilitate project portfolio management. They believed that monitoring KPIs would serve as an "early warning system," allowing them to catch troubled projects early enough in the development cycle to identify corrective actions. At the same time, they hoped that tracking the KPIs would encourage individual project managers to improve project performance. However, contextual factors prevented *TelSoft* from realizing these benefits. The biggest problem was that information

supplied by project managers was frequently in an unsuitable format, incomplete, or submitted too late to be included in discussions:

“He did finally give me the KPIs about five minutes before the meeting, so I didn’t have time to get it together here.” (*Development Manager, SCG #1*)

In another instance, the Development Manager neglected to provide current project information during the second meeting due to his confusion about the meeting time. The SCG tolerated these information quality issues and did not hold the project managers accountable.

Another problem involved the market intelligence underlying business cases presented by the Product Manager. When the SCG members asked questions during his presentation, the Product Manager admitted that he lacked supporting evidence for many of his assumptions. At one point the VP of Software called the estimates in the business case “outrageous.” Despite such problems, the group decided to pursue one of the opportunities presented.

There was also improvement to the social support context, particularly in the Division President’s involvement of more people in strategic planning. The “reason for being” and software strategy were created in a collaborative manner and shared with others in the organization. The commitment to the action research study showed a willingness to break with tradition and consider alternative ways of thinking. With respect to product innovation, the Division President wanted anyone within the organization to be able to make suggestions for new business possibilities. He referred to the Product Manager as being the “gatekeeper of opportunities”:

“He might think it’s the craziest damn idea he ever heard. But I think, to be open to that person that’s come with the idea, [he should] at least give it the credibility of being recorded.” (*Division President, SCG #1*)

The SCG members were open to direction, criticism, and new ways of thinking from the UIC researchers. For instance, the following comment challenged *TelSoft* management to think more deliberately about the level of discipline on projects which were internally funded:

“Do you treat yourself as a customer on equal footing with other customers or do you give yourself bigger freedom in being flexible and democratic in the way that you deal with yourself as a customer? You know, you would never accept from [major client] all that jockeying back and forth.” (*Researcher, SCG #1*)

Content. During the visioning phase, alignment was increased among SCG members through the creation of systems for defining, debating, and modifying performance against business objectives. The software strategy and reason for being were explicit, shared understandings of the criteria that would be used for assessing product innovations. The fixed agenda documented important areas to be discussed each month. Agreement on KPIs specified key business objectives to the project managers at *TelSoft*. Although beliefs were changing among members of

the SCG, it was too early to tell whether others outside the SCG would adapt their behavior based upon these systems.

With respect to adaptability, the SCG struggled to think radically about new markets and uses for their software product. In fact, the business cases proposed were largely in line with old modes of doing business targeting the same markets. Yet, their openness in allowing outsiders from the UIC to challenge existing practices at *TelSoft* and their commitment to monthly meetings were both promising signs that changes to adaptability could take place.

Intervening Phase

The intervening phase began in January 2006, the first meeting in which the Development Manager provided data about current projects using the KPIs. The key characteristic of this phase was the SCG's uncertainty in interpreting information that was brought to its meetings. This uncertainty continued through July 2006, at which point the group began to base decisions more confidently on the data presented.

Process. The SCG spent substantial time during the intervening phase extending practices initiated during the visioning phase. For instance, the software charter was more broadly communicated to employees through workshops and to external customers through a letter from the Division President. The metrics used for current projects were also reported on time, although the data itself could not always be trusted. This revealed a larger deficiency in the systems and tools used for tracking actual project performance against the project plan. To begin addressing this deficiency, the VP of Software developed a tool to retrieve data from the human resource time tracking system automatically and to calculate critical values needed for the KPI report. Finally, the format for presenting business opportunities changed. Instead of presenting detailed business cases justifying a specific software innovation, the Product Manager reported on the list of sales leads being pursued and the status of those leads.

The SCG also introduced periodic customer account reviews as an important new practice during this phase. In these reviews, the project managers reflected on the performance of the most recent releases, identified open issues, and talked about future business opportunities. These more formal reviews held the project managers accountable to the new Division President. At the same time, attending the SCG meetings allowed these project managers to learn first hand about the activities of the SCG and the importance of the KPI data.

Context. During the intervening phase, project managers were held more accountable for project performance, and feedback was used to improve performance. The VP of Software enforced the discipline of weekly written status reports and instituted periodic oral customer account reviews. One noticeable feature during this phase was that the SCG members began to use status information about the projects, despite their limitations, to identify troubled projects. Project managers typically reported that their projects were "going smoothly" even as the evidence suggested otherwise. The VP of Software then accepted responsibility for following up with project managers when there appeared to be discrepancies with the data presented, as evidenced through the following comment:

“I’m going to invite [project manager] to do a [major account] project review at the next meeting and we’ll rake him over because it ain’t going smoothly.” (*VP of Software, SCG #6*)

Although the monthly reporting of KPIs increased awareness of problems, *TelSoft’s* project managers were urged to stretch themselves more to meet project goals. Monthly KPI reports continued to show that most projects missed deadlines and went over budget – even projects that the group had thought were going to be successful:

“I don’t see any corrective action plans coming from the projects when schedules slip. What I see is, you know, ‘this took longer than we thought or we had this issue come up’ ...and then there’s no attempt to make a corrective action plan to get back on track” (*VP of Software, SCG #8*)

As more pressure was placed on the project managers to provide reliable status information, problems with the social support context became apparent. The system of gathering project information required people throughout the organization to work together: the project managers created the overall plans; the development coordinator scheduled developers for specific tasks; developers provided status against those plans; and the project manager adjusted the project plan. The project managers complained that the developers did not provide appropriate estimates. For their part, the project managers did not always adjust their plans to reflect what was learned as the project tasks solidified. Overall, this lack of coordination and communication among the project managers, development coordinator, and developers caused confusion and prevented progress.

Other social support problems also reduced project performance. Projects remained open and incurred cases long after the development work was complete. In some cases, the project manager insisted on personally completing certain aspects of the project rather than trusting others within the department to handle them:

“I haven’t had a chance to read three of the file documents and I typically I don’t like to ship documents that I haven’t had a chance to read and review and edit.” (*Project Manager, SCG #7*)

Content. During the intervening phase, *TelSoft* was more successful with adaptability, as they tried new techniques to attract potential customers. They purchased a new contact management system and began to track sales leads, pursuing customers outside of their traditional markets. Breaking with the tradition of responding to customer requests, *TelSoft* managers proactively planned to revive the failed spatial database software. This product vision was shared with one of the major clients and *TelSoft* requested feedback regarding the most attractive product features. Although the potential for financial sponsorship was uncertain, the *TelSoft* managers felt this exercise would provide useful insights.

Practicing Phase

The practicing phase began in August 2006 and ended in February 2007, when the initial *TelSoft-UIC* collaboration ended. During this phase, the SCG started to focus mostly on practicing

project portfolio management as developed over the previous phases. Also, toward the end of the phase, we interviewed several employees about the impact of the initiative as well as the effectiveness of the SCG.

Process. During the practicing phase, the SCG continued to meet and became an integral part of the management structure at *TelSoft*. There were several areas of improvement: the VP of Software took more ownership of the meetings with less interaction from the researchers; the software charter was posted to the company's website and shared face-to-face with management representatives from the major clients; and a new procedure for conducting post-project reviews was created. Furthermore, the Division President and CEO agreed to continue working with the UIC for another year with the specific focus on developing the project management capabilities of selected employees. Not all changes were positive, however. During this phase, *TelSoft* experienced a critical shortage of sales personnel and loss of market intelligence when one of its two sales people resigned. The poor quality of status information during project reviews also persisted.

Context. The practicing phase was characterized by more critical discussions and questioning during the current project review, again trying to use KPI's to make decisions. There was an increased emphasis on holding project managers accountable:

“So what I've done there is ask major project managers for [major clients] to watch the numbers, ...try to take some responsibility for what time is being charged to their space.” (*VP of Software, SCG #10*)

During this phase, the VP of Software decided to assign a project manager to plan and track this money. The SCG members valued having a historical record of the project data. The group realized that their KPI reports were not the early warning system they had imagined; however, managers were interested in learning from their failures. They informally spoke about lessons learned from each project and also looked forward to incorporating knowledge learned from more formal post-project reviews.

“Four months ago we thought we were going to do a whole lot better with the project, so when we do a post project review on this, one of things we'll be looking at is what kind of things happened [here] (*VP of Software, SCG #15*)

There were still some issues with people at lower levels of the organization not sharing information. For instance, in discussing reasons for a project slipping, the Development Manager indicated that a developer had wasted 15 hours trying to figure something out alone instead of asking his immediate supervisor for assistance.

Content. During this phase, alignment among SCG members continued to grow. The software charter made even non-SCG members aware of the company's strategic direction. However, there remained opportunities for working more coherently across levels of the organization. Adaptability was sustained through the business opportunity reviews, and *TelSoft* decided to invest resources in training project managers.

Discussion

We framed our inquiry into becoming ambidextrous as a contextualist study employing the methodology of action research. The principal advice on building contextual ambidexterity into organizations comes from Birkinshaw and Gibson (2004), who recommend that organizations initially diagnose their context and take specific actions based upon those findings. For organizations, like *TelSoft*, that are diagnosed as weak in performance management but stronger at social support, the recommended action is to focus first on performance management. Performance management can be improved through top-down interventions such as clarifying and communicating the company's strategic goals, focusing on cost reduction and quality, and establishing incentives for performance among unit managers. Such focused attempts at change should be consistently communicated throughout the organization. At the same time, individuals within the organization should be encouraged to increase both alignment and adaptability through specific work practices. Finally, both structural and contextual means of achieving ambidexterity should be considered.

Our action research study incorporated this advice by mapping research activities onto phases in the change process. We began by conducting an initial diagnosis of *TelSoft*'s organizational context and identified the company as fitting the country-club context (i.e., strong social support, weak performance management) in which employees felt comfortable in an informal, collegial working situation but were not pushed to high performance. Given the need to improve performance management, a top-down change initiative was envisioned with the assistance of the UIC researchers. The intervention engaged employees from all levels of the organization to participate on innovation teams. The SCG was formed to facilitate alignment and adaptability with respect to project portfolio management. The fixed agenda of the SCG was a symbol that allowed integration between what was primarily a short-term, alignment based activity (current projects) and a long-term focus on adaptability (new business opportunities). *TelSoft*'s management increased leadership during the practicing phase as the researchers gradually reduced their level of activity and influence.

Although the concept of contextual ambidexterity proved to be a useful guide to our research efforts, the primary limitation of this concept is its ambiguity about the actual process of becoming ambidextrous. The existing literature provides some guidelines for building ambidexterity into organizations (Birkinshaw and Gibson 2004; Gibson and Birkinshaw 2004), but prior studies have not taken a process perspective by tracking either contextual or content changes over time. Consequently, one may assume that there are alternative paths to becoming ambidextrous, but the absence of even one empirically supported process represents a serious gap in theory about ambidexterity.

To compensate for the lack of specificity regarding process, we complemented the insights from contextual ambidexterity with principles of contextualist inquiry (Pettigrew 1985, 1987). Contextualist inquiry offered us an expanded framing that proved compatible with the concept of contextual ambidexterity while at the same time suggesting that content and context interact and mutually shape each other through the process of becoming ambidextrous. In the spirit of building theory from process data and case study research (Eisenhardt 1989; Langley 1999; Eisenhardt and Graebner 2007), we propose a four-phase model for becoming ambidextrous in Table 3. The model incorporates contextualist inquiry's two-dimensional approach by focusing

on the horizontal unfolding of the change process across the four phases of the action research and the interaction between content and context.

The close association between the four phases and the phases of the action research process should not be surprising. Because action research has the dual purpose of guiding organizational change and contributing to scientific knowledge (Rapoport 1970; McKay and Marshall 2001), the resulting theoretical model should closely match the change activities. Hence, we adopted phases consistent with the action research cycle (Susman and Evered 1978) in which each phase is characterized by specific objectives and actions which, in turn, affect context and content in subsequent phases.

Although it is not shown in Table 3, the process is cyclical. This means that changes to practice following one cycle should be diagnosed at the beginning of a second cycle. While our empirical data follow only one cycle to completion, it is clear that *TelSoft* has additional room for improvement in both alignment and adaptability. There is a risk that gains would erode over time without continued cycles, and we also learned about future areas targeted for improvement. For instance, in light of the continuing problems related to status information quality, the VP of Software has recently designed an intervention in which the Development Manager and project managers would meet the day before SCG meetings to ensure that the data presented to the SCG was both accurate and up-to-date. Thus, the cycles could continue indefinitely.

When looking across the horizontal dimension of the model (i.e. the changes in context and content over time), deeper insights become apparent. Table 3 shows that *TelSoft* first dealt with contextual issues (social support and performance management) before realizing improvements to content (alignment and adaptability). In fact, the main emphasis during the visioning phase was not on improving ambidexterity per se, but rather on transforming the context to better facilitate ambidexterity. The visioning phase focused on creating shared beliefs among SCG members with respect to performance management and social support through exercises such as creating a reason-for-being statement, and crafting a software strategy with specific policies. These activities helped integrate the top management team, an important enabler of higher ambidexterity particularly in small firms (Lubatkin, Simsek et al. 2006). However, very few specific actions to change alignment and adaptability were identified initially. Actions during the intervening phase concentrated on investments in context, this time yielding some improvements in adaptability. Finally, the practicing phase saw changes to both context and content. Given that nearly ten months passed before impacts on alignment and adaptability became visible suggests that becoming ambidextrous is a long-term process requiring managerial patience.

Our analysis suggests that transformation of context is not a simple progression of improvements. Although performance management and social support at *TelSoft* both improved across the phases, setbacks were apparent, especially during the intervening phase when social support suffered. Given the seriousness of the issues tackled, we should not expect the road to ambidexterity to be smooth. At *TelSoft*, it was only after both the performance management and social support context had stabilized during the practicing phase that major improvements to alignment were demonstrated.

To the insights drawn from the model, we add a conclusion regarding the importance of choosing initial targets for becoming ambidextrous. Prior research into ambidextrous organizations has considered ambidexterity as a property at the organizational, business unit, and individual levels (Tushman and O'Reilly III 1996; Gibson and Birkinshaw 2004). Our research also finds that the process of becoming ambidextrous can be applied to specific managerial practices within the organization. Managers should carefully select the managerial practices that will drive the innovation process. Identifying and evaluating salient aspects of organizational context is difficult when seen from a general point of view. Instead, approaching organizational context from the vantage point of specific managerial practices creates the backdrop against which sense making about and intervention into the organizational context becomes operational.

TelSoft had a number of management practices which might have been the focus of an innovation effort. For instance, *TelSoft* was also concerned about the management practices throughout the software development process: from managing software requirements elicited from customers, to developing software to match those requirements, to certifying the resulting software product. At *TelSoft*, we identified project portfolio management as a key managerial activity in which the firm's ability to align and adapt was challenged. Although the diagnosis strongly suggested that *TelSoft* also needed to transform management of individual projects, beginning with project portfolio management had a number of advantages. Focusing on project portfolio management required involvement of most developers and managers within the organization and also required critical reflections over the interactions between development, sales, and marketing. In this way, our choice of a target at *TelSoft* allowed more participation on core issues. Alternatively, focusing on transforming management of individual projects could have led to sub-optimizing behaviors that could easily have ignored the organization's overall position in the marketplace. A project focus could also emphasize process innovations over product innovations, again ignoring external market needs. Our conclusion is, therefore, to focus initially on key issues that have wide impact in the organization.

Conclusion

Ambidexterity is increasingly acknowledged as an important organizational capability, yet managers receive limited actionable advice on how it can be developed. To fill this void, we conducted a two-year action research study with *TelSoft*, a small software firm attempting to innovate project portfolio management. Drawing from Birkinshaw and Gibson's arguments concerning contextual ambidexterity (2004) and Pettigrew's contextualist inquiry (1985; 1987), we generated a process model showing how alignment and adaptability practices improved over four phases of managed change: diagnosing, visioning, intervening, and practicing. The model draws attention to the dynamics of change and the interactions between process, context, and the content of planned change.

As with all research, this study has limitations that should be acknowledged and that also have implications for future research. By design, we report from activities within a single organization focusing on the managerial practice of project portfolio management. Such a single-case design does not allow for comparisons across contrasting cases that could further substantiate our findings. For example, the later stages of our model may be sensitive to the antecedent conditions revealed in the diagnostic phase. Other organizations may likely have different initial diagnoses that require the remaining phases to be conducted differently. Although the phases of

the model are sufficiently generic to apply across many organizations, the particular dynamics involving context and content may differ depending on antecedent conditions.

A second limitation derives from our narrow focus on one aspect of improvement at *TelSoft*. Although the selection of project portfolio management over tasks such as project management had a purported benefit, our isolated analysis prevents the generation of insights about learning across different innovation projects. Such research could address questions about the possibility for an organization to become ambidextrous in some ways but not others. Conceivably, lessons learned from one managerial practice might transfer to another practice, yet further research is needed to unravel the process.

Another limitation of the research lies in the restricted conceptualization of organizational context, which rested exclusively on Gibson and Birkinshaw's (2004) original conception. Future research could enrich theory by inducting different aspects of organizational context that influence the process of becoming ambidextrous.

Our findings have direct implications for practicing managers seeking to create more ambidextrous organizations. Our analysis of the change process indicates the value of structuring discrete phases within which various areas of context or content receive emphasis. For example, we discovered the importance of addressing contextual issues early so that the proper conditions (social support, heightened performance management) for improving other capabilities are established. Over time, managers should anticipate such shifts between improvements in context and content.

As action researchers, two of the authors of this paper participated directly as change agents at *TelSoft*. However, the organizational objective of improvement does not necessarily depend on external change agents. Although we believe in the value added by independent researchers and change agents, managers may follow the same process without outside intervention. The analysis provided in this paper can thus serve as a template for manager-led process of becoming ambidextrous.

References

1. Adler, P. S., B. Goldoftas, et al. 1999. Flexibility versus efficiency? A case study of model changeovers in the Toyota production system. *Organ. Sci.* **10**(1) 43-68.
2. Argyris, C. (1985). *Action science*. San Francisco, Jossey-Bass.
3. Barley, S. R. and G. Kunda. 2001. Bringing work back in. *Organ. Sci.* **12**(1) 76-95.
4. Baskerville, R. and T. Wood-Harper. 1998. Diversity in information systems action research methods. *Eur. J. Inform. Systems.* **7**(2) 90-107.
5. Birkinshaw, J. and C. Gibson. 2004. Building ambidexterity into an organization. *Sloan Management Rev.* **45**(4) 47-55.
6. Boehm, B. W. 2002. Get ready for agile methods, with care. *Comput.* **35**(1) 64-69.
7. Boehm, B. W. and R. Turner (2004). *Balancing agility and discipline: A guide for the perplexed*. Boston, Addison-Wesley.
8. Checkland, P. (1981). *Systems thinking, systems practice*. Chichester, Wiley.
9. Checkland, P. (1990). *Soft systems methodology in practice*. Chichester, Wiley.

10. Clark, K. B. and S. C. Wheelwright. 1992. Organizing and leading 'heavyweight' development teams. *California Management Rev.* **34**(3) 9-28.
11. Davison, R. M., M. G. Martinsons, et al. 2004. Principles of canonical action research. *Inform. Systems J.* **14**(1) 65-86.
12. De Reyck, B., Y. Grushka-Cockayne, et al. 2005. The impact of project portfolio management on information technology projects. *Internat. J. Project Management.* **23**(7) 524-537.
13. Eisenhardt, K. 1989. Building theories from case study research. *Acad. Management Rev.* 532-550.
14. Eisenhardt, K. and M. E. Graebner. 2007. Theory building from cases: Opportunities and challenges. *Acad. Management J.* **50**(1) 25-32.
15. Floyd, S. and P. Lane. 2000. Strategizing throughout the organization: Managing role conflict in strategic renewal. *Acad. Management Rev.* **25** 154-177.
16. Ghoshal, S. and C. A. Bartlett. 1994. Linking organizational context and managerial action: The dimensions of quality of management. *Strategic Management J.* **15**(Summer) 91-112.
17. Gibson, C. and J. Birkinshaw. 2004. The antecedents, consequences, and mediating role of organizational ambidexterity. *Acad. Management J.* **47**(2) 209-226.
18. Haeckel, S. 1995. Adaptive enterprise design: The sense-and-respond model. *Planning Rev.* **23**(3) 6-13, 42.
19. Haeckel, S. (1999). Adaptive enterprise: Creating and leading sense-and-respond organizations. Boston, MA, Harvard Business School Press.
20. He, Z.-L. and P.-K. Wong. 2004. Exploration vs. Exploitation: An empirical test of the ambidexterity hypothesis. *Organ. Sci.* **15**(4) 481-494.
21. Hobday, M. 2000. The project-based organisation: An ideal form for managing complex products and systems. *Res. Policy.* **29**(7-8) 871-893.
22. Holmberg, L. and L. Mathiassen. 2001. Survival patterns in fast-moving software organizations. *IEEE Software.* **18**(6) 51-55.
23. Horvat, R. V., I. Rozman, et al. 2000. Managing the complexity of SPI in small companies. *Software Process: Improvement and Practice.* **5**(1) 45-54.
24. Langley, A. 1999. Strategies for theorizing from process data. *Acad. Management Rev.* **24**(4) 691-710.
25. Lee, G., W. DeLone, et al. 2006. Ambidextrous coping strategies in globally distributed software development projects. *Comm. of the ACM.* **49**(10) 35-40.
26. Lee, G., W. DeLone, et al. 2007. Ambidexterity and global IS project success: A theoretical model. *40th Annual Hawaii Internat. Conf. System Sci.* 44-44.
27. Lubatkin, M. H., Z. Simsek, et al. 2006. Ambidexterity and performance in small-to medium-sized firms: The pivotal role of top management team behavioral integration. *J. Management.* **32**(5) 646-672.
28. Markowitz, H. 1952. Portfolio selection. *The Journal of Finance.* **7**(1) 77-91.
29. Mathiassen, L. 2002. Collaborative practice research. *Inform. Tech. & People.* **15**(4) 321-345.
30. Mathiassen, L. and A. M. Vainio. 2007. Dynamic capabilities in small software firms: A sense-and-respond approach. *IEEE Trans. on Engrg. Management.* **54**.
31. McFarlan, F. W. 1981. Portfolio approach to information systems. *Harvard Bus. Rev.* **59**(5) 142-150.

32. McKay, J. and P. Marshall. 2001. The dual imperatives of action research. *Inform. Tech. & People*. **14**(1) 46-59.
33. Napier, N. P., L. Mathiassen, et al. (2006). Negotiating response-ability and repeat-ability in requirements engineering. International Conference on Information Systems, Milwaukee, Wisconsin.
34. Paulk, M., B. Curtis, et al. (1993). Capability maturity model for software, version 1.1. Pittsburgh, PA, Software Engineering Institute.
35. Paulk, M., C. V. Weber, et al., Eds. (1995). The capability maturity model: Guidelines for improving the software process. Sei series in software engineering. Boston, Addison-Wesley.
36. Pettigrew, A. M. (1985). Contextualist research: A natural way to link theory and practice. Doing research that is useful for theory and practice. E. E. Lawler. San Francisco, Jossey-Bass.
37. Pettigrew, A. M. 1987. Context and action in the transformation of the firm. *J. Management Stud.* **24**(6) 649-670.
38. Pettigrew, A. M. 1990. Longitudinal field research on change: Theory and practice. *Organ. Sci.* **1**(3) 267-.
39. Ramesh, B., J. Pries-Heje, et al. (2002). Internet software engineering: A different class of processes. Annals of software engineering, Kluwer Academic Publishers. **14**: 169-195.
40. Rapoport, R. 1970. Three dilemmas in action research. *Human Relations*. **23**(6) 499-513.
41. Sabherwal, R. 1999. The role of trust in outsourced IS development projects. *Comm. of the ACM*. **42**(2) 80-86.
42. Salo, O. and P. Abrahamsson (2005). Integrating agile software development and software process improvement: A longitudinal case study. International Symposium on Empirical Software Engineering.
43. Susman, G. and R. Evered. 1978. An assessment of the scientific merits of action research. *Admin. Sci. Quart.* **23**(4) 582-603.
44. The Standish Group International. (2004). "2004 third quarter research report." from URL http://standishgroup.com/sample_research/PDFpages/q3-spotlight.pdf.
45. Tushman, M. and C. A. O'Reilly III. 1996. Ambidextrous organizations: Managing evolutionary and revolutionary change. *California Management Rev.* **38**(4) 8-30.
46. Vinekar, V., C. W. Slinkman, et al. 2006. Can agile and traditional systems development approaches coexist? An ambidextrous view. *Inform. Systems Management*. **23**(3) 31-42.
47. Weinberg, G. M. (1986). Becoming a technical leader: An organic problem solving approach. New York, Dorset House.
48. Weinberg, G. M. and D. P. Freedman (1982). Handbook of walkthroughs, inspections and technical reviews. Boston, Little Brown and Company.

Figures and Tables

Figure 1: Contextualist Inquiry into Becoming Ambidextrous

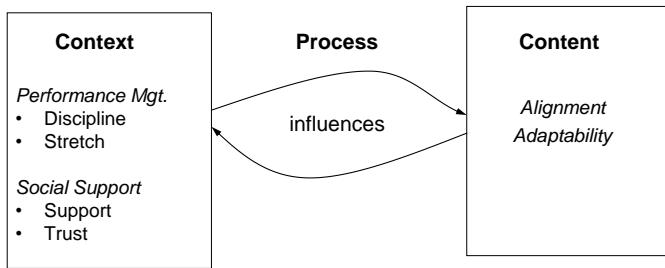


Figure 2: Action Research Organization

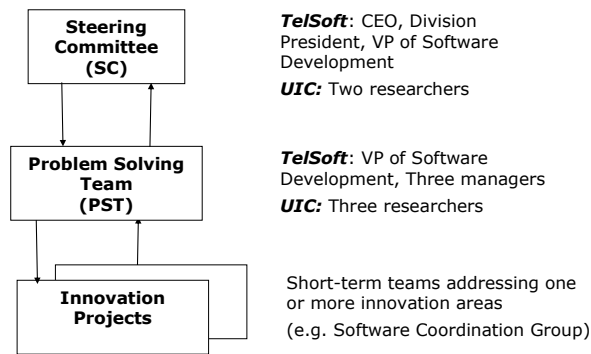
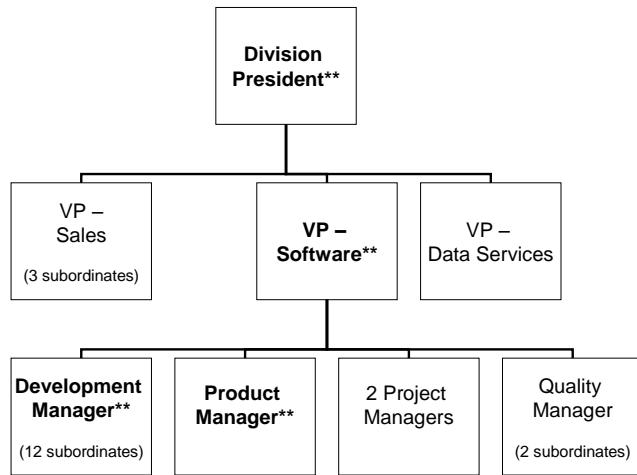


Figure 3: TelSoft Management Organization



** Indicates Software Coordination Group member

Table 1: Data Sources by Project Phases

| Phase | Start Date | Trigger | Data Sources |
|-------------|------------|--|--|
| Diagnosing | 10/2004 | Began diagnosis of software practices at <i>TelSoft</i> | 22 Interviews <ul style="list-style-type: none"> • Software development group • Internal customers • External customers 2 Workshops <ul style="list-style-type: none"> • Software development group • Internal customers Diagnosing report |
| Visioning | 6/14/2005 | Steering Committee commits to eighteen month improvement initiative | 3 SCG Planning meetings <ul style="list-style-type: none"> • Meeting notes and minutes • Project plan and fixed agenda 2 SCG Meetings <ul style="list-style-type: none"> • Transcription of meetings • Reason for being and software strategy • Current project reviews • Business case assessment |
| Intervening | 11/2/2005 | SCG Meeting #3: First complete review of Current Projects and Business Opportunities | 7 SCG Meetings <ul style="list-style-type: none"> • Transcription of meetings • Reason for being and software strategy • Current project reviews • Business opportunities list • Customer account reviews |
| Practicing | 8/9/2006 | SCG Meeting #10: Smooth operation of meetings. Meetings used to facilitate decision making and take action. | 6 SCG Meetings <ul style="list-style-type: none"> • Transcription of meetings • Reason for being and software strategy • Current project reviews • Business opportunities list • Customer account reviews 10 Assessment Interviews |

Table 2: Coding Scheme (Gibson and Birkinshaw 2004)

| <i>Area</i> | <i>Measure</i> |
|------------------------|---|
| Performance Management | <p>The extent to which systems, processes, and beliefs:</p> <ul style="list-style-type: none"> ▪ Encourage people to set challenging and aggressive goals ▪ Create challenges to their people, instead of narrowly defining tasks ▪ Encourage people to focus more on getting their job done well than on getting promoted ▪ Make a point of stretching their people ▪ Reward or punish based on rigorous measurement of performance against goals ▪ Hold people accountable for their performance ▪ Use appraisal feedback to improve people's performance |
| Social Support | <p>The extent to which systems, processes, and beliefs:</p> <ul style="list-style-type: none"> ▪ Prioritize developing subordinates ▪ Give everyone sufficient authority to do their jobs well ▪ Push decisions down to the lowest appropriate level ▪ Give ready access to needed information ▪ Emphasize execution of overall strategy and vision ▪ Base decisions on facts and analysis, not politics ▪ Treat failure as a learning opportunity, not something to be ashamed of ▪ Encourages taking prudent risks ▪ Set realistic goals |
| Alignment | <p>Management systems and practices:</p> <ul style="list-style-type: none"> ▪ Work coherently to support the overall objectives of this organization ▪ Cause the organization to use resources on productive activities ▪ State non-conflicting objectives for specific tasks and projects |
| Adaptability | <p>Management systems and practices:</p> <ul style="list-style-type: none"> ▪ Encourage people to challenge outmoded practices ▪ Allow the organization to respond quickly to changes in markets ▪ Evolve rapidly in response to shifts in business priorities |

Table 3: Becoming Ambidextrous at TelSoft

| <i>CONSTRUCT</i> | <i>PROCESS</i> | | | |
|---|--|---|--|--|
| | <i>Diagnosing</i> | <i>Visioning</i> | <i>Intervening</i> | <i>Practicing</i> |
| <p>CONTEXT:</p> <p>Performance Management</p> | <p>Low</p> <ul style="list-style-type: none"> Project outcomes and processes varied by project manager Few rewards or incentives Limited training opportunities Unsystematic process for resource allocation across projects | <p>Some improvement</p> <ul style="list-style-type: none"> SCG committed to idea of using objective information for decision making Information quality issues | <p>Major improvement</p> <ul style="list-style-type: none"> Beginning to hold project managers accountable for information quality Increased feedback to improve performance | <p>Neutral</p> <ul style="list-style-type: none"> Increased emphasis on holding project managers accountable Historical KPI data considered in decision making Instituted formal post-project reviews |
| <p>CONTEXT:</p> <p>Social Support</p> | <p>Medium</p> <ul style="list-style-type: none"> Selected individuals drive innovation and strategy Hindered by prior failed innovation attempts High trust among long-term employees | <p>Some improvement</p> <ul style="list-style-type: none"> More participative means for directing innovation and setting strategy SCG members accept critique from researchers on improvement | <p>Some setbacks</p> <ul style="list-style-type: none"> Problems coordinating and communicating project tasks among employees Failure to delegate impacts project success | <p>Some improvement</p> <ul style="list-style-type: none"> Continued communication issues about project tasks Emphasis on learning from failed projects |
| <p>CONTENT:</p> <p>Alignment</p> | <p>Medium</p> <ul style="list-style-type: none"> Employees ensure work completed for individual projects Reactive mode for deciding upon whether to initiate projects | <p>Neutral</p> <ul style="list-style-type: none"> SCG fixed agenda and software charter yet to be tested | <p>Neutral</p> <ul style="list-style-type: none"> Social support problems prohibit alignment among employees | <p>Major improvement</p> <ul style="list-style-type: none"> Software charter widely distributed SCG fixed agenda deemed useful for continuing |
| <p>CONTENT:</p> <p>Adaptability</p> | <p>Low</p> <ul style="list-style-type: none"> Focused on known products and services Limited investment in innovating products or processes | <p>Neutral</p> <ul style="list-style-type: none"> Still focused on known products and services | <p>Some improvement</p> <ul style="list-style-type: none"> New techniques implemented for generating leads Product roadmap describes long-term vision for innovation | <p>Some improvement</p> <ul style="list-style-type: none"> Diversity of business opportunity list continues Plans to create roadmap for entire product suite |

Part III: Problem Solving Cycle

This part of the dissertation documents key events from August 2004 through March 2007 designed to understand and improve software practices at *TelSoft*.

Prologue

This study originated from a directed readings course on action research taken with Dr. Lars Mathiassen in Fall 2004. Dr. Roy Johnson also attended these class sessions. We decided to complement the intellectual study of the methodology with actual practice. We explored the idea of trying to establish new relationships with local software companies. However, it soon became clear that my former employer, a small software organization in Atlanta, would provide an optimal fit in terms of geographic proximity and my research interests. After serving as a software engineer at *TelSoft* from September 1999 to August 2003, I left on good terms to pursue graduate education.

Our first challenge was getting the attention of *TelSoft* management. In mid-August 2004, I began contacting my former manager by email and voice mail regarding possible research-industry collaboration. After weeks passed with no response, Dr. Mathiassen became involved in trying to speak to *TelSoft*'s Vice President of Software Development as well as the Division President about this opportunity. Again, there was no response. After much persistence, Dr. Mathiassen finally spoke with *TelSoft*'s CEO by phone. The CEO agreed to a lunch meeting on October 12, 2004 for the GSU researchers to propose a collaboration arrangement.

This "Invitation to Collaboration" meeting was attended by the newly formed research team (Napier, Mathiassen, and Johnson) along with *TelSoft* managers (CEO, VP of Software Development, Division President, and Division Director). The research team presented slides [1] consisting of information about the three team members, expected outcomes, required commitments from each of the partners, and a suggested structure for managing the collaboration. During this presentation, the Division President began sharing concerns about the way requirements were managed at *TelSoft*. After hearing the presentation, the *TelSoft* management took a short break for a private meeting. Upon returning, the CEO announced that *TelSoft* would agree to participate through at least the diagnosing phase of the proposed study. The *TelSoft* managers in attendance would serve as the project's SC. At the end of the diagnosing phase, the SC would assess whether to continue the project.

Thus began a collaboration that led to an SPI initiative that spanned two years and formed the basis for this dissertation. As the research project is organized according to the IDEAL model (McFeeley 1996), this structure is also used in presenting the problem solving cycle, see Figure 4. After *initiating* the project, we *diagnosed* existing strengths, weaknesses, and opportunities with respect to requirements practices. These insights fed two intervention cycles, each focused on *establishing* improvement teams to recommend suggested changes and *acting* upon those suggested changes. The project closed with a *learning* phase which asked identified stakeholders to reflect upon the initiative's impact and the effectiveness of the improvement organization.

Figure 4: Problem Solving Timeline



Chapters 1-5 detail the activities in each phase of the IDEAL model. Appendix A provides a list of problem solving documents generated during the course of the collaboration. Each document is given a unique number which is cross referenced during the description of activities. Appendix B provides the full text of selected project documentation.

Chapter 1: Initiating

The purpose of the initiating phase was to secure commitment from the client to begin work on an improvement area (McFeeley 1996). This section describes the interactions with *TelSoft* required to establish the “mutually acceptable ethical framework” (Rapoport 1970) serving as a foundation for this action research study. Table 1: Initiating Key Dates provides an overview of key dates during the initiating phase at *TelSoft* which are discussed in more detail in the next sections.

Table 1: Initiating Key Dates

| <i>Date</i> | <i>Activity</i> |
|-------------------|---|
| August 13, 2004 | First email sent to software development manager regarding possible collaboration |
| October 12, 2004 | Invitation to Collaboration meeting with TelSoft senior management [1] |
| November 17, 2004 | IRB Approval for Protocol #H05176 “Managing Requirements in Providing and Innovating Software Services” [4] |
| November 19, 2004 | First PST meeting |
| November 29, 2004 | Diagnosing Phase begins: First diagnosing interview of software development manager |

Because the company attributed issues with its processes for discovering, managing, and changing requirements, *TelSoft’s* management initially requested that we focus on the requirements engineering (RE) process. After receiving a verbal commitment from *TelSoft*, several actions followed to firmly establish the project:

1. The research team drafted a project focus document [2] describing the improvement area in more detail. This document is based upon concerns expressed by SC members at the initial meeting.
2. The research team created a memorandum of understanding (MoU) [3] which served as the researcher-client agreement (RCA) (Davison, Martinsons et al. 2004). The MoU documents the roles of the SC and PST, clarifies the dual objectives of contributing to research and practice, and provides an overview of project outcomes. The MoU was refined and agreed to by *TelSoft* in November 2004.
3. I applied for and received Institutional Review Board (IRB) approval [4] for the research study (#H05176).
4. The SC selected the *TelSoft* members of the PST. The first PST meeting was held on November 19, 2004 to begin planning the diagnosing phase.
5. *TelSoft* provided electronic copies of the company’s existing process documentation: 53 files consisting of templates, process flows, guidelines, and example usage. These documents had been created during an earlier attempt to reach SW-CMM level 3 and had remained largely unchanged.

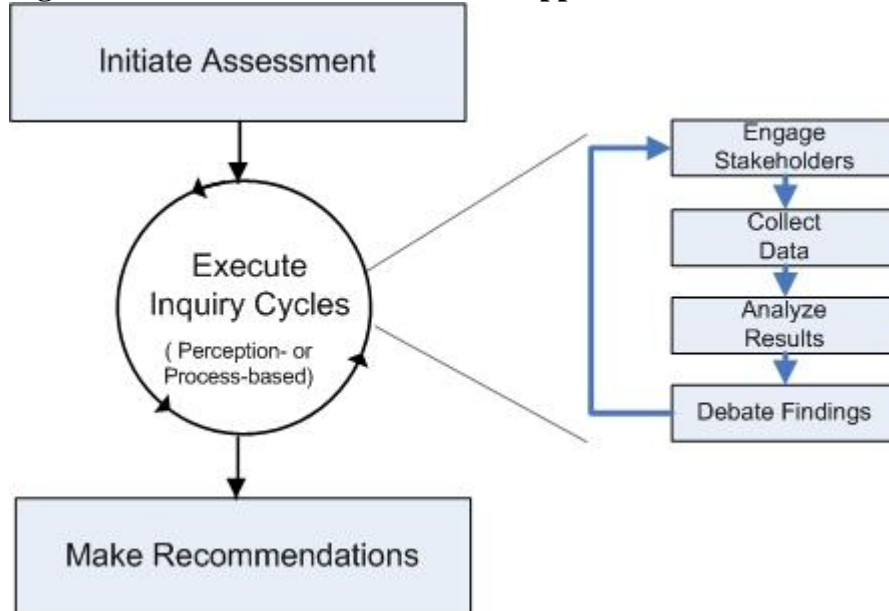
Chapter 2: Diagnosing

The purpose of the diagnosing phase was to understand the current problems and practices within the organization that may need changing. This section describes the data collected between November 2004 and May 2005 to assess *TelSoft's* software practices from the viewpoint of relevant stakeholders (see Table 2: Diagnosing Key Dates). At *TelSoft*, this effort involved 22 semi-structured interviews, two 3-hour workshops, a standardized assessment, and nearly a dozen meetings of the problem solving and research teams.

Table 2: Diagnosing Key Dates

| <i>Date</i> | <i>Activity</i> |
|-------------------|---|
| November 29, 2004 | First diagnosing interview of software development manager |
| January 19, 2005 | Workshop: Software Development Problem Diagnosis [6, 7] |
| January 19, 2005 | New Division President announced |
| March 16, 2005 | SC meeting: Interim Status and first contact with new Division President |
| March 16, 2005 | Workshop: Internal Customers Problem Diagnosis [8, 9] |
| March 30, 2005 | REGPG Assessment completed [11] |
| May 25, 2005 | Last diagnosing interview with external customer |
| May 30, 2005 | First draft of diagnostic report [11] |
| June 1, 2005 | Intervention Cycle 1 begins: First PST meeting to plan improvement strategy |

In thinking about the diagnosing plan, the PST valued the context-specific judgments of the *TelSoft's* employees and customers as well as the general insights that could be provided by standardized assessment methods. To accommodate the desire for both perception-based and process-based assessment, we developed an assessment framework that integrates the two approaches. Our combined approach to RE assessment consists of three steps: initiating the assessment, executing multiple inquiry cycles, and making recommendations based upon the findings (Napier, Mathiassen et al. 2006).

Figure 1: Combined RE Assessment Approach

The assessment was organized as one process-based and three perception-based inquiries. During this time, the PST met as needed (roughly once a month). At these meetings, the research team would present initial findings and describe any issues that arose during data collection. The *TelSoft* members of the PST identified representatives to be interviewed in each of the stakeholder groups and facilitated creation of the group workshops.

For the process-based portion of the assessment, the research team selected the assessment from the book *Requirements Engineering: A Good Practice Guide* (REGPG) (Sommerville and Sawyer 1997). REGPG has been successfully used in both academia and industry. In addition, the research team had access to a REGPG assessment tool (Sommerville and Ransom 2005) that simplified data collection, provided process guidance, ensured accurate calculation of requirements maturity, and automated report generation. The REGPG assessment was conducted during a two hour meeting with members of the PST on March 30, 2005. Participants were provided a written report containing a description of each of the 66 practices and expected benefits to including the practice. Each relevant practice was read aloud and categorized as being standardized, normalized, discretionary, or never followed. For questions the group did not feel prepared to answer, they solicited response from appropriate people after the meeting. The REGPG assessment identified *TelSoft*'s strengths as being in the areas of documenting, eliciting, and describing requirements [10]. Areas for improvement were in analyzing, validating, and managing requirements. The company's overall RE maturity level was assessed at the lowest level: initial.

The perception-based portion of the assessment was designed based upon my prior knowledge of *TelSoft*. We identified three stakeholder groups involved in RE: software development, internal customers, and external customers. The research team created interview guides [5] which asked objective and subjective data on requirements-related documentation and activities that were tailored for each stakeholder group. To ensure participant confidentiality, the research team took

responsibility for data collection and analysis, reporting results at an aggregate level. I was the primary interviewer joined by either Dr. Johnson or Dr. Mathiassen where possible.

The first perception-based inquiry cycle focused on the software development group at *TelSoft*; this group is responsible for interacting with clients to generate a software requirements specification, creating the GIS software based upon these software requirements, evaluating the impact of requirements changes, and ensuring the quality of the resulting software product. We interviewed nine representatives from the software development group (see Table 3: Summary of Diagnosing Interview Sources). The research team analyzed interviewees' responses for similar themes. This analysis produced two key documents: a summary of *TelSoft's* actual requirements process and a list of seventeen potential problem areas. On January 19, 2005, all members of the software development group participated in a three-hour workshop to evaluate this list. For each problem area, workshop participants individually provided an assessment of criticality, feasibility, and priority. These individual responses were then debated and again prioritized in break-out sessions during the workshop. A plenary session was then held in which representatives from each of the break-out groups described their top issues. The primary outcome from this cycle was a prioritized list of problem areas as perceived by the software development group [7, 11].

Table 3: Summary of Diagnosing Interview Sources

| <i>Stakeholder Group</i> | <i>Count</i> | <i>Role</i> |
|----------------------------|--------------|---|
| Software Development Group | 9 | 2 Development Managers 2 Project Managers 2 Software Engineers 2 Systems Analysts 1 Quality Assurance Analyst |
| Internal Customers | 6 | 1 Liaison to Software Group 3 Project Managers 2 Sales Representatives |
| External Customers | 7 | 6 Managers 1 Engineer |
| Total | 22 | |

The second inquiry cycle focused on the internal groups that interacted with the software development group in generating and managing software requirements. The software development group receives requirements from both the marketing organization and an internal production group that uses its GIS software. Once the interviews were completed, the research team again analyzed the interview data for common themes that suggested potential problem areas. On March 16, 2005, the PST sponsored a workshop for validating and prioritizing the 14 identified problem areas. Workshop participants included those interviewed as well as other users within the internal production group. The primary outcome from this cycle was a prioritized list of problem areas as perceived by the internal customers [9, 11].

In the final perception-based inquiry cycle, we interviewed external customers who interacted with *TelSoft* to generate software requirements, request requirements changes, and perform user acceptance testing. The PST selected seven client representatives from three of *TelSoft*'s long-time customers. A new interview guide was created that asked about requirements documentation, requirements management, and process innovation. In this cycle, there was no workshop used as a discussion forum. The customers praised the *TelSoft* personnel for understanding their business, responding promptly to customer requests, and adapting internal practices to client's needs; however, they also identified areas for improvement (e.g. customer relationship management, software release packaging procedures and documentation). The primary outcome from this cycle was a list of strengths and areas for improvement [11].

The research team met to synthesize information from the four inquiry cycles. Although the initial focus was on requirements management practices, the inquiry revealed broader issues that prevented *TelSoft* from effectively satisfying its customers. In total, the research team identified seven improvement areas: software vision management, project portfolio management, software configuration management, customer relations management, requirements management, software quality assurance, and end-user interaction (see Table 4: Identified Improvement Areas for description). In light of these findings, we expanded our research interests to focus more broadly on improving software practices.

Based upon the diagnosing data, we diagnosed *TelSoft* as lacking enterprise agility, the ability to sense opportunities and respond as an intrinsic part of organizational practices (Overby et al., 2006). Enterprise agility is related to existing literature streams on agility (Abrahamsson et al., 2002; Borjesson and Mathiassen, 2005; Dove, 2001; Gunneson, 1997), alertness (Zaheer and Zaheer, 1997), and adaptive enterprises (Haeckel, 1995; Haeckel, 1999). Sensing capability refers to the organization's ability to recognize new business opportunities and technologies as they appear and interpret the impact they might have for the organization (Overby et al., 2006). *TelSoft* was unable to sense new opportunities; instead, the organization was dominated by old ways of thinking. Responding capability refers to the organization's ability to act based upon the information gathered (Overby et al., 2006). Even in those instances when *TelSoft* sensed the need for change, they were not able to respond appropriately; they lacked the capability to effectively adapt and innovate. Seen from the standpoint of sensing capability and responding capability, *TelSoft* needed to combine the ability to sense customer needs and technological and market opportunities while dynamically responding once aware of suitable opportunities. Based upon this assessment, we recommended that *TelSoft* abandon strict command-and-control approaches and use governing principles and defined roles to become a more adaptive enterprise (Haeckel, 1995). Principles from Haeckel's (1995; 1999) sense-and-respond model were chosen to address this issue.

The research team documented these findings in a comprehensive Phase 1 Diagnostic Report which was revised and approved by the PST [11]. The improvement strategy would be addressed through a number of focused and dedicated project teams with clear success criteria and specified deliverables. These project teams would be established, monitored, and coordinated through the PST. The SC would be responsible for approving the overall plans for the improvement.

The SC was kept informed of the PST's activities through periodic status meetings. It is important to note that there were several personnel changes in the SC during this cycle. By the end of Intervention Cycle 1, a new Division President was named. To introduce the new Division President to the initiative, an interim presentation and report was provided on March 16, 2005. The next SC meeting was held on June 9, 2005 to describe the findings and overall recommendations moving forward. Within two weeks of this meeting, SC committed to the improvement strategy and to further collaboration with the research team through December 2006.

Table 4: Identified Improvement Areas

| <i>Area</i> | <i>Issues</i> |
|--------------------------------------|---|
| 1. Software vision management | <i>TelSoft</i> strategy for software development and customer service should be explicated, maintained, and communicated. This provides a value-based foundation for requirements coordination and management that is consistent with <i>TelSoft</i> ' business strategy. |
| 2. Project portfolio management | <i>TelSoft</i> software project portfolio should be managed explicitly and coordinated across internal and external stakeholders. This creates the necessary dynamic capability to respond effectively to different and emerging customer and innovation requests. |
| 3. Software configuration management | <i>TelSoft</i> software configuration management should be improved to ensure consistent and transparent modification and packaging to individual customers. This ensures effective coordination with customers and minimizes adverse effects across projects. |
| 4. Customer relations management | <i>TelSoft</i> should improve its management of customer relations to ensure more symmetric information sharing and proactive expectation and change management. This leads to increased customer satisfaction. |
| 5. Requirements management | <i>TelSoft</i> must improve the transparency and consistency of requirements change management as well as the approach to specify requirements. This lead to improved efficiency, transparency throughout the process, fewer errors, and increased customer satisfaction. |
| 6. Software Quality assurance | <i>TelSoft</i> must build a consistent and systematic software quality assurance process and commit people on all levels to adopt it. This will lead to early detection of errors, improved efficiency, and increased customer satisfaction. |
| 7. End-user interaction | <i>TelSoft</i> must establish closer interaction between software development and end-users. This will lead to improved understanding of requirements and to enhance change management in collaboration with internal and external customers. |

Chapter 3: Intervention Cycle 1

The PST created two separate cycles of establishing and acting. This was done for several reasons. First, the PST wanted to focus on quick, visible, high impact changes to reenergize the organization’s belief in the improvement initiative. There was a cynicism that existed from prior SPI efforts, and we needed to combat that with immediate success. Second, our diagnosis had revealed more problems than could be adequately addressed within a four to six month period. Finally, following the CPR approach (Mathiassen 2002), we believed it was important to actively involve as many people in planning as possible – preferably those that would be responsible for implementing the new actions.

Between June 2005 and August 2005, the PST designed the first cycle of improvement teams (see Table 5: Intervention Cycle 1 Key Dates). As before, the research team took the lead in proposing project teams and prioritizing improvement areas. The research team iterated these plans with the *TelSoft* members of the PST who also identified resources to work on the teams. On September 1, 2005, the PST sponsored a kick-off meeting for all employees in the software development group to present the diagnosing results and describe the upcoming project teams. At the kick-off meeting, Dr. Mathiassen explained the need for a sense-and-respond approach to improvement (Haeckel 1995) and the importance of governing principles. Furthermore, all participants participated in breakout sessions to provide additional input to the proposed improvement teams.

Table 5: Intervention Cycle 1 Key Dates

| <i>Date</i> | <i>Activity</i> |
|-------------------|--|
| June 1, 2005 | PST meeting to plan improvement teams |
| June 9, 2005 | SC status meeting and discussion of project continuation Presented final Diagnostic Report [11] |
| September 1, 2005 | Intervention Cycle 1 Kick-off Meeting [13] |
| October 7, 2005 | Improvement team project plans due [14] |
| November 3, 2005 | First Software Coordination Group (SCG) Meeting [15] SCG assumes responsibility for managerial oversight of project |
| March 15, 2006 | Interim status meeting for Software Development managers [17] |
| March 21, 2006 | Interim status meeting for Software Development staff [17] |
| March 24, 2006 | Deliverables from project teams due to PST [18, 19, 20, 21] |
| March 28, 2006 | Finalized First Wave Report [23] |

| <i>Date</i> | <i>Activity</i> |
|----------------|---|
| March 2006 | Software Charter finalized and included on customer mailings [16, 21] |
| April 18, 2006 | Intervention Cycle 2 begins: Kick-off meeting |

The five improvement teams formed for intervention cycle 1 (also known at *TelSoft* as the First Wave) were software coordination, quality assurance, configuration management, customer relations, and requirements management. The VP of Software Development advised team members to spend no more than four hours every two weeks on the initiative. The PST provided each team with an initial set of objectives and suggested activities based upon the diagnosing stage. Their first task was to evaluate these suggested activities, make modifications, and create a project plan. The teams typically met every two weeks to discuss new ways of operating that would incorporate the suggested activities into *TelSoft's* processes. The VP of Software Development directed the project managers to do the following:

- Use *position papers* as a working document insights, ideas, and proposed decisions resulting from the groups activities.
- Generate brief, high-level *process documents* suitable for existing and potential customers
- Provide simple *templates* that help people follow the processes described in the *process documents*

In most cases, the project managers for the improvement teams created meeting minutes to document key decisions. The research team decided to split up to support the teams. I would try to attend and record all meetings for all the teams. Dr. Mathiassen would support the SCG and requirements management teams. Dr. Johnson agreed to support the configuration management and quality assurance teams. The improvement teams created a number of process documents, position papers, and templates that were reviewed and approved by the PST. The key outcomes for each of the project teams are briefly described below.

Software Coordination

The software coordination group (SCG) was established to address two improvement areas: software vision management and project portfolio management. The SCG consisted of four members: Division President, Vice President of Software Development, Software Development Manager, and Product Manager. Beginning November 2005, the group met monthly and followed a fixed agenda covering status of current projects, business opportunities, improvement initiative, and strategy review. With the inclusion of the improvement initiative on its agenda, the SCG now assumed the role of the SC. To raise awareness of customer relations issues, the SCG periodically invited account managers to provide status on the customer relationship and identify areas of improvement.

As suggested by the sense-and-respond model, the first item of business for the SCG was to clarify the mission of the organization, their targeted markets, and governing principles (Haeckel 1995; Haeckel 1999). The following three items became *TelSoft's* Software Charter [16] and have been shared with employees and customers.

- *Reason for Being.* The reason for being statement succinctly states the organization's mission. The SCG members and CEO were asked to provide a candidate for the division's reason for being by completing the following statement: *TelSoft's* software division exists to.....[fill in action, primary beneficiary, qualifiers, and outcome]." These inputs were collected by the research team and discussed at the second and third SCG meetings. After iteration and discussion, the SCG reached consensus.
- *Software Strategy.* The software strategy articulates the organization's main customers, products, and development approach. As new business opportunities arise, the SCG can use the software strategy to evaluate how closely those opportunities match.
- *Policies.* In general, policies are guiding principles identified by senior management to guide decision-making and drive day-to-day operations (CMMI Product Team 2002). In particular, software policies explicate the organization's governing principles for successful software development. The improvement teams were each asked to propose no more than 5 software policies – brief, enforceable rules stating desired practices that *TelSoft* should adopt. These policies were consolidated by the PST, debated by software development employees, and approved by the SCG.

Quality Assurance

The quality assurance team was designed to address the software quality assurance improvement area. This team wrote position papers on desired standard operating procedures for certification, regression, and acceptance testing. The team also developed a workflow that detailed the internal testing process and produced templates for regression testing [18].

Configuration Management

The configuration management team was designed to address the software configuration management improvement area. This team focused on improving the software release process by ensuring the integrity of the software product which was built and delivered to customers. A key decision here was that responsibility for building the software product would shift to the quality assurance group; quality assurance would become the designated "gatekeeper" for products that got sent to clients. The configuration management team developed a software release specification template [20] for capturing information needed by the software quality assurance department to create the final end product.

Customer Relations

The customer relations team was designed to address four improvement areas: customer relations management, software quality assurance, software configuration management, and end-user interaction. This team started with a lot of energy and ideas, but the project manager got distracted with other work activities, leaving many of the initial plans for the group incomplete. By February 2005, the decision was made to reduce the scope of the project and change project managers. The key activity of the customer relations team was to communicate information about the improvement initiative to the customers that participated in the diagnosing phase and more broadly to *TelSoft's* customer base. This was accomplished through a letter sent by the Division President which also included *TelSoft's* newly developed software charter Software Charter. The group also responded directly to one of the specific customer comments from the diagnosing phase by reinstating weekly status reports to that client [16, 21].

Requirements Management

The requirements management team was designed to improve requirements management, customer relations management, and configuration management. This team was also challenged by problems with the project manager who was temporarily disabled from a car accident early during the project. A replacement was not made, and the team's performance was negatively impacted. This team simplified the functional specification to reduce the number of required sections and created a change control template to be used for all changes to requirements [19].

By February 2006, the PST also recognized the need to better communicate status to the software development group. Although the September 2005 kick-off meeting had engaged the larger group, there had been no further communication about the improvement teams' progress, the Software Charter, or existence of the SCG. To remedy this, I provided a 45-minute status update [17] at the software development manager's meeting on March 15, 2006 and at the software development staff meeting on March 21, 2006.

Lessons Learned

The PST produced the First Wave Summary Report [23] documenting accomplishments from the first intervention cycle. The improvement teams had been asked to provide suggestions for what should be focused on in the second intervention cycle and to provide implementation plans for initiating the proposed actions. These reports made the members reflect upon how they could improve going forward. The PST met on March 30, 2006 to finalize this report and plan the second intervention cycle. The *TelSoft* members of the PST assessed the overall mood regarding improvement to be positive for the employees that were actively involved. Some lessons learned and decisions made:

- *TelSoft*'s website would be updated with the software charter Software Charter as well as a few high-level process documents [22].
- The PST needed to ensure there was a mechanism in place for monitoring and changing the newly created templates and associated process documents.
- The SCG needed to focus more on executing the work outlined in the fixed agenda and less on the mechanics of running the meeting (e.g. metrics provided by project managers). A possible goal could be 90% execution and 10% mechanics. GSU involvement in those meetings would continue for the next several months until such a goal was met.
- The next intervention cycle would have fewer than five improvement teams to economize on *TelSoft*'s limited resources. During Intervention Cycle 1, increased coordination costs were associated with having more teams. For instance, there was some overlap between the work of the quality assurance and configuration management teams that required a joint team meeting and several rounds of email to resolve.
- By February 2006, the PST also recognized the need to better communicate status to the software development group. Although the September 2005 kick-off meeting had engaged the larger group, there had been no further communication about the improvement teams' progress, the Software Charter, or existence of the SCG. To remedy this, I provided a 45-minute status update [17] at the software development manager's meeting on March 15, 2006 and at the software development staff meeting on March 21, 2006.

By the end of Intervention Cycle 1, the composition of the PST changed. Dr. Roy Johnson left the research team and the PST to accept a Fulbright Fellowship in South Africa. One of the *TelSoft* managers on the PST had resigned while another had been fired. The VP of Software Development appointed one of his direct reports to serve on the PST.

Chapter 4: Intervention Cycle 2

This section describes the activities at *TelSoft* between April 2006 and November 2006 to continue making improvements (see Table 6: Intervention Cycle 2 Key Dates). The planning for Intervention Cycle 2 was accomplished at two PST meetings (March 30, 2006 and April 5, 2006). The PST decided to form three improvement teams for Intervention Cycle 2 (also known at *TelSoft* as the Second Wave): customer relations, quality results, and process management. The first two teams continued work from teams in Intervention Cycle 1 while the last team was formed to ensure that process documents would be effectively managed and communicated. On April 18, 2006, the PST sponsored a Kick-off meeting for Intervention Cycle 2 [24]. The objectives of the meeting were to describe key processes and templates created, identify questions regarding the software policies, discuss how implementing these policies would impact employees, and introduce the upcoming improvement teams. The Division President and VP of Software Development played an active role in presenting the software charterSoftware Charter and emphasizing that all employees should be considered “guardians of the policies.”

At the Kick-off meeting [24], the PST provided each team with an initial set of objectives and suggested activities. As before, the first task for the project teams was to provide a draft project plan to the PST by May 1, 2006 [25]. Building upon lessons learned from Intervention Cycle 1, the original plan for Intervention Cycle 2 also included time for an interim status report to the software development group; however, this was cancelled due to scheduling difficulties and the pressing business needs at *TelSoft*. The project teams provided deliverables to the PST by September 29, 2006 for review [26, 27, 28, 29]. The PST met to review materials and provide feedback to the teams. The completion meeting to close Intervention Cycle 2 was held on November 8, 2006 [31].

Table 6: Intervention Cycle 2 Key Dates

| <i>Date</i> | <i>Activity</i> |
|--------------------|--|
| April 18, 2006 | Intervention Cycle 2 Kick-off Meeting [24] |
| May 1, 2006 | Project plans due to PST [25] |
| July 12, 2006 | Planned interim status meeting (Cancelled) |
| September 29, 2006 | Deliverables from project teams due to PST [26, 27, 28, 29] |
| October 17, 2006 | Second Wave Report finalized [30] |
| November 8, 2006 | Learning Phase begins: Intervention Cycle 2 Completion Meeting |

Below, the key outcomes for each of the project teams are briefly described.

Quality Results

Recognizing the overlap in Intervention Cycle 1 between the configuration management and quality assurance teams, the PST decided to combine these efforts during Intervention Cycle 2. This decision had the added benefit of reducing the number of teams which needed to be

managed. The project manager for the quality results team was also the manager of the quality assurance group. The resulting quality results team identified new procedures to enhance internal processes for the software quality assurance unit. More specifically, the group developed guidelines for conducting post-project analysis to determine root cause of problems, cleaning up the software defect database, and improving the efficiency of the regression testing [27].

Customer Relations

The customer relations team was revived during Intervention Cycle 2 by the appointment of a new project manager and an expanded list of members, including the Division President, marketing representative, and customer support personnel. The goals of the team included maintaining contact information for customers and prospects, improving the image of *TelSoft* through customer deliverables, and increasing *TelSoft*'s presence with the customer. By the end of Intervention Cycle 2, the group had agreed to purchase contact management software for sales representatives and management, redesigned the packaging for software releases, and developed guidelines for engaging customers from the proposal through the deployment stage [28].

Process Management

The process management team was the only new team formed during Intervention Cycle 2, and it included employees that had not been active on improvement teams during Intervention Cycle 1. The team's project manager was a member of the software quality assurance group with extensive experience leading projects. The team members included a marketing representative, the software quality assurance department's manager, a software developer, and a customer support representative who was also responsible for updating *TelSoft*'s website. By the end of Intervention Cycle 2, the group had accomplished the following goals [26]:

- Updated *TelSoft*'s website to reflect the most useful information about processes and templates
- Evaluate all existing processes in relation to future use at *TelSoft*
- Created standards for templates and reviewed newly created templates in light of these standards
- Create a plan for process management to be integrated into the software quality assurance department by the end of Intervention Cycle 2. This plan included a fixed agenda for the PST which included oversight of the process management process [29].

Chapter 5: Learning

This section describes the activities at TelSoft between December 2006 and March 2007 to reflected on the impact of the overall change process and assess outcomes (see Table 7: Learning Key Dates). The final assessment of the SPI initiative was designed using the Combined RE assessment framework (Napier, Mathiassen et al. 2006) with a focus on evaluating SPI impact, organization, and perceptions. Concerning SPI impact, our goal was to identify changes in each of the seven improvement areas, the effect of the software policies on day-to-day practice, challenges that occurred in enacting changes, and suggestions for improvement. Concerning the SPI organization, our goal was to assess how effectively the PST, SCG, and improvement teams had managed the SPI effort. Finally, concerning SPI perception, our goal was to determine how different stakeholders perceived the overall value of the SPI effort, their satisfaction with their own level of involvement, and the usefulness of communication methods used.

Table 7: Learning Key Dates

| <i>Date</i> | <i>Activity</i> |
|-------------------|--|
| December 19, 2006 | Assessment Interviews begin |
| February 25, 2007 | Assessment Interviews end |
| March 20, 2007 | Completed administration of employee online questionnaire regarding SPI impact |
| June 19, 2007 | Requirements Engineering Assessment completed |

The final assessment of the SPI initiative was designed using the Combined RE assessment framework (Napier, Mathiassen et al. 2006) with a focus on evaluating SPI impact, organization, and perceptions. Concerning SPI impact, our goal was to identify changes in each of the seven improvement areas, the effect of the software policies on day-to-day practice, challenges that occurred in enacting changes, and suggestions for improvement. Concerning the SPI organization, our goal was to assess how effectively the PST, SCG, and improvement teams had managed the SPI effort. Finally, concerning SPI perception, our goal was to determine how different stakeholders perceived the overall value of the SPI effort, their satisfaction with their own level of involvement, and the usefulness of communication methods used. The resulting assessment plan consisted of two perception-based (interviews and questionnaire) and one process-based (REGPG assessment). We identified four major stakeholder groups: customers, improvement team participants, SPI leadership (SCG & PST), and other software development employees. Table 8 shows the method and content of the inquiry for each stakeholder group.

Table 8: Stakeholder-based View of Learning Assessment

| <i>Inquiry Content</i> | <i>Inquiry Method</i> | <i>Customers</i> | <i>Improvement Team participants</i> | <i>SPI Leadership</i> | <i>Software Development employees</i> |
|------------------------|-------------------------------------|------------------|--------------------------------------|-----------------------|---------------------------------------|
| SPI Impact | Interview Questionnaire REGPG | Yes | Yes | Yes | Yes |
| SPI Organization | Interview | No | Yes | Yes | No |
| SPI Perception | Questionnaire Interview | Yes | Yes | Yes | Yes |

The first perception-based inquiry cycle was based upon ten semi-structured interviews. An interview guide was created based upon the objectives of evaluating SPI impact, SPI organization, and SPI perceptions [32]. Three representatives from two external customers consented to phone interviews. Since a questionnaire would be sent to all employees, the PST selected only seven employees for face-to-face interviews: five managers involved in the PST and SCG plus two developers who had actively participated on improvement teams. Each interview lasted roughly 45 minutes, was audibly recorded, and was later transcribed. The findings were compiled into multiple reports and shared at various levels throughout the organization. The summary of external customer interviews [34] was provided to the PST as well as the primary customer liaison at *TelSoft*. The comments regarding the SCG were presented in an assessment report [35] and discussed during the March 2007 SCG meeting. Other interview comments were combined with data from the questionnaire (described next) as part of an overall SPI impact report [36].

The second inquiry cycle was based on an online questionnaire [33] sent to twenty-five *TelSoft* employees who either reported to the VP of Software Development or had otherwise been involved in the SPI effort. The content of the questionnaire was first created by the research team and then refined and piloted by the PST. The questionnaire asked each individual to assess the impact of the overall initiative, the software policies, and the modified processes and templates. In addition, several open-ended questions allowed the respondent to provide additional detail to explain their answers. Data from the questionnaire played a key role in the overall SPI impact report [36].

The third inquiry cycle relied on the REGPG assessment. The assessment was completed by the VP of Software Development and the QA manager on June 19, 2007, and the assessment results were compared against those from the diagnosing phase [37].

An overall assessment of the usefulness of the initiative has been summarized in Part I, Section 5.2. For detailed results from this phase, see the full text of the following assessment reports in Appendix B:

- B.11 SPI Impact Results Summary
- B.12 Requirements Engineering Assessment Results

Appendix A: Comprehensive List of Problem Solving Documents

| <i>ID</i> | <i>Title</i> | <i>Date</i> | <i>Authors</i> | <i>Description</i> |
|-------------------------|---|-------------|----------------|--|
| <i>Initiating Phase</i> | | | | |
| 1. | Invitation to collaboration slides | 10/10/2004 | Research team | Introduces the research team members, expected project outcomes, and suggested collaboration structure. |
| 2. | Project focus document | 11/17/2004 | Research team | Describes the initial focus of the research based upon concerns of the steering committee. |
| 3. | Memorandum of understanding (MoU) | 11/1/2004 | Research team | Serves as Researcher-client agreement. Documents the roles of steering committee, problem solving team, and researchers. (Full text in Appendix B.1) |
| 4. | Institutional Review Board approval (#H05176) | 11/17/2004 | Napier | Provides approval for use of human subjects in research and informed consent form. (Full text in Appendix B.2) |
| <i>Diagnosing Phase</i> | | | | |
| 5. | Diagnosis interview guides | 12/1/2004 | Research team | Guides developed for leading the initial assessment with software development, internal customers, and external customers. (Full text in Appendix B.3) |
| 6. | Software development workshop preparation materials | 1/19/2005 | Research team | Materials provided consisted of: Agenda, Requirements process comparison summary, list of potential problem areas based upon software development interviews |
| 7. | Software development problem diagnosis final workshop report | 2/16/2005 | Research team | Summarized responses from workshop regarding prioritized problems. |
| 8. | Internal customers problem diagnosis workshop preparation materials | 3/16/2005 | PST | Materials provided consisted of: Agenda, list of potential problem areas based upon internal customer interviews |
| 9. | Internal customers problem diagnosis final workshop report | 3/16/2005 | Research team | Summarized responses from workshop regarding prioritized problems. |

| <i>ID</i> | <i>Title</i> | <i>Date</i> | <i>Authors</i> | <i>Description</i> |
|-----------------------------|---|-------------|-------------------|--|
| 10. | Requirements engineering process assessment results – initial | 3/30/2005 | Research team | Results of performing the REGPG assessment. |
| <i>Intervention Cycle 1</i> | | | | |
| 11. | Phase 1 final diagnostic report | 6/9/2005 | PST | Summary diagnosis of software practices from various viewpoints: software development, internal customers, external customers, and REGPG assessment. (Full text in Appendix B.4) |
| 12. | Phase 1 summary slides | 6/9/2005 | PST | Slides presented to SC identifying problems found and suggested interventions |
| 13. | First Wave Kick-off Meeting Preparation Materials | 9/1/2005 | PST | Agenda, slides, summarizing [11], assigning improvement teams, presenting sense-and-respond model, and 2 Haeckel papers |
| 14. | First Wave Project Plans | 10/7/2005 | Improvement teams | Goals and schedule for the five First Wave improvement teams: quality assurance, configuration management, requirements management, customer relations, and software coordination |
| 15. | SCG Fixed Agenda | 11/2005 | SCG | Fixed agenda defined to guide SCG meetings. Topics covered included current projects, business opportunities, improvement initiative, and strategy review. (Full text in Appendix B.6) |
| 16. | Software charter | 3/2006 | SCG | Reason for Being, Software Strategy, Policies (Full text in Appendix B.5) |
| 17. | Interim status meeting summary slides | 3/15/2006 | PST | During this meeting, the Software Charter was announced, status was provided on implementation of Wave 1 activities, and tentative plans for Wave 2 were discussed |

| <i>ID</i> | <i>Title</i> | <i>Date</i> | <i>Authors</i> | <i>Description</i> |
|-----------------------------|--|-------------|-------------------|---|
| 18. | First wave deliverables – Quality assurance team | 3/28/2006 | Improvement team | Position papers: <ul style="list-style-type: none"> • Maintain stability level • Client data • Enforce standard operating procedures Process document: QA workflow Template: Regression Checklist |
| 19. | First Wave deliverables – Requirements management team | 3/28/2006 | Improvement team | Revised templates: <ul style="list-style-type: none"> • Functional specification • Change control |
| 20. | First Wave deliverables – Configuration management team | 3/28/2006 | Improvement team | Position papers: <ul style="list-style-type: none"> • Document Release Differences • QA Executes Builds • Software Release Specification Process documents: <ul style="list-style-type: none"> • Development and Quality Assurance workflow • Software Release Specification • Document Release Differences • QA Executes Builds Templates <ul style="list-style-type: none"> • Impact Statement • Software Release Specification |
| 21. | First Wave deliverables – Customer relations team | 3/28/2006 | Improvement team | Letter about improvement initiative to customers |
| 22. | Prototype <i>TelSoft</i> website with policies | 3/28/2006 | PST | Created web pages with content from the Software Charter as well as example documents showing how <i>TelSoft</i> supports each policy |
| 23. | First Wave summary report | 3/28/2006 | PST | Compilation of the results from each of the improvement teams, proposed implementation plans for First Wave, and suggested activities for Second Wave |
| <i>Intervention Cycle 2</i> | | | | |
| 24. | Second Wave kick-off meeting preparation materials | 4/18/2006 | PST | Agenda, slides, First Wave processes and templates, Software Charter, description of Second Wave activities |
| 25. | Second Wave project plans | 5/1/2006 | Improvement teams | Goals and schedule for the three Second Wave improvement teams: quality results, customer relations, and process management |

| <i>ID</i> | <i>Title</i> | <i>Date</i> | <i>Authors</i> | <i>Description</i> |
|-----------------------|---|-------------|-------------------|---|
| 26. | Second Wave deliverables – Process management team | 9/29/2006 | Improvement teams | |
| 27. | Second Wave deliverables – Quality results team | 9/29/2006 | Improvement teams | Position papers: <ul style="list-style-type: none"> ▪ PDPR Database Cleanup ▪ QA Archiving of builds and releases ▪ Improve efficiency of QA department ▪ Post Release Quality Review Process documents: <ul style="list-style-type: none"> ▪ PDPR database cleanup ▪ Improve efficiency of QA department ▪ QA archiving of builds and releases ▪ Post release quality review |
| 28. | Second Wave deliverables – Customer relations team | 9/29/2006 | Improvement teams | Policy Statement: <ul style="list-style-type: none"> ▪ <i>TelSoft</i> Email Correspondence Policy Statement Guidelines: <ul style="list-style-type: none"> ▪ Proposals to Include Deployment Support ▪ Deliver Proposals with a Presentation ▪ Management Discussion Points ▪ Customer Engagement |
| 29. | PST Fixed agenda | 9/29/2006 | PST | Full text in Appendix B.7 |
| 30. | Second Wave final report | 10/17/2006 | PST | Full text in Appendix B.8 |
| <i>Learning Phase</i> | | | | |
| 31. | Completion meeting: “Process Improvement: Status & Plans” | 11/8/2006 | PST | Agenda, improvement team reports from Second Wave |
| 32. | Learning interview guide | 12/19/2006 | Research team | Full text in Appendix B.10 |
| 33. | SPI impact questionnaire | 1/15/2007 | Research team | Full text in Appendix B.9 |
| 34. | External customer interview summaries | 1/25/2007 | Research team | Summary of comments from customer interviews (2 from Far Telco, 1 value-added reseller) |
| 35. | SCG assessment report | 1/23/2007 | Research team | Summarized strengths and improvement opportunities based upon interviews with SCG members |

| <i>ID</i> | <i>Title</i> | <i>Date</i> | <i>Authors</i> | <i>Description</i> |
|-----------|---|-------------|----------------|----------------------------|
| 36. | SPI Impact results report | 4/18/2007 | Research team | Full text in Appendix B.11 |
| 37. | Requirements Engineering Assessment results | 7/17/2007 | Research team | Full text in Appendix B.12 |

Appendix B: Problem Solving Cycle Documentation

| | |
|---|-----|
| B.1: Memorandum of Understanding | 183 |
| B.2: Institutional Review Board Approval #H05176 | 188 |
| B.3: Diagnosing Interview Guide | 190 |
| B.4: Phase 1 Diagnostic Report | 192 |
| B.5: Software Charter | 221 |
| B.6: Software Coordination Group Fixed Agenda | 222 |
| B.7: Problem Solving Team Fixed Agenda | 225 |
| B.8: Second Wave Summary Report | 228 |
| B.9: Employee Survey | 238 |
| B.10: Learning Interview Guide | 242 |
| B.11: SPI Impact Results Summary..... | 244 |
| B.12: Requirements Engineering Assessment Results | 255 |

B.1: Memorandum of Understanding

November 1st 2004

The purpose of this Memorandum of Understanding (MoU) is to describe the agreed upon content, structure, and approach to Research & Development (R&D) collaboration between *TelSoft* and Center for Process Innovation, Georgia State University (CEPRIN).

Theme

The theme is “Managing Requirements in Providing and Innovating Software Services at *TelSoft* Engineering”. This includes management of requirements from internal as well as external stakeholders and relates to both Legacy Group and Division software. The collaboration will address the following tasks:

1. Model and assess *TelSoft*'s existing practices and tools as they are applied to requirements elicitation, analysis, documentation and management.
2. Describe all key sources of requirements, the interests of the involved stakeholders, and the different ways in which new requirements are negotiated and used as the basis to define the scope of development projects.
3. Describe existing practices and tools used to continuously manage the scope of projects by tracing project activities and product functionality to the requirements of the project.
4. Identify strengths and weaknesses in current requirements practices as well as opportunities for improvement. Generate new or changed process documentation to assist *TelSoft* future requirements management efforts. (i.e., checklist to identify issues that must be considered and scoped such as client dependencies, assumptions, risk, IP considerations, computing environment, etc)
5. Implement and assess selected improvements in requirements management practices.

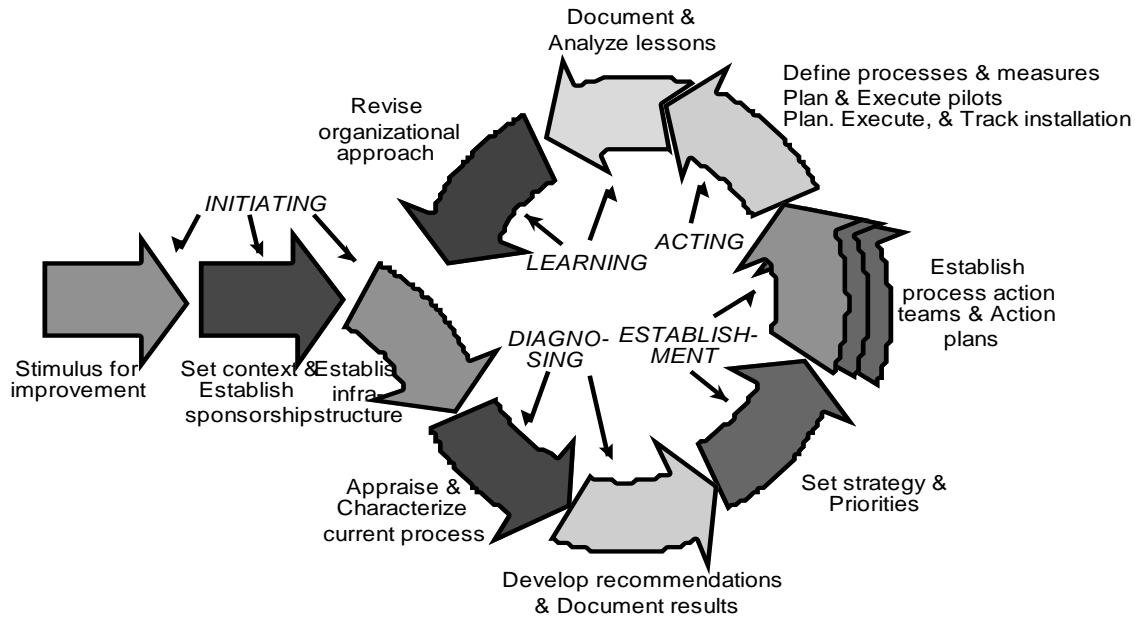
Objectives

The collaboration has the double objective of:

- 1) Improving the quality and productivity of software services at *TelSoft* through enhanced requirements management practices;
- 2) Contributing to research into software requirements management.

Approach

The collaboration proceeds in a stepwise, iterative fashion based on the approach described in the IDEAL model:



The following steps are planned with contents, deliverables, and estimated duration as indicated:

| <i>Step</i> | <i>Contents</i> | <i>Deliverables</i> | <i>Estimated Duration</i> |
|-------------|---|--|---------------------------|
| ID | <ul style="list-style-type: none"> ▪ Initiate collaboration ▪ Diagnose current practices | <ul style="list-style-type: none"> ▪ Model of current practices ▪ Map of key stakeholders and interactions ▪ Assessment of strengths, weaknesses and opportunities | 4 months |
| EAL | <ul style="list-style-type: none"> ▪ Prioritize improvements ▪ Develop and implement new practice ▪ Identify key lessons | <ul style="list-style-type: none"> ▪ Plan for improvement project ▪ Implemented improvement ▪ Lessons from project | 6 months |
| DEAL | <ul style="list-style-type: none"> ▪ Update diagnosis ▪ Prioritize improvements ▪ Develop and implement new practice ▪ Identify key lessons | <ul style="list-style-type: none"> ▪ Updated models and maps ▪ Re-assessment of strengths, weaknesses and opportunities ▪ Plan for improvement project ▪ Implemented improvement ▪ Lessons from project | 6 months |
| DEAL | <ul style="list-style-type: none"> ▪ Update diagnosis ▪ Prioritize improvements ▪ Develop and implement new practice ▪ Identify key lessons | <ul style="list-style-type: none"> ▪ Updated models and maps ▪ Re-assessment of strengths, weaknesses and opportunities ▪ Plan for improvement project ▪ Implemented improvement ▪ Lessons from project | 6 months |

TelSoft and CEPRIN can independently decide to stop the R&D collaboration after each step.

Management

The R&D collaboration is managed by a joint SC (SC) with representatives from *TelSoft* and Lars Mathiassen, Nannette Napier and Roy Johnson representing CEPRIN. Lars Mathiassen coordinates SC meetings to take place 2-4 times a year as needed.

Plan

Step 1 is carried out by a joint problem solving team (PST) consisting of

- EH, *TelSoft*.
- VR, *TelSoft*.
- MB, *TelSoft*.
- Nannette Napier, CEPRIN.
- Lars Mathiassen, CEPRIN.
- Roy Johnson, CEPRIN.

The PST is coordinated by EH and Nannette Napier and it meets routinely every month. Problem solving activities will take place at and between group meetings.

The detailed plan for Step 1 is as follows:

| <i>Start Date</i> | <i>Proposed Duration</i> | <i>Activities</i> | <i>Personnel</i> |
|-------------------|--------------------------|---|--|
| November 1 | 4 weeks | <p>Software Provider View: Understand, analyze, and document requirements management practices at <i>TelSoft</i></p> <p><i>Gather Information</i></p> <ul style="list-style-type: none"> ▪ Collect and review written documentation of practices. ▪ Interview key players at <i>TelSoft</i> regarding the “As-is” process. ▪ Identify key issues related to requirements management from the perspective of <i>TelSoft</i> | Napier with <i>TelSoft</i> personnel |
| Late November | | <p>Workshop #1: Present initial findings and strategize as a group</p> <ul style="list-style-type: none"> ▪ Have we accurately captured practices and key issues? ▪ Which directions and priorities are suggested for further exploration? | PST and representatives from Legacy Group and Division |

| <i>Start Date</i> | <i>Proposed Duration</i> | <i>Activities</i> | <i>Personnel</i> |
|--------------------------|--------------------------|---|---|
| Late November – December | 4 weeks | <p>Internal Software Customer View: Understand how requirements are generated and negotiated</p> <p><i>Gather Information</i></p> <ul style="list-style-type: none"> ▪ Review written documentation on requirements generation and negotiation. ▪ Interview internal software customers about the “As-is” process. ▪ Identify key issues related to requirements management from the perspective of internal software customers. | Napier with <i>TelSoft</i> personnel |
| Late December | | <p>Workshop #2: Present initial findings and strategize as a group</p> <ul style="list-style-type: none"> ▪ Have we accurately captured practices and key issues? ▪ Which directions and priorities are suggested for further exploration? | PST and representatives from Legacy Group, Division, and internal customers |
| January 1 – February 1 | 4 weeks | <p>External Software Customer View: Understand how requirements are generated and negotiated</p> <p><i>Gather Information</i></p> <ul style="list-style-type: none"> ▪ Review written documentation on requirements generation and negotiation. ▪ Interview selected external software customers about the “As-is” process. (Note: We may elect not to involve an external customer. This is TBD.) ▪ Identify key issues related to requirements management from the perspective of external software customers. | Napier with <i>TelSoft</i> personnel |
| Late January | | <p>Workshop #3: Present Information and Strategize as a group</p> <ul style="list-style-type: none"> ▪ Have we accurately captured practices and key issues? ▪ Which directions and priorities are suggested for further exploration? | PST and representatives from Legacy Group, Division, and external customers |

| <i>Start Date</i> | <i>Proposed Duration</i> | <i>Activities</i> | <i>Personnel</i> |
|-------------------|--------------------------|--|------------------|
| February 1 – 28 | 4 weeks | <ul style="list-style-type: none"> ▪ Identify and describe possible improvements. ▪ Develop all deliverables from Step 1. ▪ Capture learning. | PST |
| March 1 | | Workshop # 4: Debate results of Step 1 and outline plans for Step 2. | PST-SC |

Commitments

The R&D collaboration is based on the following commitments:

- **CEPRIN**
 - Help improve requirements management practices at *TelSoft*.
 - Coordinate SC.
 - Develop research contributions based on findings from *TelSoft*.
 - Provide resources to Research Team (Lars Mathiassen, Nannette Napier, and Roy Johnson).

- ***TelSoft***
 - Commit to improving requirements management practices.
 - Provide Research Team access to and cooperation with *TelSoft* employees.
 - Provide resources for *TelSoft* participants in PST.
 - Participate in SC.
 - Provide CEPRIN with funding each quarter of the R&D collaboration starting October 2004. The funding is provided to support CEPRIN through the GSU Foundation.


B.2: Institutional Review Board Approval #H05176

GEORGIA STATE UNIVERSITY INSTITUTIONAL REVIEW BOARD

Mailing Address:
PO Box 3999
Atlanta GA 30302-2999

Phone: 404/443-8674
Fax: 404/654-3838
Website: <http://www.gsu.edu/~irb/>

November 24, 2004



MEMORANDUM

TO: Mathiassen, Lars
eCommerce Institute

FROM: Ann C. Kruger
Institutional Review Board

RE: Approval of Human Subjects Application No. H05176
Type of Review: Expedited
Approval Period: 11/17/2004 – 11/16/2005

The Georgia State University Institutional Review Board reviewed and **approved** your IRB protocol entitled "Managing Requirements in Providing and Innovating Software Services", and your informed consent(s). The approval period is listed above.

Approval periods are one (1) year in length. This protocol **must be renewed at least 30 days before** 11/16/2005 if research is to continue beyond that time frame. Renewal proposals may be resubmitted in abbreviated form.

Any adverse reactions or problems resulting from this investigation must be reported immediately to the University Institutional Review Board. For more information, please visit our website at www.gsu.edu/irb.

RHC:slk

Federal Wide Assurance Number: 00000129

Georgia State University is a unit of the University System of Georgia, is an equal opportunity educational institution and is an equal opportunity/affirmative action employer.

The Management of Requirements in Providing and Innovating Software Services

Consent Form

You have been asked to participate in a research study examining the Management of Software Requirements. You will be interviewed for approximately one to two hours, during which time you will answer questions regarding this topic and focused on the following main subjects: How software requirements currently managed? What is your role in this process? How might this process be improved? What barriers prevent you from making these improvements? The interview is conducted in a semi-structured form allowing me to provide comments as we go along. You will also be asked to share documentation on the company's requirements management process.

There is no physical discomfort or risk associated with this activity. The interview will be recorded to ensure an accurate reflection of your views. If participating in this study causes problems or concerns regarding the information being requested, you can elect to stop at any time. The data that are collected will be handled in the strictest of confidence with access limited to the principal research group. The study will not benefit you directly, but it may lead to a better understanding of the software requirements management process.

The findings will be analyzed and reported for academic purposes only. Your participation will not be identified. Personally identifiable information will be disguised in the academic reporting. You will not be identified by name, nor will your organizational unit be identified.

Information you provide will be kept confidential to the extent allowed by law and not reported to others outside the research project in a way that personally identifies you or your organizational unit. The tape recording of your interview will be stored in a secure location and available only to study personnel.

You may ask questions about this project to the researchers at any time during the project or to the investigator Dr. Lars Mathiasen (404-651-0933), Susan Vogner in the GSU Research Office (404-463-0674) can provide you with general information about the rights of human subjects in research.

A P P R O V E D
FOR 1 YEAR BEGINNING

NOV 17 2004

GSU IRB

B.3: Diagnosing Interview Guide

Development group

The following guide was used for the *TelSoft* personnel who developed software or offered support to the software development process. Your personal view and role regarding the following:

Table 1: Development Group Diagnosing Interview Guide

| <i>Requirements Documents</i> | <i>Requirements Activities</i> |
|---|---|
| <ul style="list-style-type: none"> ▪ Which? ▪ Inputs to you? ▪ Contributions? ▪ Output to whom? | <ul style="list-style-type: none"> ▪ Which? ▪ Interactions? ▪ Collaboration? ▪ Resources? |
| <ul style="list-style-type: none"> ▪ Strengths ▪ Weaknesses ▪ Opportunities | <ul style="list-style-type: none"> ▪ Strengths ▪ Weaknesses ▪ Opportunities |

Internal customers

The following guide was used for the *TelSoft* personnel who used the software as a production tool. Your personal view and role regarding the following:

Table 2: Data Services Diagnosing Interview Guide

| <i>Requirements Activities</i> | <i>Requirements Management</i> |
|--|--|
| <ul style="list-style-type: none"> ▪ Sources and triggering events? ▪ Who do you interact with? ▪ What forms of interaction? ▪ Extent of collaboration with contact? | <ul style="list-style-type: none"> ▪ How are requirements documented? ▪ How are requirements negotiated and decided? ▪ How are requirements changed? ▪ How do you validate deliverables? |
| <ul style="list-style-type: none"> ▪ Strengths ▪ Weaknesses ▪ Opportunities | <ul style="list-style-type: none"> ▪ Strengths ▪ Weaknesses ▪ Opportunities |

The following guide was used for the *TelSoft* personnel supporting sales and marketing. Your personal view and role regarding the following:

Table 3: Marketing Diagnosing Interview Guide

| <i>Product Management</i> | <i>Product Innovation</i> |
|--|--|
| <ul style="list-style-type: none"> ▪ How do you assess market demands? ▪ How do you identify potential customers? ▪ How do you assess product potential? ▪ How do you process feedback from customers? | <ul style="list-style-type: none"> ▪ How do you identify innovations? ▪ How are innovations documented? ▪ How are innovations communicated? ▪ Who do you collaborate with and how? |
| <ul style="list-style-type: none"> ▪ Strengths ▪ Weaknesses ▪ Opportunities | <ul style="list-style-type: none"> ▪ Strengths ▪ Weaknesses ▪ Opportunities |

Additional questions: How difficult/easy is it to sell *TelSoft* products? Is the market receptive?

External customers

Your personal view and role regarding the following:

Table 4: External Customer Diagnosing Interview Guide

| <i>Requirements Activities</i> | <i>Documents</i> | <i>Requirements Management</i> | <i>Process Innovation</i> |
|---|--|---|---|
| <ul style="list-style-type: none"> ▪ Who do you interact with at <i>TelSoft</i>? ▪ What forms of interaction? | <ul style="list-style-type: none"> ▪ How are requirements documented? ▪ How do you validate documents from <i>TelSoft</i>? | <ul style="list-style-type: none"> ▪ How are requirements negotiated and decided? ▪ How are requirements changed? | <ul style="list-style-type: none"> ▪ How well has <i>TelSoft</i> responded to process changes? |
| <ul style="list-style-type: none"> ▪ Strengths ▪ Weaknesses ▪ Opportunities | <ul style="list-style-type: none"> ▪ Strengths ▪ Weaknesses ▪ Opportunities | <ul style="list-style-type: none"> ▪ Strengths ▪ Weaknesses ▪ Opportunities | <ul style="list-style-type: none"> ▪ Strengths ▪ Weaknesses ▪ Opportunities |

Additional questions: What is your role at the company? How long have you worked with *TelSoft*? Given the many competitors, why do you continue to work with *TelSoft*? How would you evaluate the current quality or “state of the art” of *TelSoft* software?

B.4: Phase 1 Diagnostic Report

Executive Summary

The theme is “Managing Requirements in Providing and Innovating Software Services at *TelSoft*”. This includes management of requirements from internal as well as external stakeholders.

The collaboration began in October 2004 with an overall plan described in the Memorandum of Understanding. This report summarizes the results of Step 1: the Initiating and Diagnosing phases of the IDEAL model. The following objectives were addressed during Step 1:

1. Model and assess *TelSoft*'s existing practices and tools as they are applied to requirements elicitation, analysis, documentation and management.
2. Describe all key sources of requirements, the interests of the involved stakeholders, and the different ways in which new requirements are negotiated and used as the basis to define the scope of development projects.
3. Identify strengths and weaknesses in current requirements practices as well as opportunities for improvement.

The assessment has identified many strengths, weaknesses, and opportunities related to requirements management at *TelSoft*. These relate to:

- identification, negotiation, validation, implementation and change of requirements,
- software development, internal customers, as well as external customers,
- resources, approaches, and values in requirements practices,
- operational as well as managerial aspects of requirements practices and
- architecture of the software as well as configuration of the processes.

A feasible approach to turning these insights into improved requirements practices must:

- Align with *TelSoft*'s priorities, traditions, and culture,
- Build on a comprehensive and systemic view of the above aspects of requirements practices,
- Take advantage of possible short-term improvements that can help move requirements practices, and software practices in general, towards higher performance and better customer service, and
- Build sustainable levels of improved practices through appropriate sequencing of efforts.

The following table summarizes potential ideas for action recommended to the Steering Committee:

Table 1: Potential Ideas (arranged by Project)

| <i>Description</i> | <i>Investment</i> |
|--|-------------------|
| Software Coordination – <i>First Wave</i> | |
| Communicate vision: Management team communicates face-to-face the long-term vision for <i>TelSoft</i> software – both internally and externally. This should be followed by periodic revisions and progress reports as the organization moves towards these goals. | Low |
| Publicize commitment: Publicize the reports from Phase 1 of this project. Communicate key findings and how <i>TelSoft</i> plans to address the major problems. Describe level of commitment to Software Process Improvement. | Low |
| Establish Software Coordination Group: Establish a Software Coordination Group that takes the overall responsibility for making priorities, allocating resources, and monitoring <i>TelSoft</i> project portfolio. | Low |
| Software Coordination – <i>Second Wave</i> | |
| Enhance tools: Enhance <i>TelSoft</i> 's suite of tools and processes for project portfolio management. | Medium |
| Quality Assurance – <i>First Wave</i> | |
| Borrow qualified resources: Borrow 2-3 Data Services operators to work in QA for a specific period of time or to help with a specific release. | Low |
| Mandate stability period before shipment: Implement a mandatory stability period between the time a software package is created and the time it is sent to customers. | Low |
| Create accumulated checklist for testing: Update testing scripts to exploit lessons learned from other projects. This prevents old problems from creeping into the software again. | Medium |
| Enforce Standard Operating Procedures: Prioritize a minimal set of standard operating procedures for QA and enforce them. One rule might be to test all changes – particularly core code changes – in all configurations. | Medium |
| Quality Assurance – <i>Second Wave</i> | |
| Analyze root cause: Determine and address root cause of why customer deadlines are not met. | Medium |
| Use a formal process: Use a formal process for eliminating errors (e.g. Requirements standards assessment, Six Sigma) | High |
| Customer Relations – <i>First Wave</i> | |
| Publicize action plan: Communicate to external customers interviewed how Phase 1 issues will be addressed. | Low |
| Standardize <i>TelSoft</i> -Far Telco email interaction: Address Far Telco's specific concerns regarding <i>TelSoft</i> ' email interaction. Clarify their preferred format for documents and ensure that <i>TelSoft</i> personnel consistently use this format. | Low |

| <i>Description</i> | <i>Investment</i> |
|--|-------------------|
| Offer FMT training with every release to Far Telco | Low |
| Weekly conference call with decision makers | Low |
| Prioritize next Local Telco release: Allocate required resources for Quality Assurance and Configuration Management for the next Local Telco release to minimize errors and rebuild client trust. | Medium |
| Visit end-user after deployment: Plan to visit end-users to understand and address their concerns about 1-2 weeks after deployment of each release. | Medium |
| Customer Relations – <i>Second Wave</i> | |
| Solicit end-user input: Solicit input from end users at Far Telco, Local Telco, or Data Services. Create list of enhancements from these visits. Create proposal to address these needs. | Medium |
| Formalize account executive role: Formalize account executive role and responsibilities for each key customer to drive enhanced customer relationship management. | Medium |
| Understand client’s business processes: Solicit more information on customer’s business processes and systems to understand where <i>TelSoft</i> ’ software fits now and in the future. | |
| Configuration Management – <i>First Wave</i> | |
| Utilize software release checklist: Generate checklist for building a software release. Ensure that the correct process is consistently followed. | Low |
| Generate report on differences from previous release: Generate a report with each release that shows the differences between the client’s production version and the new release. Use for input to the Release Notes and Quality Assurance test plans. | Low |
| Configuration Management – <i>Second Wave</i> | |
| Restrict core code changes: Place tighter restrictions on changes to the Core Code (e.g. infrequently scheduled release dates, extensive time for regression test plans, high visibility of changes). | Medium |
| Upgrade configuration management tools and processes: Systematically review and update <i>TelSoft</i> ’ tools and processes for configuration management. | High |
| Requirements Management – <i>First Wave</i> | |
| Enforce change management practices: Review and update change management practices for each key customer and make sure they are followed. | Low |
| Better review of Requirements Documents: Spend more time thoroughly reviewing requirements documents during the design phase. This may also involve getting “the right” people involved in the review. | Low |
| Requirements Management – <i>Second Wave</i> | |
| Establish traceability between requirements and design documents: In the design documents, clearly list which requirements are being satisfied by each part of the design. | Medium |

| <i>Description</i> | <i>Investment</i> |
|---|-------------------|
| Enforce Standard Operating Procedures: Identify the set of tools and processes for change management and enforce them as standard operating procedures across all projects. | Medium |
| Upgrade configuration management tools and processes: Adopt a standard process with state-of-the-art tools for configuration management | High |

Software Development View

The following section provides conclusions from the Requirements Management Workshop held January 19th, 9:30 am – 1:00 pm.

Participants: <Names withheld>

Workshop Process:

- Participants corrected the “Requirements Process Comparison” chart.
- Lars explained the list of “Potential Problem Areas.” Participants added 5 new issues to the list.
- Participants individually assessed each issue on Criticality and Feasibility.
- Participants individually assigned a priority to (at least) the top 5 issues. The highest priority issue was assigned a value of 1.
- Participants were divided into predetermined groups to discuss the issues. The group reached consensus on the top priority issues.
- All participants met to share group findings.

Report Contents

- Complete list of Potential Problem Areas
- Top Issues
- Top Issues by Role
- Software Development Model of Issues

Complete list of Potential Problem Areas

After all interviews were completed, the Research Team (Nannette, Lars, and Roy) created a list of “Potential Problems” (issues 1 through 12 below). During the workshop, each “Potential Problem” was described. Participants added five additional problems to the list. Participants provided an individual assessment of

- Criticality – How important is it to solve this problem? (1=irrelevant, 2=maybe useful, 3=useful, 4=very useful, 5=critical)
- Feasibility – How feasible is it to solve this problem? (1=impossible, 2=difficult, 3=possible, 4=easy, 5=no problem)
- Priority – What are the top problems that should be addressed?

Participants were divided into three predetermined groups to discuss the issues. The group reached consensus on the top priority issues.

Table 2 summarizes the data collected during this process.

- **Description:** Complete text of the “Potential Problem” as shown to the workshop participants
- **Occurrence in Group Top Five:** Number of times a group prioritized this as a top five problem

- **Occurrence in Individual Top Five:** Number of times an individual prioritized this as a top five problem
- **Average Criticality:** Average criticality score assigned to this problem by individuals
- **Average Feasibility:** Average feasibility score assigned this problem by individuals

Table 2: Potential Problems: Software Development View

| <i>ID</i> | <i>Description</i> | <i>Count in Group Top Five</i> | <i>Count in Individual Top Five</i> | <i>Average Criticality</i> | <i>Average Feasibility</i> |
|-----------|---|--------------------------------|-------------------------------------|----------------------------|----------------------------|
| 1 | <i>Customer Variation</i> There are considerable variations in requirements management and quality assurance practices across customers; innovations are driven by customers or ad-hoc initiatives; these innovations are not prioritized or coordinated. | 0 | 2 | 2.69 | 1.85 |
| 2 | <i>Process vs. Practice</i> TelSoft described requirements management process is considerably different from practices; the ongoing maintenance and innovation of the described processes is not institutionalized. | 0 | 2 | 3.54 | 2.46 |
| 3 | <i>QA Disintegration</i> Quality assurance practices are insufficiently integrated with development practices; quality assurance is more like a formal administrative procedure than a facilitator of requirements and software quality. | 3 | 9 | 3.92 | 2.85 |

| <i>ID</i> | <i>Description</i> | <i>Count in Group Top Five</i> | <i>Count in Individual Top Five</i> | <i>Average Criticality</i> | <i>Average Feasibility</i> |
|-----------|---|--------------------------------|-------------------------------------|----------------------------|----------------------------|
| 4 | <i>Documentation Standards</i> Documentation standards are practices vary; there are considerable variations in style and level of detail across authors; the most appropriate documentation form is not necessarily chosen to effectively target documentation users; some documentation standards do not fit current needs. | 1 | 4 | 2.92 | 2.46 |
| 5 | <i>Change Management</i> Requirements changes are not addressed in a systematic fashion; documents are as a result not kept updated and consistent; these practices create problems for some stakeholders. | 2 | 8 | 3.62 | 2.92 |
| 6 | <i>Centralized vs. Decentralized</i> Key activities are centralized or decentralized in questionable ways; requirements identification and approval is in some cases highly centralized; allocation of resources is decentralized. | 0 | 1 | 1.62 | 3.54 |
| 7 | <i>Customer-driven innovation</i> Software product innovation and development is driven by customer requests in a rather ad-hoc fashion; this practice threatens the long-term market value of Byes software products. | 2 | 5 | 3.19 | 2.15 |
| 8 | <i>Outdated tools</i> Tools and methodologies for requirements management are not state-of-the art; there are no procedures or responsibilities in place to facilitate improvements. | 0 | 3 | 3.12 | 3.17 |

| <i>ID</i> | <i>Description</i> | <i>Count in Group Top Five</i> | <i>Count in Individual Top Five</i> | <i>Average Criticality</i> | <i>Average Feasibility</i> |
|-----------|--|--------------------------------|-------------------------------------|----------------------------|----------------------------|
| 9 | <i>Inconsistent Signoff</i> Sign-off of requirements happen in many different ways both in relation to customers and internally at <i>TelSoft</i> . | 0 | 1 | 2.42 | 2.58 |
| 10 | <i>No Req. Baseline</i> No commonly agreed baseline of requirements is established, documented or maintained to help coordinate implementation efforts and assess and manage changes. | 1 | 3 | 2.12 | 3.69 |
| 11 | <i>Ad Hoc Review</i> Review of requirements is often performed in ad-hoc fashion where reviewers are unprepared and critique is not systematically fed back into the requirements process. | 2 | 7 | 3.96 | 2.81 |
| 12 | <i>Avoid Confrontation</i> Conflicts related to requirements implementation and quality are often avoided rather than used as basis for innovation. | 0 | 3 | 3.23 | 2.54 |
| 13 | <i>Lack Time</i> There is not enough time to do a good job in software development (time) | 1 | 9 | 3.96 | 1.92 |
| 14 | <i>Resource Allocation</i> QA, core development have difficulties in prioritizing tasks and requests across projects (resources) | 1 | 6 | 3.77 | 2.35 |
| 15 | <i>BA SW Access</i> BA become involved in requirements tasks where they don't know or have access to the software (training) | 1 | 4 | 3.75 | 3.75 |

| <i>ID</i> | <i>Description</i> | <i>Count in Group Top Five</i> | <i>Count in Individual Top Five</i> | <i>Average Criticality</i> | <i>Average Feasibility</i> |
|-----------|---|--|---|--------------------------------|--------------------------------|
| 16 | <i>Lack Domain Expertise</i> TelSoft has limited expertise in customers' business domains (training) | 1 | 4 | 3.92 | 2.69 |
| 17 | <i>Insufficient Sparring</i> Insufficient sparring with customers on feasibility of requirements and solutions. | 0 | 0 | 3.33 | 2.63 |

Top Issues

Table 3 shows issues that received a high priority from several groups or individuals. An issue was included below if

- (a) 2 or more groups ranked the issue in the Top Five and/or
- (b) 6 or more individuals ranked the issue in the Top Five.

Table 3: Top Issues: Software Development View

| <i>ID</i> | <i>Description</i> | <i>Group Count (Max=3)</i> | <i>Individual Count (Max=13)</i> |
|-----------|---|--------------------------------|--------------------------------------|
| 3 | <i>QA Disintegration</i> Quality assurance practices are insufficiently integrated with development practices; quality assurance is more like a formal administrative procedure than a facilitator of requirement and software quality. | 3 | 9 |
| 5 | <i>Change Management</i> Requirements changes are not addressed in a systematic fashion; documents are as a result not kept updated and consistent; these practices create problems for some stakeholders. | 2 | 8 |
| 7 | <i>Customer-driven Innovation</i> Software product innovation and development is driven by customer requests in a rather ad-hoc fashion; this practice threatens the long-term market value of <i>TelSoft</i> software products. | 2 | 5 |
| 11 | <i>Ad Hoc Review</i> Review of requirements is often performed in an ad-hoc fashion where reviewers are unprepared and critique is not systematically fed back into the requirements process. | 2 | 7 |
| 13 | <i>Lack Time</i> There is not enough time to do a good job in software development. (Time) | 1 | 9 |
| 14 | <i>Resource Allocation</i> QA and core development have difficulties in prioritizing tasks and requests across projects. (Resource) | 1 | 6 |

Top Issues by Roles

Responses were grouped by role to determine whether priorities and needs differed. Note: Responses from the one architect were not grouped since he did not seem to fit any of the categories.

Priority Assignments

Table 4 looks at the “Priority” column. The chart only reports on issues that were ranked HI by at least one group of stakeholders.

- HI: the majority of the people in the group ranked the issue in the top 5
- LO: at least one person in the group ranked the issue in the top 5
- – : no one in the group ranked the issue in the top 5

Table 4: Role-based view of top priority issues

| Issue | Description | Quality Assurance (2 people) | Management (6 people) | Development (2 people) | Business Analyst (2 people) |
|-------|----------------------------|------------------------------|-----------------------|------------------------|-----------------------------|
| 3 | QA Disintegration | HI | HI | LO | HI |
| 5 | Change Management | – | HI | LO | LO |
| 7 | Customer-driven innovation | – | LO | HI | – |
| 11 | Ad Hoc Review | LO | LO | LO | HI |
| 13 | Lack Time | HI | LO | HI | HI |
| 14 | Resource Allocation | LO | LO | HI | – |
| 15 | BA SW access | – | LO | HI | LO |

Criticality Assignments

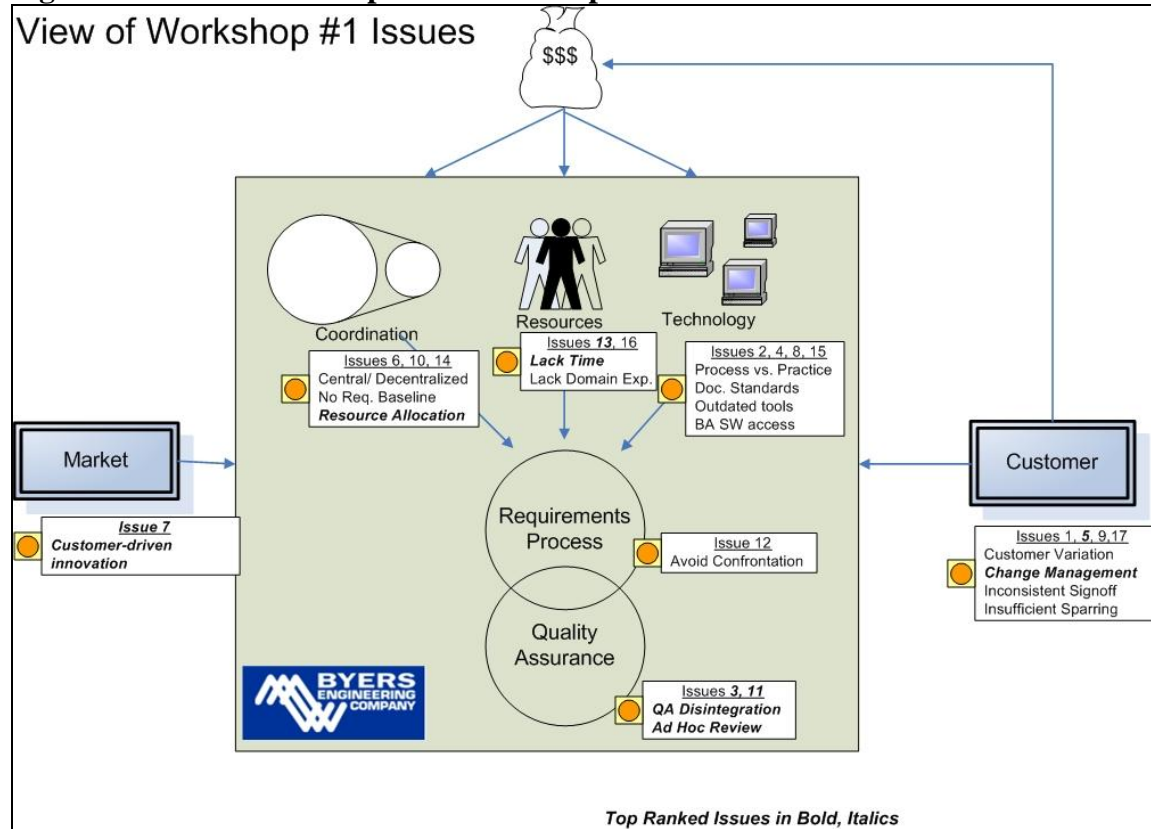
Table 5 reports the average “Criticality” score (1=irrelevant, 2=maybe useful, 3=useful, 4=very useful, 5=critical) by role. The table only shows those issues where there were differences among stakeholder groups.

Table 5: Role-based view of critical issues

| Issue | Description | Quality Assurance (2 people) | Management (6 people) | Development (2 people) | Business Analyst (2 people) |
|-------|---------------------|------------------------------|-----------------------|------------------------|-----------------------------|
| 8 | Outdated Tools | 3.5 | 2.8 | 4.5 | 3.5 |
| 12 | Avoid Confrontation | 3.5 | 3.0 | 4.0 | 3.5 |
| 13 | Lack Time | 5.0 | 3.8 | 3.0 | 5.0 |
| 14 | Resource Allocation | 4.5 | 3.5 | 3.0 | 4.5 |
| 15 | BA SW Access | 2.5 | 4.3 | 3.0 | 4.0 |

Software Development Model of Issues

Figure 1: Software Development Workshop Issues



Internal Customer View

Requirements Management Workshop #2 Report
Held March 16th, 10:00 am – 12:30 pm

Theme: Problem Areas in Requirements Management at *TelSoft*

Participants: *Names Withheld*

Workshop Process:

- Lars explained the list of “Potential Problem Areas.” Participants divided one of our original issues into two separate issues. Therefore, there were a total of 14 potential problems to assess.
- Participants individually assessed each issue on Criticality and Feasibility.
- Participants individually assigned a priority to (at least) the top 5 issues. The highest priority issue was assigned a value of 1.
- Participants were divided into predetermined groups to discuss the issues. Each group reached consensus on the top priority issues.
- All participants met to share group findings.

Report Contents

- Complete list of Potential Problem Areas
- Top Issues
- Top Issues by Function
- Internal Customer Model of Issues

Complete list of Potential Problem Areas

After all interviews were completed, the Research Team (Nannette, Lars, and Roy) created a list of “Potential Problems” (issues 1 through 13 below). During the workshop, each “Potential Problem” was described. Participants decided to split part of the original formulation of issue #8 into a new issue – #14. Participants provided an individual assessment of

- Criticality – How important is it to solve this problem? (1=irrelevant, 2=maybe useful, 3=useful, 4=very useful, 5=critical)
- Feasibility – How feasible is it to solve this problem? (1=impossible, 2=difficult, 3=possible, 4=easy, 5=no problem)
- Priority – What are the top problems that should be addressed?

Participants were divided into two predetermined groups to discuss the issues. The group reached consensus on the top priority issues.

Table 6 summarizes the data collected during this process.

- **Description:** Complete text of the “Potential Problem” as shown to the workshop participants

- **Occurrence in Group Top Five:** Number of times a group prioritized this as a top five problem
- **Occurrence in Individual Top Five:** Number of times an individual prioritized this as a top five problem
- **Average Criticality:** Average criticality score assigned to this problem by individuals
- **Average Feasibility:** Average feasibility score assigned this problem by individuals

Note: Results from 7 respondents were used for individual rankings. Results from 9 respondents were used for the average criticality and average feasibility.

Table 6: Potential Problems: Internal Customer View

| <i>ID</i> | <i>Description</i> | <i>Count in Group Top Five</i> | <i>Count in Individual Top Five</i> | <i>Average Criticality</i> | <i>Average Feasibility</i> |
|-----------|---|--------------------------------|-------------------------------------|----------------------------|----------------------------|
| 1 | <i>Unsystematic early capture of requirements</i> <i>TelSoft</i> representatives (e.g. Sales and Marketing) often capture client requirements in unsystematic, non-documented ways as basis for later interaction with other <i>TelSoft</i> stakeholders. | 1 | 3 | 4.33 | 2.89 |
| 2 | <i>Market and technology opportunities not translated into requirements</i> <i>TelSoft</i> stakeholders are aware of opportunities that would enhance the marketability of <i>TelSoft</i> software (e.g. servicing energy clients, adding drawing capability to spatial product). These opportunities are not translated into software requirements even though such innovations could enhance customer interaction and services. | 1 | 1 | 4.00 | 2.67 |
| 3 | <i>Complex chain of requirements communication</i> There are several <i>TelSoft</i> stakeholders (e.g. Sales, Project Management, Business Analysts, Software Developers) involved in the requirements process. That leads to many interpretations and necessary translations, each introducing new sources of error. | 1 | 2 | 3.67 | 2.67 |

| <i>ID</i> | <i>Description</i> | <i>Count in Group Top Five</i> | <i>Count in Individual Top Five</i> | <i>Average Criticality</i> | <i>Average Feasibility</i> |
|-----------|--|--------------------------------|-------------------------------------|----------------------------|----------------------------|
| 4 | <i>Changes not systematically communicated to Data Services operators</i> Procedural and software changes are not systematically communicated to Data Services operators across the organization. | 2 | 4 | 4.33 | 3.89 |
| 5 | <i>Problematic requirements collaboration between Sales and Data Services</i> Sales desires more timely, professional interaction with Data Services to enhance project estimation and planning. Data Services desires more detailed information from Sales regarding Client requirements to support the bid process. | 0 | 1 | 3.89 | 2.78 |
| 6 | <i>Varying contribution of Source To Target Matrix</i> There are different opinions about the role and value of the Source To Target Matrix. The intention is to create this document during the bid process to price the project. However, most Clients spent little time specifying requirements upfront, and they tend to primarily present good, standard cases of data. That leads to inaccurate pricing. | 1 | 1 | 4.33 | 2.11 |
| 7 | <i>Data Services pricing squeezes requirements implementation</i> The pricing of Data Services does not permit enough resources for implementation of software requirements. | 1 | 2 | 4.33 | 2.44 |
| 8 | <i>Software often rejected by Data Services</i> Data Services frequently rejects software from <i>TelSoft</i> Development due to insufficient quality assurance practices. | 2 | 5 | 4.56 | 3.33 |

| <i>ID</i> | <i>Description</i> | <i>Count in Group Top Five</i> | <i>Count in Individual Top Five</i> | <i>Average Criticality</i> | <i>Average Feasibility</i> |
|-----------|--|--------------------------------|-------------------------------------|----------------------------|----------------------------|
| 9 | <p><i>Development not aligned with business volume</i> Although internal customers generate the largest business volume, Software Development focuses on external customers. Software Development organization and management are remnants of previous traditions rather than effective responses to current business needs (e.g. Data Services, software services, and software innovations).</p> | 1 | 6 | 4.22 | 2.89 |
| 10 | <p><i>Deadlines not met for Data Services software</i> Deadlines for delivering software to Data Services are often not met. Ad-hoc software management practices jeopardize the profitability of Data Services projects.</p> | 2 | 5 | 4.67 | 2.56 |
| 11 | <p><i>Data Services pays for development errors</i> The difference in nature and content between external contracts and internal contracts implies that Data Services pays for software development errors.</p> | 1 | 1 | 3.00 | 3.44 |
| 12 | <p><i>Unsystematic error tracking</i> There is no systematic process for tracking errors in requirements and software related to Data Services. While software deficiencies are known, they are not tracked, root causes are not determined, and appropriate interventions are not enacted.</p> | 0 | 1 | 4.00 | 3.11 |

| <i>ID</i> | <i>Description</i> | <i>Count in Group Top Five</i> | <i>Count in Individual Top Five</i> | <i>Average Criticality</i> | <i>Average Feasibility</i> |
|-----------|--|--------------------------------|-------------------------------------|----------------------------|----------------------------|
| 13 | <i>Data Services not exploited for process and product innovation</i> Knowledgeable Data Services employees are rarely consulted as a source for innovating Data Services –Software Development interactions or the legacy software. | 0 | 0 | 3.56 | 3.56 |
| 14 | <i>Data Services product rejection</i> Data Services rejects roughly 50% of the work done by subcontractors. Client rejects roughly 25% of the exchanges completed by Data Services. | 0 | 2 | 3.86 | 3.43 |

Top Issues

Table 7 shows issues that received a high priority from several groups or individuals. An issue was included below if

- (a) Both groups ranked the issue in the Top Five and/or
- (b) Average Criticality ranked by the 9 individual respondents is greater than 4.00.

Note: Frequency of individuals that ranked this item in Top Five is included for informational purposes only.

Table 7: Top Issues: Internal Customer View

| ID | Description | Group Count (Max=2) | Individual Count (Max=7) | Average Criticality |
|----|--|------------------------|-----------------------------|---------------------|
| 1 | <i>Unsystematic early capture of requirements</i> TelSoft representatives (e.g. Sales and Marketing) often capture client requirements in unsystematic, non-documented ways as basis for later interaction with other TelSoft stakeholders. | 1 | 3 | 4.33 |
| 4 | <i>Changes not systematically communicated to Data Services operators</i> Procedural and software changes are not systematically communicated to Data Services operators across the organization. | 2 | 4 | 4.33 |
| 6 | <i>Varying contribution of Source To Target Matrix</i> There are different opinions about the role and value of the Source To Target Matrix. The intention is to create this document during the bid process to price the project. However, most Clients spent little time specifying requirements upfront, and they tend to primarily present good, standard cases of data. That leads to inaccurate pricing. | 1 | 1 | 4.33 |
| 7 | <i>Data Services pricing squeezes requirements implementation</i> The pricing of Data Services does not permit enough resources for implementation of software requirements. | 1 | 2 | 4.33 |
| 8 | <i>Software often rejected by Data Services</i> Data Services frequently rejects software from TelSoft Development due to insufficient quality assurance practices. | 2 | 5 | 4.56 |
| 10 | <i>Deadlines not met for Data Services software</i> Deadlines for delivering software to Data Services are often not met. Ad-hoc software management practices jeopardize the profitability of Data Services projects. | 2 | 5 | 4.67 |

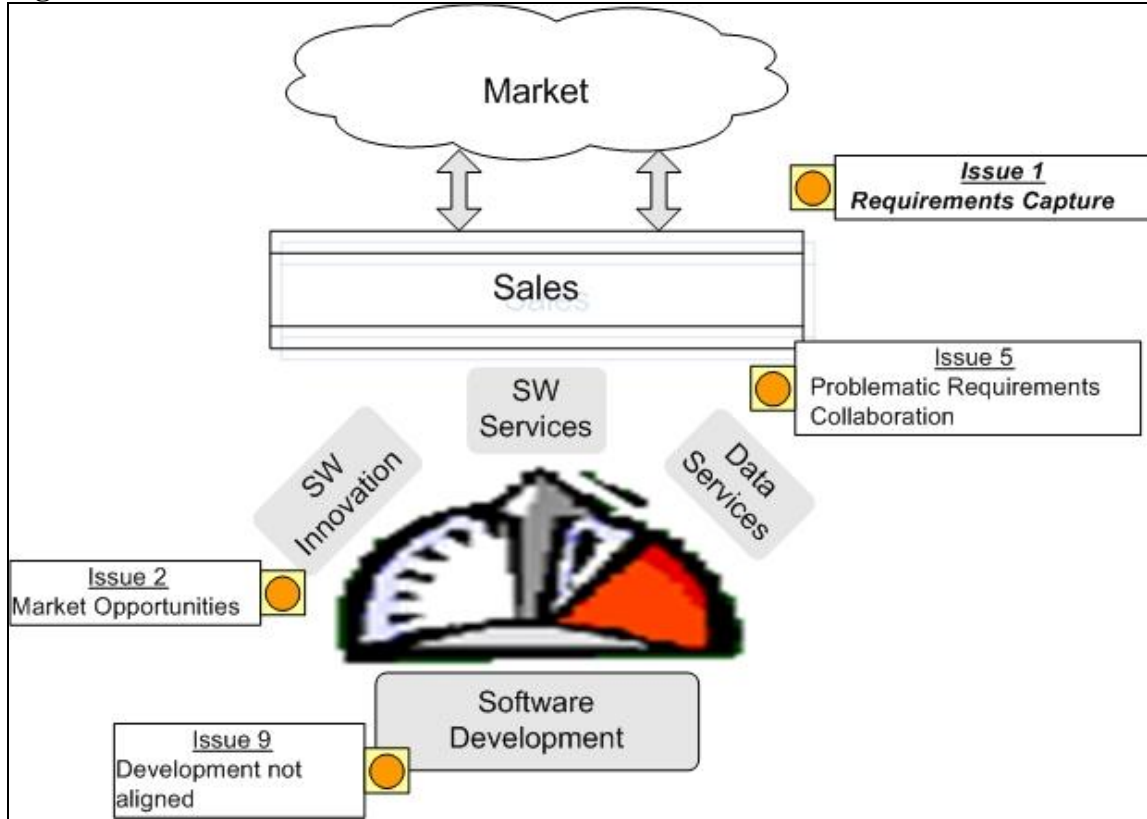
Major Differences between functions

Responses were grouped by function to determine whether priorities and needs differed. Table 8 reports the average “Criticality” score (1=irrelevant, 2=maybe useful, 3=useful, 4=very useful, 5=critical) by role. The table only shows those issues where there were differences among stakeholder groups.

Table 8: Role-based view of critical issues

| <i>Issue</i> | <i>Description</i> | <i>Sales & Marketing (2 people)</i> | <i>Data Services (6 people)</i> | <i>Development (1 person)</i> |
|--------------|--|---|-------------------------------------|-----------------------------------|
| 1 | Unsystematic early capture of requirements | 3.5 | 4.8 | 3.0 |
| 3 | Complex chain of requirements communication | 2.5 | 4.0 | 4.0 |
| 7 | Data Services pricing squeezes requirements implementation | 3.5 | 4.5 | 5.0 |
| 11 | Data Services pays for development errors | 2.0 | 3.5 | 2.0 |
| 12 | Unsystematic error tracking | 4.5 | 4.0 | 3.0 |
| 14 | Data Services product rejection | 4.5 | 3.3 | 5.0 |

Figure 2: Software-Sales Model



Note: *Issue* in bold italics is from the “Top Issues” list.

Figure 3: Data Services - Software model

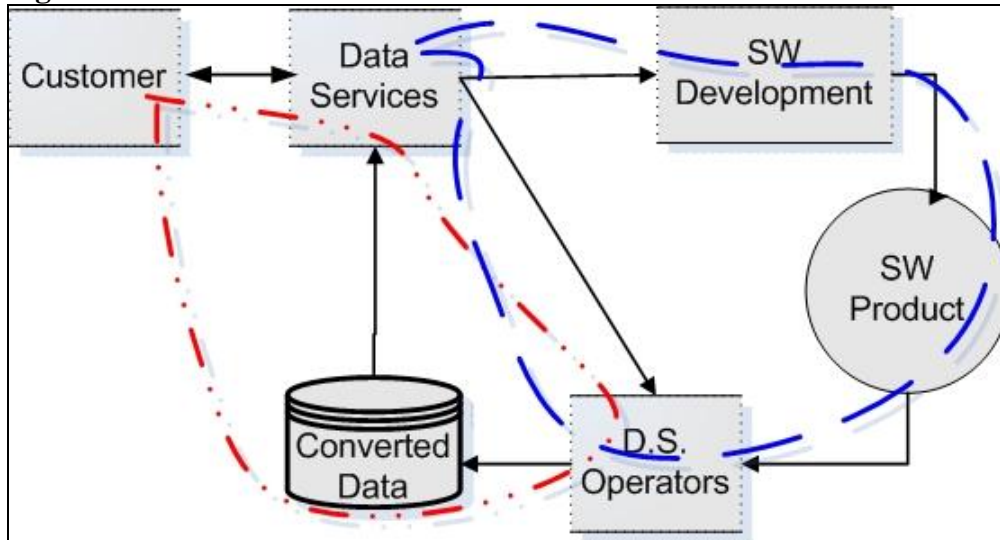


Table 9: Requirements Issues from Data Services perspective

| <i>Interaction</i> | <i>Related Issue(s)</i> |
|---------------------------------------|--|
| Data Services – SW Development | 3: Complex chain of requirements communication <i>7: Data Services pricing squeezes requirements implementation</i> <i>8: Software often rejected by Data Services</i> <i>10: Deadlines not met for Data Services software</i> 11: Data Services pays for Development errors 12: Unsystematic error tracking |
| Data Services – D.S. Operators | <i>4: Changes not systematically communicated to Data Services operators</i> 13: Data Services not exploited for process and product innovation |
| Customer – Data Services | <i>6: Varying contributions of Source to Target Matrix</i> 14: Data Services product rejection |

Note: *Issues* in bold italics are from the “Top Issues” list.

External Customer View

Researchers from the Center for Process Innovation (CEPRIN) at Georgia State University interviewed *TelSoft*'s external customers regarding the requirements management processes.

The following customer representatives generously gave their time to participate in this effort:
<Names Withheld>

At least one participant expressed keen interest in receiving a copy of the findings from this round of interviews. We recommend that a separate report describing the actions to be taken in response to the interviews be distributed to the external customers as soon as possible.

Executive Summary of Far Telco interviews

Far Telco employees were consistent regarding *TelSoft*'s strengths: dedicated personnel who are knowledgeable about Far Telco's processes and business needs. At this point, the *TelSoft*-Far Telco relationship seemed stronger and closer than the IBM-Far Telco relationship. Most of the time, they liked the fact that *TelSoft* plays a consulting role, making recommendations on alternative solutions and warnings of change impacts. *TelSoft* is seen as responsive when called upon by Far Telco. The EWO software may be old, but it meets the needs that Far Telco currently has.

Direct quotes include:

- "Out of all the different vendors I work with, this one works pretty smoothly."
- "We choose *TelSoft* software because they have a good relationship with us in the past. They've performed when other people have not performed. They know our business. They pretty much understand our engineering processes."
- "I know I can get in contact with them and ask a question. I'm also confident that they'll respond to me in a timely manner."
- "*TelSoft* has a good handle on our business and our needs – sometimes even better than our process owners."

Some challenges for *TelSoft* going forward:

Reactive rather than Proactive. A recurring weakness mentioned is that *TelSoft* is not proactive in its relationship with Far Telco. Two problems occur as a result. First, customer feels "taken for granted." Second, business opportunities are missed.

Early detail-orientation bogs down the process. Client understands the need for *TelSoft* to know details in order to provide estimates. However, they would prefer a ballpark figure instead of getting down into details early.

Great relationship but don't take it for granted. Compared to other vendors, *TelSoft* does not have an onsite presence. They don't visit monthly, talk about future plans for the software, or provide ongoing training. Need to keep in mind that Far Telco upper management compares *TelSoft* to other vendors that have flashier presentation styles.

Respond to the little concerns as though they were big. Clients described problems to us that they had previously mentioned to *TelSoft* personnel. For example, several minor irritations with email communication were mentioned (e.g., irrelevant subject line, text in the body of the message instead of an attachment, and replying with attached files).

Better manage the Testing Process. *TelSoft* typically delivers the “Testing Requirements” when the code is delivered. One interviewee preferred to see these at the time of the Design Walkthrough when the Functional Spec is reviewed. That way, they can better know the kinds of things that *TelSoft* might potentially miss during testing.

Executive Summary of Local Telco interviews

Local Telco agreed with Far Telco that the strength at *TelSoft* is in its people. *TelSoft* knows and understands their business. They loved having onsite support in the past. They felt that their current contacts at *TelSoft* are responsive and willing to help when called upon.

Overall, Local Telco expressed a “lack of confidence” in *TelSoft* ability to consistently deliver quality code. One interviewee stated that *TelSoft* was in “fast delivery mode” and “throwing software over the wall as a time-savings device.” They were concerned that the software packages they received contained unsolicited changes that were put in for other customers. Selected quotes:

- “We don’t have a confidence level in what we receive in a software package. We don’t even have confidence that it’s ours.”
- “Unless we ask for it, we don’t get documentation on what’s in the release, what changes have been made. We don’t get the packaging instructions on the package... I don’t believe there is a repository for me to roll back to.”

Reactive rather than Proactive. When contacted, *TelSoft* is responsive. However, when *TelSoft* discovers a problem, they don’t initiate communication about that to Local Telco.

Better manage the Testing Process. Testing is too limited and doesn’t catch as much as it should. One interviewee speculated that the level of testing done was a more related to what their (*TelSoft*’s) schedule allowed rather than the needs of the software.

Customer Relationship Management.

- ”We don’t have a partnership relationship. A lot of times we kind of feel like there’s animosity from them toward us. I don’t know how big of a customer we are in their eyes, but I don’t feel treated like a valued customer.”

Strengths and Challenges

The following tables summarize the customer perspectives on strengths and challenges in their relationship with *TelSoft*.

Table 10: Strengths: External Customer View

| <i>Strengths</i> | <i>Level of Agreement</i> |
|--|-------------------------------------|
| <i>TelSoft</i> understands our business. | Local Telco, Far Telco |
| <i>TelSoft</i> 's software basically meets our business need. | Local Telco, Far Telco |
| <i>TelSoft</i> is responsive to customer requests. | Other Telco, Local Telco, Far Telco |
| <i>TelSoft</i> is flexible in adapting to our processes and tools. | Other Telco, Local Telco, Far Telco |
| <i>TelSoft</i> has dedicated and knowledgeable employees. | Other Telco, Local Telco, Far Telco |
| <i>TelSoft</i> plays a consulting role and recommends alternative solutions. | Far Telco |
| <i>TelSoft</i> explains the rationale behind estimates well. | Far Telco |

Table 11: Challenges: External Customer View

| <i>Challenges</i> | <i>Level of Agreement</i> |
|---|-------------------------------------|
| <i>TelSoft</i> needs to decrease the number of bugs and unexpected changes in delivered software. | Local Telco, Far Telco, Other Telco |
| <i>TelSoft</i> needs to increase the transparency and consistency of its configuration management, documentation, and test activities. | Local Telco |
| <i>TelSoft</i> needs to enhance its customer relationship management. | Local Telco, Far Telco, Other Telco |
| <i>TelSoft</i> needs to improve its packaging procedures and related release notes. | Local Telco, Far Telco |
| <i>TelSoft</i> should become more involved with end users to identify and anticipate changes and to support training. | Far Telco |
| <i>TelSoft</i> needs to be better at proactively sharing relevant information about revisions and plans with the client. | Local Telco |
| <i>TelSoft</i> needs to increase the frequency and consistency of their communication with the client. | Far Telco |
| <i>TelSoft</i> should seek to increase its access to and utilization of client systems and facilities (e.g., EDP, NetMeeting, Local Telco test facilities). | Local Telco, Far Telco |
| <i>TelSoft</i> should be better at making early estimates to help scope projects. | Far Telco |
| <i>TelSoft</i> should streamline its software interface to be more competitive. | Other Telco |

Standardized Assessment

The following report was obtained from administering the assessment from Sommerville and Sawyer's 1997 book: *Requirements Engineering: A Good Practice Guide (REGPG)*. The assessment was conducted on March 30, 2005.

Table 12: Area Strength Matrix

| <i>Area</i> | <i>Area ID</i> | <i>Weak</i> | <i>Average</i> | <i>Good</i> | <i>Strong</i> |
|---|----------------|-------------|----------------|-------------|---------------|
| Requirements Document | 3 | | | | * |
| Requirements Elicitation | 4 | | | * | |
| Requirements Analysis and Negotiation | 5 | * | | | |
| Describing Requirements | 6 | | | | * |
| System Modeling | 7 | | * | | |
| Requirements Validation | 8 | * | | | |
| Requirements Management | 9 | * | | | |
| Requirements Engineering for Critical Systems | 10 | * | | | |

Note: Area ID corresponds to the chapter in the REGPG book.

Table 13: Guideline Usage Summary

| <i>Area ID</i> | <i>03</i> | <i>04</i> | <i>05</i> | <i>06</i> | <i>07</i> | <i>08</i> | <i>09</i> | <i>10</i> |
|------------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| Guideline counts | 7 | 9 | 2 | 4 | 2 | 2 | 2 | 0 |
| Maximum | 8 | 13 | 8 | 5 | 6 | 8 | 9 | 9 |
| % Usage | 88 | 69 | 25 | 80 | 33 | 25 | 22 | 0 |

Table 14: Overall Summary

| | <i>Basic</i> | <i>Intermediate</i> | <i>Advanced</i> |
|--------------------|----------------|---------------------|-----------------|
| Guidelines Used | 19 | 9 | 0 |
| Weighted Score | 37 | 14 | 0 |
| Maximum Possible | 105 | 66 | 27 |
| Score % of Maximum | 35 | 21 | 0 |
| Level | Initial | | |

Score for Basic, Intermediate, and Advanced Guidelines**Table 15: Score against basic guidelines**

| <i>ID</i> | <i>Guideline</i> | <i>Score</i> |
|--|---|--------------|
| 03.02 | Explain how to use the document | 0 |
| 04.02 | Be sensitive to organizational and political considerations | 0 |
| 04.05 | Define the system's operating environment | 0 |
| 04.06 | Use business concerns to drive requirements elicitation | 0 |
| 05.01 | Define system boundaries | 0 |
| 05.02 | Use checklists for requirements analysis | 0 |
| 05.05 | Prioritize requirements | 0 |
| 06.01 | Define standard templates for describing requirements | 0 |
| 07.03 | Model the system architecture | 0 |
| 08.01 | Check that the requirements document meets your standards | 0 |
| 08.02 | Organize formal requirements inspections | 0 |
| 08.04 | Define validation checklists | 0 |
| 09.02 | Define policies for requirements management | 0 |
| 09.04 | Maintain a traceability manual | 0 |
| Number of Not Used Scores = 14 | | |
| 03.08 | Make the document easy to change | 1 |
| 04.01 | Assess system feasibility | 1 |
| 04.03 | Identify and consult system stakeholders | 1 |
| 04.04 | Record requirements sources | 1 |
| 05.04 | Plan for conflicts and conflict resolution | 1 |
| 06.02 | Use language simply, consistently and concisely | 1 |
| 06.04 | Supplement natural language with other descriptions of requirements | 1 |
| 07.01 | Develop complementary system models | 1 |
| 07.02 | Model the system's environment | 1 |
| Number of Discretionary Scores = 9 | | |
| 05.03 | Provide software to support negotiations | 2 |
| 06.03 | Use diagrams appropriately | 2 |
| Number of Normal Scores = 2 | | |
| 03.01 | Define a standard document structure | 3 |
| 03.03 | Include a summary of the requirements | 3 |
| 03.04 | Make a business case for the system | 3 |
| 03.05 | Define specialized terms | 3 |
| 03.06 | Lay out the document for readability | 3 |
| 03.07 | Help readers find information | 3 |
| 08.03 | Use multi-disciplinary teams to review requirements | 3 |
| 09.01 | Uniquely identify each requirement | 3 |
| Number of Standardized Scores = 8 | | |
| Number of Basic Guidelines Assessed = 33 | | |
| Final Score | | 37 |

Table 16: Score against intermediate guidelines

| <i>ID</i> | <i>Guideline</i> | <i>Score</i> |
|---|---|--------------|
| 05.06 | Classify requirements using a multi-dimensional approach | 0 |
| 05.07 | Use interaction matrices to find conflicts and overlaps | 0 |
| 07.04 | Use structured methods for system modelling | 0 |
| 07.05 | Use a data dictionary | 0 |
| 07.06 | Document the links between stakeholder requirements and system models | 0 |
| 08.05 | Use prototyping to animate requirements | 0 |
| 08.07 | Propose requirements test cases | 0 |
| 09.03 | Define traceability policies | 0 |
| 09.05 | Use a database to manage requirements | 0 |
| 09.06 | Define change management policies | 0 |
| Number of Not Used Scores = 10 | | |
| 04.10 | Prototype poorly understood requirements | 1 |
| 04.11 | Use scenarios to elicit requirements | 1 |
| 04.12 | Define operational processes | 1 |
| 08.06 | Write a draft user manual | 1 |
| 09.07 | Identify global system requirements | 1 |
| Number of Discretionary Scores = 5 | | |
| 04.08 | Record requirements rationale | 2 |
| 04.09 | Collect requirements from multiple viewpoints | 2 |
| 06.05 | Specify requirements quantitatively | 2 |
| Number of Normal Scores = 3 | | |
| 04.07 | Look for domain constraints | 3 |
| Number of Standardized. Scores = 1 | | |
| Number of Intermediate Guidelines Assessed = 19 | | |
| Final Score | | 14 |

Table 17: Score against advanced guidelines

| <i>ID</i> | <i>Guideline</i> | <i>Score</i> |
|--|--------------------------------|--------------|
| 04.13 | Reuse requirements | 0 |
| 05.08 | Assess requirements risks | 0 |
| 08.08 | Paraphrase system models | 0 |
| 09.08 | Identify volatile requirements | 0 |
| 09.09 | Record rejected requirements | 0 |
| Number of Not Used Scores = 5 | | |
| Number of Advanced Guidelines Assessed = 5 | | |
| Final Score | | 0 |

Unused Guidelines by Cost of Implementation**Table 18: Very low cost of implementation**

| <i>ID</i> | <i>Guideline</i> | <i>Type</i> |
|-----------|---------------------------------|-------------|
| 03.02 | Explain how to use the document | Basic |

Table 19: Low cost of implementation

| <i>ID</i> | <i>Guideline</i> | <i>Type</i> |
|-----------|---|--------------|
| 04.02 | Be sensitive to organizational and political considerations | Basic |
| 04.05 | Define the system's operating environment | Basic |
| 04.06 | Use business concerns to drive requirements elicitation | Basic |
| 05.01 | Define system boundaries | Basic |
| 05.05 | Prioritize requirements | Basic |
| 08.01 | Check that the requirements document meets your standards | Basic |
| 09.04 | Maintain a traceability manual | Basic |
| 10.02 | Involve external reviewers in the validation process | Basic |
| 05.07 | Use interaction matrices to find conflicts and overlaps | Intermediate |
| 07.06 | Document the links between stakeholder requirements and system models | Intermediate |
| 08.07 | Propose requirements test cases | Intermediate |
| 10.04 | Derive safety requirements from hazard analysis | Intermediate |
| 10.05 | Cross-check operational and functional requirements against safety requirements | Intermediate |
| 09.08 | Identify volatile requirements | Advanced |
| 09.09 | Record rejected requirements | Advanced |

Table 20: Low to Moderate cost of implementation

| <i>ID</i> | <i>Guideline</i> | <i>Type</i> |
|-----------|--|--------------|
| 05.02 | Use checklists for requirements analysis | Basic |
| 07.03 | Model the system architecture | Basic |
| 08.04 | Define validation checklists | Basic |
| 10.01 | Create safety requirement checklists | Basic |
| 05.06 | Classify requirements using a multi-dimensional approach | Intermediate |

Table 21: Moderate cost of implementation

| <i>ID</i> | <i>Guideline</i> | <i>Type</i> |
|-----------|---|--------------|
| 06.01 | Define standard templates for describing requirements | Basic |
| 08.02 | Organize formal requirements inspections | Basic |
| 09.02 | Define policies for requirements management | Basic |
| 07.05 | Use a data dictionary | Intermediate |
| 09.03 | Define traceability policies | Intermediate |
| 05.08 | Assess requirements risks | Advanced |

Table 22: Moderate to high cost of implementation

| <i>ID</i> | <i>Guideline</i> | <i>Type</i> |
|-----------|--|--------------|
| 07.04 | Use structured methods for system modeling | Intermediate |
| 08.05 | Use prototyping to animate requirements | Intermediate |
| 09.05 | Use a database to manage requirements | Intermediate |
| 09.06 | Define change management policies | Intermediate |
| 10.03 | Identify and analyze hazards | Intermediate |
| 04.13 | Reuse requirements | Advanced |
| 08.08 | Paraphrase system models | Advanced |

Table 23: High cost of implementation

| <i>ID</i> | <i>Guideline</i> | <i>Type</i> |
|-----------|---|-------------|
| 10.06 | Specify systems using formal specifications | Advanced |
| 10.07 | Collect incident experience | Advanced |
| 10.08 | Learn from incident experience | Advanced |
| 10.09 | Establish an organizational safety culture | Advanced |

B.5: Software Charter

March 2006

Reason for Being

TelSoft Division exists to provide AM/FM/GIS software and services in an innovative and disciplined environment while earning a fair profit and enhancing our clients' business.

Software Strategy

TelSoft Division develops and maintains a standardized portfolio of software for delivering AM/FM/GIS solutions to clients. The portfolio is tailored to support the management and analysis of location-based asset information with a suite of tools to mechanize and streamline processes for planning, building, provisioning, and maintaining these assets.

TelSoft Software Policies

1. *TelSoft* will strive to operate based on the highest professional standards and processes.
2. *TelSoft* will strive to understand and incorporate its customers' business knowledge in our products.
3. *TelSoft* will maintain a proactive professional relationship to its customers.
4. *TelSoft* will manage each development project with a two-phase approach that separates requirement and development activities.
5. *TelSoft* will only engage resources to start design and construction when *TelSoft* has a baseline of identifiable and agreed upon requirements.
6. *TelSoft* will only engage resources to address requirement change requests that are documented, agreed upon and applied to the requirements baseline.
7. *TelSoft* will communicate status to its customers of all active projects on a regular basis.
8. *TelSoft* will only deliver official releases of software to a client with the written approval of Quality Assurance.
9. Each release of *TelSoft* software will include documentation of all changes and new features since the previous release.

B.6: Software Coordination Group Fixed Agenda

Strategy Revision – Lead, Division President

Preconditions

- Latest version of Software Strategy is available
- A log of concerns and opportunities related to *TelSoft's* Software Strategy is available

Meeting activity

- Review and possibly revise *TelSoft's* Software Strategy based on Log
- Keep a log of concerns and opportunities related to the Strategy as foundation for future revisions

Expected outcome

- Continuous communication of Software Strategy to external and internal stakeholders
- Strong foundation for managing customer relationships
- Strong foundation for the Software Coordination Group

Software Project Review – Lead, Software Manager

Preconditions

- A list of all current and future software projects is available
- Each current project has predefined key performance indicators (should fit on a single page)
- Updated status on progress against these key performance indicators (KPI) have been provided to all members of the SCG at least 48 hours prior to the meeting. The KPI should be aligned with *TelSoft's* required set of minimal disciplines for software development.
- Each future project is described in terms of champion, business rationale, and expected resources and outcomes.

Meeting Activity

- Review KPI for each software project
- Prioritize resources for projects across the entire portfolio
- Review overall portfolio performance

Expected outcome

- Recommendations for improving individual project performance (e.g., adjust the software project plan, provide additional personnel, provide incentives for employees, etc.)
- Recommendations for improving overall portfolio performance (e.g., reallocate resources to higher priority projects, consider terminating low-performing projects, etc.)
- Prioritized and transparent portfolio of projects

Opportunity Review – Lead, Product Manager

Preconditions

- A list of market, technology and customer opportunities is available
- Opportunities can be described as either emerging opportunities or mature opportunities
- A cost-benefit analysis for each mature opportunity has been performed and provided to all members of the SCG at least 48 hours prior to the meeting

Meeting Activity

- Review cost-benefit analysis for each opportunity
- Provide additional feedback regarding these opportunities
- Evaluate whether each opportunity fits with the software strategy
- Recommend which opportunities should be promoted as future projects

Expected outcome

- Prioritized list of opportunities
- Possible revision of Software Strategy
- Software projects that will strengthen *TelSoft's* competitive position

Improvement Review – Lead, VP of Software

Preconditions

- A list of all current and future improvement initiatives is available
- Each current project has predefined key performance indicators (should fit on a single page)
- Updated status on progress against these key performance indicators (KPI) have been provided to all members of the SCG at least 48 hours prior to the meeting
- Each future project is described in terms of champion, business rationale, and expected resources and outcomes. (*Roughly half page*)
- Annual or bi-annual assessments of software development practices are conducted to identify possible new improvement initiatives

Meeting Activity

- Review KPI for each improvement project
- Prioritize improvement projects across the entire portfolio
- Identify areas in which new improvement initiatives should be considered

Expected outcome

- Recommendations for improving individual project performance (e.g., adjust the software project plan, provide additional personnel, provide incentives for employees, etc.)
- Recommendations for improving overall portfolio performance. Determine whether to invest more or less money in these improvement activities (e.g., reallocate resources to higher priority projects, consider terminating low-performing projects, etc.)
- Recommendations for new improvement initiatives.
- Prioritized portfolio of projects

Major Account Review – Rotating Lead

Preconditions

- Lead has assembled information regarding the relationship between the client and *TelSoft* Software group (e.g., Has the customer's perception of us changed? What currently threatens this relationship? Are there other people within these organizations that we should be talking with?)
- Lead may also present specific recommendations for improving this relationship

Meeting Activity

- Listen, identify potential opportunities, and recommend actions

Expected outcome

- New directions, some decision making, possible realignment

B.7: Problem Solving Team Fixed Agenda

Improvement Project Monitoring – Lead TBD

Preconditions

- Status (plan comparison, intermediate results, issues, suggestions, requests, lessons learned) of all current process improvement teams provided two days in advance of meeting
- Artifacts (position papers, templates, process documents, etc.) from process improvement teams provided two days in advance of meeting

Meeting Activity

- Review and discuss status and artifacts
- Review and update Process Document Summary
- Record any recommendations
- Determine if any follow up or support from PST is needed
- Budget and schedule review

Expected outcome

- Recommendations for current improvement projects are communicated
- PST gains appreciation for status of continuous improvement

Process Management Monitoring – Lead TBD

Preconditions

- Quality Assurance group is responsible for the day-to-day management of processes
- Status from Process Management Activity (policy and process issues, infrastructure and repository issues, resources, alignment of practice with process management process)

Meeting Activity

- Review and discuss status
- Determine if any follow up or support from PST is needed

Expected outcome

- Feedback and recommendations to Quality Assurance group
- PST gains appreciation for process management practice and process

Practice, Policy, and Process Assessment – Lead TBD

Preconditions

- *TelSoft* is committed to assess software practice, policies, and processes on a regular basis
- Plan for next assessment of software practice, policies, and processes
- Preliminary results from ongoing assessments

Meeting Activity

- Discuss and decide upon assessment plans
- Discuss and decide upon stakeholder involvement in assessments
- Discuss preliminary results from ongoing assessment and provide feedback

Expected outcome

- Assessments of software practice, policies, and processes are conducted on a regular basis
- Ongoing assessments are facilitated and monitored

Improvement Identification and Prioritization – Lead TBD

Preconditions

- Post-project review documentation for any projects recently completed
- List of process improvement ideas submitted from the web page suggestion box
- Survey results for any surveys conducted
- Final assessment reports (when available)

Meeting Activity

- Review post-project review documentation for any process issues
- Brainstorm ideas for other process improvement activities we should undertake
- Discuss and prioritize recommendations based upon final assessment reports
- Determine what new process improvements should be implemented, assign resources for implementation

Expected outcome

- Recommendations from various sources assessments are continuously prioritized for action
- The portfolio of ongoing and possible improvement initiatives is maintained
- Proposal for new process improvement initiatives – including focus, goals, deliverables, and resources – is sent to SCG

Participation and Communication

Preconditions

- The portfolio of ongoing and possible improvement initiatives is maintained
- Status and plans for ongoing improvement initiatives are available

Meeting Activity

- Review and discuss stakeholder involvement in improvement activities
- Review and discuss communication needs and opportunities about improvement activities
- Decide on improved participation and communication strategies
- Identify opportunities to communicate issues and celebrate results

Expected outcome

- Stakeholders are appropriately involved and sufficiently informed about *TelSoft* improvement initiatives

B.8: Second Wave Summary Report

Background

The collaboration began in October 2004 with an overall plan described in the Memorandum of Understanding. Between December 2004 and May 2005, CEPRIN assessed requirements practices by interviewing individuals from three stakeholders groups: software development, internal customers, and external customers. In addition, a standards assessment was conducted based upon the Requirements Engineering Good Practice Guide checklist. The results of this assessment were summarized in the Phase 1 Report. Based upon this data, CEPRIN identified seven improvement areas and recommended that a sense-and-respond approach be used to guide the improvement.

During the first wave, the Problem Solving Team (PST) designed five project teams for addressing these improvement areas: Software Coordination, Customer Relations, Requirements Management, Quality Assurance, and Configuration Management. Each team was given a suggested set of activities to be completed by April 1, 2006. The accomplishments of the first wave teams were documented in a First Wave report.

During the second wave, the PST reconfigured the have three project teams: Quality Results, Customer Relations, and Process Management. This report describes the accomplishments of these teams and lessons learned. A kick-off meeting was held on April 18, 2006 for all members of the software development group. The objectives of this meeting were to describe key processes and templates identified during the first wave, identify questions regarding the software policies, discuss how implementing these policies will impact employee work, and introduce the upcoming 2nd wave activities.

Improvement team results

This report summarizes the results of the three project teams from the second wave. This corresponds to the second Establishing and Acting phases of the IDEAL model.

The following sections provide the following information for each team:

- original ideas suggested at the Kick-off meeting
- team accomplishments during the second wave
- implementation activities

Choices for evaluating the state of each action (To be determined by PST):

- Done
- Deferred
- Planned, prepared, but not implemented
- Modified

Quality Results

Team lead: VR

Participants: *Names withheld*

Original Suggestions

- 1) Enhance internal Quality Assurance processes
 - Post release analysis
 - Clean up bug database
 - Improve efficiency of QA department
- 2) Improve software release management
 - Establish archiving process for releases
 - Create software release database
 - Maintain required files list

Accomplishments

- 1) The team developed six position papers:
 - PDPR Database Cleanup
 - QA Archiving of builds and releases
 - Improve efficiency of QA department
 - Post Release Quality Review
 - Software release database
 - Documenting build contents (originally called maintain required files list)

Of these six, two position papers were removed from scope of our team based on PST review of 7/21/06. The ‘software release database’ initiative was handed over to the customer relations team and the ‘documenting build contents’ was determined to be internal to QA and was essentially covered in the “QA Executes Builds” process developed in Phase 1.

- 2) The following process documents were created:
 - PDPR database cleanup
 - Defined an initial process where all bugs over 3 years old are closed and archived; bugs that are assigned to former employees are reassigned to appropriate personnel.
 - Manual process to review the remaining bugs
 - A process developed to keep the database updated longer-term on an ongoing basis.
 - Improve efficiency of QA department
 - Defined a regression testing process utilizing the regression checklist introduced in phase 1
 - Added a process step to create a high-level test case list prior to generating detailed test cases
 - Added metrics collection (cost, schedule, release, and bug metrics) and created a template for collecting/storing these metrics
 - QA archiving of builds and releases
 - [\\devsrv\certification](#) has been defined as a read-only share for QA builds
 - QA will keep create and retain master CDs of each release
 - Post Release Quality Review
 - Defined a process for post project review
 - Created a template to be used for post project reviews

Lessons learned

The following are items the team identified as lessons we learned during the course of executing the process improvement initiatives:

- It would have been more efficient to have included specific personnel from the PST in the position paper review cycle in order to hash out issues earlier. Sometimes feedback from the PST came late in the cycle.
- At the start of the process improvement initiatives, it would have been beneficial to have information such as the reason behind the initiative, perceived benefits, intended scope, etc. This would have helped the team make a better determination on how best to resolve the initiative. We ended up dropping one initiative and moving another one after we had spent time working on them.

Suggestions moving forward

The team came up with the following suggestions for moving forward.

- Continue with implementation of items in phase 1 that are not yet completed (regression checklists, for example)
- Develop details of the QA build process that were defined at a high level in Phase 1 (for example, where are build content documents stored, how file comparison from release to release is to be done, etc.).
- Implement the processes and utilize the templates developed above. Create a transition plan if necessary.

Customer Relations

Team lead: RW

Members: *Names withheld*

Original Suggestions

- 1) Maintain customer profile information
- 2) Improve image through customer deliverables
- 3) Increase *TelSoft* “presence” with the customer
 - Establish direct customer communication
 - Establish regular management communication with customer

Accomplishments

The team developed the following papers:

- Policy Statement:
 - *TelSoft* Email Correspondence Policy Statement
- Guidelines:
 - Proposals to Include Deployment Support
 - Deliver Proposals with a Presentation
 - Management Discussion Points
 - Customer Engagement

In addition, the team:

- 1) Put together packaging for all CD delivered products (Jacket, CD Label and insert)
- 2) Identified requirements for a division wide contacts database and reviewed the "ACT" product against these requirements

Lessons learned

None at this time

Suggestions moving forward

Continue with implementation of contact database and integration with existing processes in the company.

Process Management

Team lead: JV

Members: *Names withheld*

Original Suggestions

- 1) Update web site to reflect most useful information about *TelSoft*' processes and templates
- 2) Evaluate all existing processes in relation to future use at *TelSoft*
- 3) Create standards for templates and review 1st wave deliverables in light of these standards
- 4) Create plan for process management to be integrated into QA by end of 2nd wave

Accomplishments

- Weeded out process documents no longer used, identified those that need to be revised or approved and categorized them as such in Notes.
- Created standards for all process documents and templates.
- Reviewed the phase 1 and the documents on the external web page for compliance and generated compliance reports.
- Implemented a suggestion box on the web site where people can submit process-related suggestions.
- Created Process Management process document.
- Created fixed agenda for the PST.
- Developed interpersonal relationships with team members.
- Created Oracle database for tracking/managing suggestions from the web site.
- Published our software policies and templates to web site.

Lessons learned

- More frequent and earlier input/review of the web site by upper management, and the PST was needed. We spun our wheels a lot and good, clear direction did not get provided until late in the improvement project.

Suggestions moving forward

- Continue updating and bringing into compliance the documents we are keeping as part of our process.
- Get internal view of web site completed.

Planned activities through 2006

The final phase of the IDEAL model is the leveraging, or learning phase. The leveraging phase is a time of critical reflection in which lessons learned during earlier phases are used to refine the next software process improvement (SPI) cycles. In addition, we would like to evaluate the impact of the SPI effort by conducting an assessment that can assist the PST in planning future improvement initiatives.

An overview of the remaining SPI activities that will be conducted with GSU under this initial contract:

- Continued focus on implementation. Need to bridge the gap between current and desired implementation status of processes. We will especially concentrate on getting the process management plan implemented.
- Assessment of current practice and the impact of SPI using the following techniques
 - Survey for those internal to the software development group to allow complete coverage.
 - Interviews for representatives from software development, internal customers, and external customers.
 - Standardized requirements engineering assessment done by the PST
 - Interviews with members of the Software Coordination Group (SCG) regarding the group's process and overall effectiveness
- Create plan for 2007

Key Activities in Second Wave

| Date | Activity |
|--------------------|--|
| April 18, 2006 | Second Wave Kick-off Meeting |
| September 9, 2006 | PST meets to provide initial baseline of Process Documents (see Baseline of Software Processes (9/11/2006)) |
| September 29, 2006 | Deliverables from each team due to PST |
| October 17, 2006 | Planned meeting to report and celebrate results of SPI initiative (Rescheduled) |
| November 8, 2006 | Meeting to report and celebrate results of SPI initiative |
| November 29, 2006 | First PST meeting using new Fixed Agenda |

*Baseline of Software Processes (9/11/2006)***Policy Assessment**

| ID | Associated Documentation (See table below) | Policy | Current Status | Desired Status |
|-----------|---|---|-----------------------|-----------------------|
| 1 | 31, 32, 33, 34, 35, 41, 1, 2, 28, 30, 3, 5, 17, 22, 23, 29, 39, 4, 27, 36, 37, 38 | Professional Standards: <i>TelSoft</i> will strive to operate based on the highest professional standards and processes. | Normally used | Normally used |
| 2 | 5, 36, 37 | Customer Knowledge: <i>TelSoft</i> will strive to understand and incorporate its customers' business knowledge in our products. | Normally used | Normally used |
| 3 | 6, 20, 21, 25 | Relationship Management: <i>TelSoft</i> will maintain a proactive professional relationship to its customers. | Discretionary | Normally used |
| 4 | 7 | Two-phase Funding: <i>TelSoft</i> will manage each development project with a two-phase approach that separates requirement and development activities. | Normally used | Normally used |
| 5 | 7, 14, 40 | Requirements First: <i>TelSoft</i> will only engage resources to start design and construction when <i>TelSoft</i> has a baseline of identifiable and agreed upon requirements. | Normally used | Standardized |
| 6 | 7, 8, 13, 26 | Change Request: <i>TelSoft</i> will only engage resources to address requirement change requests that are documented, agreed upon and applied to the requirements baseline | Discretionary | Standardized |
| 7 | 9, 12, 20, 21, 25, 28, 29 | Communicate Status: <i>TelSoft</i> will communicate status to its customers of all active projects on a regular basis. | Standardized | Standardized |

| <i>ID</i> | <i>Associated Documentation (See table below)</i> | <i>Policy</i> | <i>Current Status</i> | <i>Desired Status</i> |
|------------------|--|---|------------------------------|------------------------------|
| 8 | 10, 11, 15, 16, 18, 19, 22 | QA Approval: <i>TelSoft</i> will only deliver official releases of software to a client with the written approval of Quality Assurance. | Standardized | Standardized |
| 9 | 18, 19, 24, 38 | Release Documentation: Each release of <i>TelSoft</i> software will include documentation of all changes and new features since the previous release. | Discretionary | Standardized |

Documentation Summary

| <i>ID</i> | <i>Processes, Templates, and Standards</i> | <i>Customer visibility</i> | <i>Related Policies</i> | <i>Source</i> | <i>Documentation status</i> | <i>Current Implementation Status</i> | <i>Desired Implementation Status</i> |
|-----------|---|----------------------------|-------------------------|---------------|-----------------------------|--------------------------------------|--------------------------------------|
| 7** | High Level Requirements Specification (HLRS) Template | Now | 4, 5, 6 | Legacy | Needs approval | Discretionary | Discretionary |
| 13** | Change Control Template | Now | 6 | First Wave | Needs approval | Discretionary | Standardized |
| 14** | Functional Specification Template | Now | 5 | First Wave | Needs approval | Normally used | Standardized |
| 6** | Statement of Work template | Now | 3 | Legacy | Needs approval | Standardized | Standardized |
| 11** | Test Procedures Template | Now | 8 | Legacy | Needs Approval | Standardized | Standardized |
| 20** | Customer Project Status Report Template | Now | 7, 3 | First Wave | Needs revision | In progress | In progress |
| 10** | Test Evaluation Report Template | Now | 8 | Legacy | Needs revision | Standardized | Standardized |
| 1 | Risk Management Guidelines | Never | 1 | Legacy | Needs revision | Discretionary | Discretionary |
| 2 | Risk Management Templates | Never | 1 | Legacy | Needs revision | Not used | Discretionary |
| 3 | Software Development Process Flow & Description | Later | 1 | Legacy | Needs revision | Discretionary | Normally used |
| 4 | Technical Specification Template | Never | 1 | Legacy | Needs revision | Discretionary | Discretionary |
| 5 | Project Planning Process Flow & Description | Later | 1, 2 | Legacy | Needs revision | Normally used | Normally used |
| 8 | Defect Management Guidelines | Never | 6 | Legacy | Needs revision | Discretionary | Normally used |
| 9 | Project Tracking and Oversight Guidelines | Later | 7 | Legacy | Needs revision | Normally used | Standardized |
| 12 | One-Page Status Report Template | Never | 7 | Legacy | Needs revision | Standardized | Standardized |
| 15 | Regression Checklists Template | Never | 8 | First Wave | Needs approval | In progress | Normally used |

| <i>ID</i> | <i>Processes, Templates, and Standards</i> | <i>Customer visibility</i> | <i>Related Policies</i> | <i>Source</i> | <i>Documentation status</i> | <i>Current Implementation Status</i> | <i>Desired Implementation Status</i> |
|-----------|--|----------------------------|-------------------------|---------------|-----------------------------|--------------------------------------|--------------------------------------|
| 16 | Regression Testing Process | Never | 8 | Second Wave | Needs creation | In progress | Normally used |
| 17 | Software Coordination Group Process | Later | 1 | Second Wave | Needs creation | Standardized | Standardized |
| 18 | Software Release Specification Template | Later | 9, 8 | First Wave | Needs approval | Normally used | Standardized |
| 19 | Software Release Specification Process | Later | 9, 8 | Second Wave | Needs approval | Normally used | Standardized |
| 21 | Customer Email Standard | Never | 7, 3 | Second Wave | Needs revision | In progress | In progress |
| 22 | Post Release Analysis Process | Later | 1, 8 | Second Wave | Needs creation | In progress | Normally used |
| 23 | Process Management Process (including approving processes and templates) | Later | 1 | Second Wave | Needs creation | In progress | Standardized |
| 24 | Software Release Management Process (including Packaging) | Later | 9 | Second Wave | Needs creation | Planned | Planned |
| 25 | Website Management Process | Never | 7, 3 | Second Wave | Needs creation | In progress | Standardized |
| 26 | Change Control Process | Later | 6 | Second Wave | Needs creation | Discretionary | Standardized |
| 27 | JCS Activity Code | Never | 1 | Legacy | Approved | Standardized | Standardized |
| 28 | Microsoft project plan template | Never | 1, 7 | Legacy | Needs approval | Discretionary | Discretionary |
| 29 | Estimating procedures | Never | 1, 7 | Legacy | Needs revision | Discretionary | Discretionary |
| 30 | Project kick-off meeting sample agenda | Later | 1 | Legacy | Needs revision | Discretionary | Discretionary |
| 31 | C++ Coding Guidelines | Never | 1 | Legacy | Needs revision | Normally used | Standardized |
| 32 | Rexx Coding Guidelines | Never | 1 | Legacy | Needs revision | Normally used | Standardized |
| 33 | Java Coding Guidelines | Never | 1 | Legacy | Needs revision | Normally used | Standardized |
| 34 | VBA Coding Guidelines | Never | 1 | Legacy | Needs revision | Normally used | Standardized |
| 35 | Java User Interface Rename: <i>TelSoft</i> GUI practices | Later | 1 | Legacy | Needs revision | Not used | Standardized |
| 36 | Unit Testing Guidelines | Never | 2, 1 | Legacy | Needs revision | Not used | Normally used |

| <i>ID</i> | <i>Processes, Templates, and Standards</i> | <i>Customer visibility</i> | <i>Related Policies</i> | <i>Source</i> | <i>Documentation status</i> | <i>Current Implementation Status</i> | <i>Desired Implementation Status</i> |
|-----------|--|----------------------------|-------------------------|---------------|-----------------------------|--------------------------------------|--------------------------------------|
| 37 | Integration Testing Guidelines | Never | 2, 1 | Legacy | Needs revision | Not used | Normally used |
| 38 | Software version numbering scheme | Never | 1, 9 | Legacy | Needs approval | Standardized | Standardized |
| 39 | Post Project Review Process | Later | 1 | Second Wave | Needs creation | In progress | In progress |
| 40 | Release Plan Template | Later | 5 | Legacy | Needs revision | In progress | Standardized |
| 41 | Java Error & Exception Handling Guidelines | Never | 1 | Legacy | Needs revision | Normally used | Standardized |
| 42 | Task Notes | Never | 1 | Legacy | Needs approval | Discretionary | Normally used |

** Indicates documents that will be made visible on the company's website.

B.9: Employee Survey

1. Assessment of Software Process Improvement

This questionnaire is being used to assess the Software Process Improvement (SPI) initiative which has been going on between *TelSoft* and Georgia State University (GSU) between 2004-2006. We are interested in your impressions regarding how the initiative was organized as well as its impact. The survey should take less than 20 minutes to complete. Your remarks will not be singled out by name. Instead, all results will be combined with all others by GSU researchers and presented in a final report.

Do you wish to participate in this online survey?

- a. Yes
- b. No

Demographic Information

2. Enter your name for purposes of following up.

3. What profit center do you primarily work for?

- a. Data Services (IDS)
- b. Software (ISW)
- c. Sales (ISL)
- d. Other _____

4. What is your primary job responsibility?

- a. Quality Assurance
- b. Sales (Account Executive, Marketing)
- c. Business Analyst
- d. Engineer (Software, Software Applications)
- e. Manager (e.g., Product, Project, Supervisor)
- f. GIS Technician
- g. Other _____

5. How long have you worked at *TelSoft*?

- a. Less than 2 years
- b. 2 - 7 years
- c. 7 - 12 years
- d. 12 - 17 years
- e. More than 12 years

Your Role in Improvement Initiative

6. Please indicate your level of involvement with the collaboration between Georgia State University (GSU) and *TelSoft*. Check all that apply.

- a. Problem Solving Team member

- b. Improvement team member
(e.g. Quality Results, Configuration management, Customer relations, etc.)
- c. Software Coordination Group member
- d. Attended workshop or kick-off meeting
- e. None

7. Please indicate your role in each of the following improvement teams:

| <i>Team</i> | <i>None</i> | <i>Participant</i> | <i>Project Manager</i> |
|-----------------------------|-------------|--------------------|------------------------|
| Configuration Management | | | |
| Customer Relations | | | |
| Problem Solving Team | | | |
| Process Management | | | |
| Quality Assurance/Results | | | |
| Requirements Management | | | |
| Software Coordination Group | | | |

Overall Impact of Initiative

- 8. Overall, what has been the impact of the improvement initiative over the last 2 years?
 - a. Made things worse
 - b. No change
 - c. Some improvement
 - d. Considerable improvement
 - e. Don't know

9. Please explain your answer:

Policy Impact

10. For each policy, what is the impact on everyday practices at *TelSoft*?
 Note: Click on link above for a reminder of policies from *TelSoft* website.

| | <i>Made things worse</i> | <i>No change</i> | <i>Some Improvement</i> | <i>Considerable Improvement</i> | <i>Don't know</i> |
|-------------------------|--------------------------|------------------|-------------------------|---------------------------------|-------------------|
| Professional Standards | | | | | |
| Customer Knowledge | | | | | |
| Relationship Management | | | | | |
| Two-phase Funding | | | | | |
| Requirements First | | | | | |
| Change Request | | | | | |

| | <i>Made things worse</i> | <i>No change</i> | <i>Some Improvement</i> | <i>Considerable Improvement</i> | <i>Don't know</i> |
|-----------------------|--------------------------|------------------|-------------------------|---------------------------------|-------------------|
| Communicate Status | | | | | |
| QA Approval | | | | | |
| Release Documentation | | | | | |

11. To what extent is each policy followed at *TelSoft*?

Note: Click on link above for a reminder of policies from *TelSoft* website.

| | <i>Not used (<20%)</i> | <i>Discretionary (<60%)</i> | <i>Normally used (<90%)</i> | <i>Standardized</i> | <i>Don't know</i> |
|-------------------------|---------------------------|--------------------------------|--------------------------------|---------------------|-------------------|
| Professional Standards | | | | | |
| Customer Knowledge | | | | | |
| Relationship Management | | | | | |
| Two-phase Funding | | | | | |
| Requirements First | | | | | |
| Change Request | | | | | |
| Communicate Status | | | | | |
| QA Approval | | | | | |
| Release Documentation | | | | | |

12. Optional area for commenting on policies:

Improvement Team Impact

13. What has been the impact of each of the specific initiatives done by the improvement teams during the First Wave?

| | <i>Made things worse</i> | <i>No change</i> | <i>Some Improvement</i> | <i>Considerable Improvement</i> | <i>Don't know</i> |
|---|--------------------------|------------------|-------------------------|---------------------------------|-------------------|
| Revised Functional Specification template | | | | | |
| Revised Change Control template | | | | | |
| Weekly Status Report Template | | | | | |
| Software Release Specification | | | | | |
| QA executes builds | | | | | |

14. What has been the impact of each of the specific initiatives done by the improvement teams during the Second Wave?

| | <i>Made things worse</i> | <i>No change</i> | <i>Some Improvement</i> | <i>Considerable Improvement</i> | <i>Don't know</i> |
|-----------------------------------|--------------------------|------------------|-------------------------|---------------------------------|-------------------|
| Refined QA process | | | | | |
| Post Project Reviews | | | | | |
| PDPR (Bug) Database Cleanup | | | | | |
| <i>TelSoft</i> Website update | | | | | |
| Suggestion Box on Website | | | | | |
| Improved Client Product packaging | | | | | |
| Customer Contact Database (ACT) | | | | | |

15. Optional area for additional comments regarding improvement team initiatives:

16. What is your perception regarding the amount of information provided about the improvement initiative?

- a. Not enough
- b. Enough
- c. Too much

17. What is your perception regarding your own level of participation the improvement initiative?

- a. Not enough
- b. Enough
- c. Too much

Open-ended Questions

18. List the 2-4 most important areas that still need to be improved.

19. List the 2-4 barriers that have limited the impact of the initiative.

20. List 1 - 3 suggestions for organizing future initiatives.

B.10: Learning Interview Guide

The objectives of the learning assessment are to evaluate SPI impact, organization, and perception. Specific questions asked were tailored based on the person's stakeholder group, level of involvement with the improvement initiative, and role and responsibilities within *TelSoft*. The comprehensive bank of questions is included below.

SPI Impact

1. In the two years that we've been working with *TelSoft*, what has been the overall impact of the improvement initiative?
2. Can you provide specific examples of how the initiative has positively impacted business?
3. How has the initiative impacted your day-to-day work?
4. How does this initiative compare with the prior CMM-based effort?
5. As we move forward, the PST is seeking advice on what was successful and what could be improved. What activities would you like to see repeated? Where do you think the PST should focus its efforts? What advice would you give to the PST moving forward?
6. Specific questions to ask about the improvement areas:

| <i>Area</i> | <i>Issues</i> | <i>Questions to ask</i> |
|--------------------------------------|---|---|
| 1. Software vision management | <i>TelSoft</i> strategy for software development and customer service should be explicated, maintained, and communicated. This provides a value-based foundation for requirements coordination and management that is consistent with <i>TelSoft's</i> business strategy. | <ol style="list-style-type: none"> a. To what extent is the strategy explicated, maintained, and communicated in all levels of the organization? b. To what extent are the policies explicated, maintained, and communicated in all levels of the organization? |
| 2. Project portfolio management | <i>TelSoft</i> software project portfolio should be managed explicitly and coordinated across internal and external stakeholders. This creates the necessary dynamic capability to respond effectively to different and emerging customer and innovation requests. | <ol style="list-style-type: none"> a. To what extent does <i>TelSoft</i> effectively manage and coordinate the project portfolio? b. Can <i>TelSoft</i> respond dynamically to different and emerging customer requests? c. Can <i>TelSoft</i> respond dynamically to innovations? |
| 3. Software configuration management | <i>TelSoft</i> software configuration management should be improved to ensure consistent and transparent modification and packaging to individual customers. This ensures effective coordination with customers and minimizes adverse effects across projects. | <ol style="list-style-type: none"> a. Is the defined process for generating software products for external customers consistent? b. Is the defined process for packaging software for external clients consistently followed? |

| <i>Area</i> | <i>Issues</i> | <i>Questions to ask</i> |
|----------------------------------|---|---|
| 4. Customer relations management | <i>TelSoft</i> should improve its management of customer relations to ensure more symmetric information sharing and proactive expectation and change management. This leads to increased customer satisfaction. | <ul style="list-style-type: none"> a. In what ways have customer relations been improved? b. Is their proactive communication with customers? c. Has customer satisfaction improved? |
| 5. Requirements management | <i>TelSoft</i> must improve the transparency and consistency of requirements change management as well as the approach to specify requirements. This lead to improved efficiency, transparency throughout the process, fewer errors, and increased customer satisfaction. | <ul style="list-style-type: none"> a. Has requirements change management been improved? b. Has requirements specification been improved? |
| 6. Software Quality assurance | <i>TelSoft</i> must build a consistent and systematic software quality assurance process and commit people on all levels to adopt it. This will lead to early detection of errors, improved efficiency, and increased customer satisfaction. | <ul style="list-style-type: none"> a. In what ways has the QA process been improved? b. How has the quality of the software product itself been improved? c. Measures of QA efficiency? d. Number of errors detected? e. Rework numbers? |
| 7. End-user interaction | <i>TelSoft</i> must establish closer interaction between software development and end-users. This will lead to improved understanding of requirements and to enhanced change management in collaboration with internal and external customers. | <ul style="list-style-type: none"> a. Amount of interaction with end-users? |

SPI Organization

Ask following questions about PST, SCG, and improvement teams:

1. What do you see as the underlying reason for having this team?
2. What is the main impact of this team?
3. How effective has this team been in managing its effort?
(For SCG: Specifically ask about each item on fixed agenda: current projects, business opportunities, improvement initiatives, account review, and strategy)
- 4.
5. What changes could improve this team's effectiveness?
6. What is your long-term vision for this team? (*PST and SCG only*)
7. What goals should this team focus on in 2007? (*PST and SCG only*)

Additional questions for SCG members:

1. What role do the policies play in business decisions and everyday actions?

2. What have you shared with your customers about policies and SPI? How do you think this has been received?

SPI Perception

1. How do different stakeholders perceive the SPI initiative (e.g., cynicism, enthusiasm, indifference)?
2. To what extent are those outside of the SPI initiative informed about the activity? Do they need more or less information? What's the preferred form for this information (e.g., workshop, newsletter, email, website update, etc.)?
3. Are the workshops an effective medium for communicating about the project?
4. What has surprised you most about this SPI effort?

Open-ended Closing: Anything else you feel that I should know that I have not covered?

B.11: SPI Impact Results Summary

April 18, 2007

Overview

This report summarizes employee perspectives on the software process improvement (SPI) initiative conducted between *TelSoft* and Georgia State University which began in October 2004. Two sources of data were gathered:

- Interviews with selected members of the Software Development group
- Online questionnaire distributed via questionpro.com given to all members of the Software Development group, marketing personnel, and select data services people involved

The purpose of this report is to gather perceptions from a diverse set of employees regarding the effectiveness of the SPI initiative and to gather suggestions for improving any future initiatives.

SPI Impact

Table 1: Overall Improvement by Work Group

| <i>Status</i> | <i>TOTAL</i> | <i>Managers</i> | <i>QA</i> | <i>Sales</i> | <i>Engineers</i> | <i>Other</i> |
|---------------------------------|--------------|-----------------|-----------|--------------|------------------|--------------|
| Made things worse | 0 | 0 | 0 | 0 | 0 | 0 |
| No change | 2 | 0 | 0 | 0 | 1 | 1 |
| Some improvement | 13 | 4 | 2 | 2 | 3 | 2 |
| Considerable improvement | 4 | 2 | 0 | 0 | 0 | 2 |
| Don't know | 7 | 0 | 2 | 0 | 5 | 0 |
| Total | 26 | 6 | 4 | 2 | 9 | 5 |

Software Development Assessment Summary

Table 2: Summary of Perceived Improvement

| <i>Area</i> | <i>Overall Assessment</i> |
|-----------------------------------|---------------------------|
| Software configuration management | Considerable improvement |
| Software quality assurance | Considerable improvement |
| Customer relations management | Some improvement |
| Requirements management | Little change |
| Software vision management | Little change |
| End-user interaction | No change |
| Project portfolio management | No change |

Improvement Areas: Considerable Improvement

Software Configuration Management

Description: *TelSoft* software configuration management should be improved to ensure consistent and transparent modification and packaging to individual customers. This ensures effective coordination with customers and minimizes adverse effects across projects.

Strengths

1. New software release process is consistently followed and allows early problem detection.
 - *TelSoft* now has documented process for building the following software products: <Name withheld>
 - Example provided during interview: VR used documentation to detect that an expected file was missing from a release.

Table 3: Questionnaire items related to release process

| <i>Area</i> | <i>Impact</i> | <i>TOTAL</i> | <i>Mgr</i> | <i>QA</i> | <i>Sales</i> | <i>Eng</i> | <i>Oth</i> |
|---|---------------------------------|--------------|------------|-----------|--------------|------------|------------|
| Impact of Software Release Specification | Made things worse | 0 | 0 | 0 | 0 | 0 | 0 |
| | No change | 2 | 0 | 0 | 0 | 1 | 1 |
| | Some improvement | 7 | 2 | 1 | 0 | 3 | 1 |
| | Considerable improvement | 6 | 5 | 1 | 0 | 0 | 1 |
| | Don't know | 11 | 0 | 2 | 2 | 5 | 2 |
| Impact on Practice: Policy on Release Documentation | Made things worse | 0 | 0 | 0 | 0 | 0 | 0 |
| | No change | 1 | 0 | 1 | 0 | 0 | 0 |
| | Some improvement | 7 | 2 | 0 | 0 | 3 | 2 |
| | Considerable improvement | 7 | 4 | 1 | 0 | 0 | 2 |
| | Don't know | 11 | 0 | 2 | 2 | 6 | 1 |
| Extent to which policy on Release Documentation is followed | Not used | 1 | 0 | 0 | 0 | 1 | 0 |
| | Discretionary | 1 | 0 | 1 | 0 | 0 | 0 |
| | Normally used | 7 | 3 | 0 | 0 | 2 | 2 |
| | Standardized | 7 | 3 | 2 | 0 | 0 | 2 |
| | Don't know | 10 | 0 | 1 | 2 | 6 | 1 |

2. Improved product packaging to customers reflects more professional image. The initiative raised awareness of importance of maintaining a professional image with all documents sent to customer

Table 4: Questionnaire items related to product packaging

| <i>Area</i> | <i>Impact</i> | <i>TOTAL</i> | <i>Mgr</i> | <i>QA</i> | <i>Sales</i> | <i>Eng</i> | <i>Oth</i> |
|---|---------------------------------|--------------|------------|-----------|--------------|------------|------------|
| Impact of improved client product packaging | Made things worse | 0 | 0 | 0 | 0 | 0 | 0 |
| | No change | 1 | 0 | 0 | 0 | 1 | 0 |
| | Some improvement | 2 | 0 | 1 | 0 | 1 | 0 |
| | Considerable improvement | 12 | 3 | 1 | 1 | 2 | 5 |
| | Don't know | 9 | 3 | 1 | 0 | 5 | 0 |

Software Development Assessment Summary

| <i>Area</i> | <i>Impact</i> | <i>TOTAL</i> | <i>Mgr</i> | <i>QA</i> | <i>Sales</i> | <i>Eng</i> | <i>Oth</i> |
|--|--------------------------|--------------|------------|-----------|--------------|------------|------------|
| Impact on Practice: Policy on professional standards | Made things worse | 0 | 0 | 0 | 0 | 0 | 0 |
| | No change | 2 | 2 | 0 | 0 | 0 | 0 |
| | Some improvement | 13 | 4 | 1 | 0 | 3 | 5 |
| | Considerable improvement | 0 | 0 | 0 | 0 | 0 | 0 |
| | Don't know | 11 | 0 | 3 | 2 | 6 | 0 |
| Extent to which policy on professional standards is followed | Not used | 0 | 0 | 0 | 0 | 0 | 0 |
| | Discretionary | 4 | 1 | 0 | 0 | 2 | 1 |
| | Normally used | 10 | 4 | 2 | 0 | 1 | 3 |
| | Standardized | 0 | 0 | 0 | 0 | 0 | 0 |
| | Don't know | 12 | 1 | 2 | 2 | 6 | 1 |

Opportunities

Respondent identified the following specific opportunity:

- Need better documentation for impact of PVCS merge. Something more specific than “there’s been a merge so test everything.” I would assume this comment is a result of merging <specific product> to trunk. That merge was an exception to what typical merges entail, normal impact statement practices will address most merge situations since branches usually have a relatively limited lifespan.

Software Quality Assurance

Description: *TelSoft* must build a consistent and systematic software quality assurance process and commit people on all levels to adopt it. This will lead to early detection of errors, improved efficiency, and increased customer satisfaction.

Strengths

The policy requiring quality assurance (QA) group to execute builds has been strictly followed and is very positively perceived. Selected comments from respondents include:

- “QA doing builds means they can trust the integrity of the builds”
- “I do see much improvement in quality assurance and that entire process - more standardized than what we had done previously and with QA doing builds it has forced us to document all our build and deployment processes + document release specifications.”

Table 5: Questionnaire items related to quality assurance

| <i>Area</i> | <i>Impact</i> | <i>TOTAL</i> | <i>Mgr</i> | <i>QA</i> | <i>Sales</i> | <i>Eng</i> | <i>Oth</i> |
|--|---------------------------------|--------------|------------|-----------|--------------|------------|------------|
| Impact of QA executes build (First Wave) | Made things worse | 0 | 0 | 0 | 0 | 0 | 0 |
| | No change | 2 | 0 | 0 | 0 | 2 | 0 |
| | Some improvement | 8 | 2 | 2 | 0 | 3 | 1 |
| | Considerable improvement | 7 | 4 | 1 | 0 | 0 | 2 |
| | Don't know | 9 | 0 | 1 | 2 | 4 | 2 |
| Impact of Refined QA process (Second Wave) | Made things worse | 0 | 0 | 0 | 0 | 0 | 0 |
| | No change | 2 | 1 | 1 | 0 | 0 | 0 |
| | Some improvement | 9 | 1 | 2 | 0 | 4 | 2 |
| | Considerable improvement | 3 | 2 | 0 | 0 | 0 | 1 |
| | Don't know | 11 | 2 | 0 | 2 | 5 | 2 |

Software Development Assessment Summary

| <i>Area</i> | <i>Impact</i> | <i>TOTAL</i> | <i>Mgr</i> | <i>QA</i> | <i>Sales</i> | <i>Eng</i> | <i>Oth</i> |
|---|---------------------------------|--------------|------------|-----------|--------------|------------|------------|
| Impact on Practice: Policy on QA Approval | Made things worse | 0 | 0 | 0 | 0 | 0 | 0 |
| | No change | 0 | 0 | 0 | 0 | 0 | 0 |
| | Some improvement | 10 | 3 | 1 | 0 | 3 | 3 |
| | Considerable improvement | 7 | 3 | 2 | 0 | 0 | 2 |
| | Don't know | 9 | 0 | 1 | 2 | 6 | 0 |
| Extent to which policy on QA Approval is followed | Not used | 0 | 0 | 0 | 0 | 0 | 0 |
| | Discretionary | 1 | 1 | 0 | 0 | 0 | 0 |
| | Normally used | 8 | 2 | 1 | 0 | 2 | 3 |
| | Standardized | 8 | 3 | 2 | 0 | 1 | 2 |
| | Don't know | 9 | 0 | 1 | 2 | 6 | 0 |

Opportunities

Many respondents pointed to integration testing as an area needing improvement. The main issues appear to be

- Lack of policies or guidelines provided for integration testing; therefore, quality varies greatly according to who does it.
- Belief that someone other than developer should conduct integration testing.

Other indicators of issues with integration testing:

- Quality of the software coming from integration → QA is not as good as it used to be. Used to take 3 cycles to get a release out the door. Last release, it took 5-6 cycles.

Selected comments

- “Integration testing - I know not on the list, but perhaps it should be. Having developers test their own stuff in integration is no better than unit testing.”
- “Development is doing more integration testing. Developers would rather stick with doing development. I would rather have another group do integration testing and have developer stick with design, consult and development.” Seems like we need to formalize some guidelines here.

Improvement Areas: Some Improvement

Customer Relations Management

Description: *TelSoft* should improve its management of customer relations to ensure more symmetric information sharing and proactive expectation and change management. This leads to increased customer satisfaction.

Strengths

Project managers are spending more face-to-face time with BST and EMBARQ. As a consequence, the relationship with BellSouth has improved. The relationship with EMBARQ has remained strong. In addition, the software charter Software Charter (reason for being, strategy, and policies) have been communicated to customers via letter and, in some case, in person.

Selected comments from questionnaire:

- “Much less squawking from employees and customers.”

Software Development Assessment Summary

- “Customer relations efforts - more focus on face/face and client communication channels; also presentation of our software has also improved - looks more professional now.”

Table 6: Questionnaire items related to customer relations

| <i>Area</i> | <i>Impact</i> | <i>TOTAL</i> | <i>Mgr</i> | <i>QA</i> | <i>Sales</i> | <i>Eng</i> | <i>Oth</i> |
|---|---------------------------------|--------------|------------|-----------|--------------|------------|------------|
| Impact of Weekly Status Report template (First Wave) | Made things worse | 1 | 0 | 0 | 0 | 0 | 0 |
| | No change | 6 | 2 | 0 | 0 | 3 | 1 |
| | Some improvement | 5 | 2 | 1 | 0 | 0 | 1 |
| | Considerable improvement | 1 | 0 | 0 | 0 | 0 | 0 |
| | Don't know | 12 | 2 | 2 | 2 | 6 | 3 |
| Impact of TelSoft website update (Second Wave) | Made things worse | 0 | 0 | 0 | 0 | 0 | 0 |
| | No change | 4 | 2 | 0 | 0 | 1 | 1 |
| | Some improvement | 9 | 1 | 2 | 2 | 1 | 3 |
| | Considerable improvement | 3 | 1 | 0 | 0 | 1 | 1 |
| | Don't know | 9 | 2 | 1 | 0 | 6 | 0 |
| Impact on Customer Contact Database (ACT) | Made things worse | 0 | 0 | 0 | 0 | 0 | 0 |
| | No change | 5 | 2 | 0 | 1 | 0 | 2 |
| | Some improvement | 3 | 0 | 0 | 0 | 1 | 2 |
| | Considerable improvement | 0 | 0 | 0 | 0 | 0 | 0 |
| | Don't know | 16 | 4 | 3 | 0 | 8 | 1 |
| Extent to which policy on Communicate Status is followed | Not used | 0 | 0 | 0 | 0 | 1 | 0 |
| | Discretionary | 5 | 3 | 0 | 0 | 1 | 1 |
| | Normally used | 5 | 2 | 1 | 0 | 2 | 3 |
| | Standardized | 3 | 0 | 0 | 0 | 0 | 0 |
| | Don't know | 13 | 1 | 3 | 2 | 5 | 1 |
| Extent to which policy on Relationship Management is followed | Not used | 0 | 0 | 0 | 0 | 0 | 0 |
| | Discretionary | 5 | 0 | 1 | 1 | 1 | 2 |
| | Normally used | 7 | 3 | 1 | 0 | 1 | 2 |
| | Standardized | 0 | 0 | 0 | 0 | 0 | 0 |
| | Don't know | 14 | 3 | 2 | 1 | 7 | 1 |
| Extent to which policy on Customer Knowledge is followed | Not used | 1 | 0 | 0 | 0 | 0 | 0 |
| | Discretionary | 4 | 2 | 0 | 0 | 0 | 2 |
| | Normally used | 4 | 2 | 2 | 2 | 0 | 2 |
| | Standardized | 1 | 0 | 0 | 0 | 1 | 0 |
| | Don't know | 16 | 2 | 2 | 0 | 8 | 1 |

Opportunities

Comments from questionnaire on barriers to success:

- “Still think we don't understand our customer's business”
- “Business knowledge, impact on business of relationship (customer) management”
- “Small customer and personnel base, few new projects to implement and refine new processes.”

Improvement Areas: Little Change

Software Vision Management

Description: *TelSoft* strategy for software development and customer service should be explicated, maintained, and communicated. This provides a value-based foundation for requirements coordination and management that is consistent with *TelSoft*' business strategy.

Strengths

The creation of the software charterSoftware Charter (reason for being, software strategy, and policies) was one of the primary ways of enhancing software vision management. Some successes in this area:

- *TelSoft* has educated Local TelCo regarding the two-phased funding policy and received agreement to operate this way.
- *TelSoft* has mapped out release schedule for products in a more collaborative way.

Note: A more detailed assessment of the software coordination group activities has been compiled separately.

Opportunities

Increase visibility of policies for both new and existing employees:

- “More knowledge of United Way campaign than company’s vision and policies.”
- Need to ensure that new hires will see the policies and be informed about processes

Reconsider *TelSoft*' real strategy, particularly with respect to emerging markets and new customers:

- “We came up with the reason for being, but it’s not necessarily a driving force. The actual product strategy is not solidified and communicated.”
- “Too few resources to adequately respond to new technologies or customers”
- “*TelSoft* has suffered due to poor overall business environment & national economy - very intense foreign competition - high level of mergers & acquisitions among customer base delayed or even halted many purchases of *TelSoft* products and services.”

Requirements Management

Description: *TelSoft* must improve the transparency and consistency of requirements change management as well as the approach to specify requirements. This lead to improved efficiency, transparency throughout the process, fewer errors, and increased customer satisfaction.

Strengths

For internal projects, *TelSoft* is doing a better job of documenting requirements than they would have done it before.

Opportunities

Functional specification:

- Functional specification (FS) is now too streamlined for development and QA. Recent FS have had “lots of holes” and had to be rewritten by development.

Software Development Assessment Summary

- Many inadequacies with FS are caught during design time. Since technical specifications (TS) are not frequently done, we catch these later and later in the process.
- Functional specification should always be reviewed by development before being sent to client. All three above continue to be issues. FS really need to be more fully fleshed out than they have been. As of late we are seeing some “requirements” in the FS being implied through screenshots and examples instead of being spelled out. This leaves the developer having to analyze the data in the screen shots to figure out what they need to implement. In a current project we are almost a month into the project, and did not have a finalized data base schema. The common pattern appears to be that more of the FS that aren’t fully fleshed out are internal projects. Something that PM has done in the past is to get development involved in discovery sessions prior to completion of the FS, I believe this worked well in determining what is and is not possible. This is something that I would like to see more of.
- “We don't have any true business analyst's left in the group

Change controls are still not consistently communicated for internal projects.

Suggested Improvement Areas

1. Scheduling

- a. Development needs input on estimates rather than being provided a date. Potential impact to code and likely problems that will be encountered may also be known by the developers. This knowledge might lead to additional items being added to the work program.
- b. Suggestion: Since PM schedules resources upfront, she could apply a rule that developers do not test their own work. While I generally agree that a developer really shouldn’t integration test their own work, I would not go as far as to say that is should never happen. Every effort should be made to avoid the situation, but sometime it may be necessary schedule wise to do this.
- c. Include time for process improvement in the schedule to adjust workload. When a person is assigned to an improvement team, add time for participating on that team into schedule; otherwise, the person may be overloaded with day-to-day work activities and not have the time to focus on improvement.

2. Project management

- a. Setup a standard protocol for managing *TelSoft* projects. Currently there is no consistency or quality control on how projects are managed
- b. Increased managerial intervention
- c. Consistency of project management between managers
- d. Project Management Process and Tools
- e. Estimation process and accuracy.

3. Communication

- a. “I am aware that some of the initiatives are in place but since they don't directly affect me that is all I can say about them.”
- b. Developers are having to communicate status and answer to too many managers

Software Development Assessment Summary

- c. Communication between managers and 'workers'.
Communication between upper management and 'workers'
 - d. In order to be effective, the goals of each aspect of the program need to be communicated to the rank and file and then implemented from the top down.
4. Resources
 - a. Lack of resources - no business analysts on staff for example
 - b. It seemed that a lack of resources may have been a factor. Low morale because of a lack of work was also a factor.
 - c. Small workforce.
 5. Implementation and refinement of designed initiatives
 - a. PDPR database cleanup and standardization of statuses
 - b. Get release documentation a little more consistent (currently it varies by PM)
 - c. Approve documents pending approval/revision.

SPI Organization

Strengths

1. Full support of management, including willingness to enforce process changes
2. Joint effort. Participatory – involved the right people who would also be responsible for making the changes. Committed team members who genuinely wanted to improve the processes.
 - a. “I think it was good to use a fresh approach and get more people involved. The various teams did a good job.”
 - b. “Increased the level communication, awareness, and understanding among the groups involved in the initiative/project - Provided opportunities for discussions focused on fundamental business issues among groups that don't normally/frequently work together”
3. Improved processes
 - a. “Processes are better understood and more consistently followed.”
 - b. “Has had a positive impact on establishing firm processes for product packaging and QA/QC authority over product releases.”
 - c. “I have seen some serious improvement in how we handle releases. QA is doing a nice job.”
4. Legitimized the topic of process improvement
 - a. PST: “If you didn't have the group, you wouldn't have anyone that looked at improvement. The improvement focus could get lost in the hectic pace of the day”
 - b. I think people are at least more in tune to the fact that process is important. I think having QA do the builds has been a positive improvement for one specific example.”
 - c. “We did "QA Does Builds" effort and a number of other improvements to our process and people think critically about our processes more now as a result of attention to these issues.”

Opportunities

1. Improvement team organization

Software Development Assessment Summary

- a. Difficulty in people having enough time to do work in between meetings.
 - b. Might have been more productive to have the time compressed (e.g. 1 ay/week for 3 weeks instead of 24 hours over 6 weeks): “We could all just sit in a room for a few days to get it done. Constant bantering back and forth every few weeks wasn’t productive.” “I would have preferred more time in a shorter period instead of dragging it out over months”
 - c. Teams need more direction and feedback from PST throughout the process. Suggestion: have the person who came up with the specific issue be present when the improvement team first meets in order to clarify things.
 - d. Smaller teams, less time - I'm concerned about the number of hours spent on this whole initiative vs. what was actually gained; 2-3 hour kickoffs and other meetings w/15 people seems excessive////strip that down and cut out much of the presentations - we simply can't spare that much time away from project activities!
2. Increase participation and involvement.
 - a. Broaden participation in the initiative (e.g. only one member of Rick’s group participated on a team)
 - b. Not only start from Top levels, also need work from bottom-up.
 - c. “Some people just had the experience of having final results presented to them; they were not really participants even though they may have wanted to be.”
 - d. “Only people it’ll be meaningful to are the ones that were on the team.”
 - e. Follow model of first workshop where there was more of an open dialogue instead of just one-way communication.
 - f. “I have not been involved enough in these initiatives to know how they are or should be impacting the company. However, that does not speak well for this program being implemented below the managerial level.” (questionnaire response)
3. PST
 - a. Consider rotating non-management level people onto the PST Good idea this would also help with 2 above, with the key being selecting the non-management types that would not resent being on the panel.
 - b. As PST becomes focused on document revisions, need to still keep engaging “larger part of the audience”
4. Close communication gaps
 - a. “I think things will happen, but folks won’t know”
 - b. Newsletters or emails about what’s happening would be excellent – could even replace the need for status workshops
 - c. Consider sharing news about business opportunities with people outside of SCG and management
 - d. Implementation - I am aware of items that directly affect me with regards to implementation of initiatives, but I answered don't know to most of the questions on implementing the initiatives because we either haven't done them yet, or I am simply unaware that we have done things.
 - e. Consider doing interviews or surveys annually. Might even do it more often (no more than bi-annually.)
 - f. Perception of amount of information provided about the improvement initiative: 19 out of 26 said enough. 7 out of 26 said Not Enough

Software Development Assessment Summary

- g. What is your perception regarding your own level of participation the improvement initiative? 19 out of 26 said enough. 7 out of 26 said Not Enough
5. Finish what we started
- a. PDPR bug cleanup
 - b. Post-project reviews represent a big opportunity for learning
 - c. "Implementation is slow, and following procedures is somewhat sporadic at times as we phase into some of the initiatives."
 - d. Slow things down somewhat - we probably really need to fully implement the initiatives prior to moving on to another round. Or I guess you could also say speed things up on the implementation. To be fair though we really need to have some projects completed or nearing completion to implement some items.
 - e. "Seems like business as usual. Although we now have some thing concrete to point to in support of the way we do things."
 - f. "I'm not convinced all initiatives have been fully implemented; for instance, I haven't seen any cleanup of the PDPR database. I never saw the email policies published. Etc"
 - g. "A lot of work went in to the web site, but I'm not sure it bought us any thing."
 - h. "Some [initiatives] appear dead or have no clear direction and/or funding"

B.12: Requirements Engineering Assessment Results

7/17/2007

This document shows the results of the latest Requirements Assessment conducted on June 19, 2007. The values are compared against a similar assessment that was conducted on March 2005.

Major findings:

- *TelSoft's* overall Requirements Maturity increased from *Initial* to *Repeatable* (comparing Tables 1 & 2).
- *TelSoft* increased the % of best practices used in 6 of the 8 areas (comparing Tables 3 & 4).
- *TelSoft* improved all of its *Weak* areas to *Average* (Table 5).

Table 1: Strength Matrix (Pre=3/30/2005; Post=6/19/2007)

| Area | Weak | Average | Good | Strong |
|---|-------------|---------|------|-------------|
| Requirements Document | | | | Pre Post |
| Requirements Elicitation | | | Pre | Post |
| Requirements Analysis and Negotiation | Pre | Post | | |
| Describing Requirements | | | | Pre Post |
| System Modeling | | Pre | Post | |
| Requirements Validation | Pre | Post | | |
| Requirements Management | Pre | Post | | |
| Requirements Engineering for Critical Systems | Pre Post | | | |

The four area strength parameters are used as follows:

| | |
|---------|------------------------|
| Weak | 0 <= % Usage <= 30 |
| Average | 30 < % Usage <= 50 |
| Good | 50 < % Usage <= 70 |
| Strong | 70 < percentage <= 100 |

Scores

- Standardized (**ST, 3**): The process or practice has a documented standard which is followed and checked as part of your quality management process.
- Normal (**N, 2**): Guideline is widely followed in your organization but is not mandatory
- Discretionary (**D, 1**): Some project managers may have introduced the guideline but it is not universally used
- Rare (**R, 0**): Never or very rarely applied

Table 2: Scores for basic guidelines

| ID | Guideline | Score (3/30/05) | Score (6/19/05) |
|-----------|---|----------------------------|----------------------------|
| 03.01 | Define a standard document structure | 3 | 3 |
| 03.02 | Explain how to use the document | 0 | 3 |
| 03.03 | Include a summary of the requirements | 3 | 3 |
| 03.04 | Make a business case for the system | 3 | 2 |
| 03.05 | Define specialized terms | 3 | 3 |
| 03.06 | Lay out the document for readability | 3 | 3 |
| 03.07 | Help readers find information | 3 | 3 |
| 03.08 | Make the document easy to change | 1 | 3 |
| 04.01 | Assess system feasibility | 1 | 1 |
| 04.02 | Be sensitive to organizational and political considerations | 0 | 1 |
| 04.03 | Identify and consult system stakeholders | 1 | 2 |
| 04.04 | Record requirements sources | 1 | 2 |
| 04.05 | Define the system's operating environment | 0 | 3 |
| 04.06 | Use business concerns to drive requirements elicitation | 0 | 3 |
| 05.01 | Define system boundaries | 0 | 1 |
| 05.02 | Use checklists for requirements analysis | 0 | 0 |
| 05.03 | Provide software to support negotiations | 2 | 2 |
| 05.04 | Plan for conflicts and conflict resolution | 1 | 2 |
| 05.05 | Prioritise requirements | 0 | 0 |
| 06.01 | Define standard templates for describing requirements | 0 | 3 |
| 06.02 | Use language simply, consistently and concisely | 1 | 1 |
| 06.03 | Use diagrams appropriately | 2 | 1 |
| 06.04 | Supplement natural language with other descriptions of requirements | 1 | 2 |
| 07.01 | Develop complementary system models | 1 | 0 |
| 07.02 | Model the system's environment | 1 | 1 |
| 07.03 | Model the system architecture | 0 | 2 |
| 08.01 | Check that the requirements document meets your standards | 0 | 0 |
| 08.02 | Organize formal requirements inspections | 0 | 3 |
| 08.03 | Use multi-disciplinary teams to review requirements | 3 | 3 |
| 08.04 | Define validation checklists | 0 | 0 |
| 09.01 | Uniquely identify each requirement | 3 | 3 |
| 09.02 | Define policies for requirements management | 0 | 3 |

| ID | Guideline | Score (3/30/05) | Score (6/19/05) |
|-------|--------------------------------|--------------------|--------------------|
| 09.04 | Maintain a traceability manual | 0 | 0 |
| | Score | 37 | 62 |

Table 3: Scores for intermediate guidelines

| ID | Guideline | Score (3/30/05) | Score (6/19/05) |
|-------|---|--------------------|--------------------|
| 04.07 | Look for domain constraints | 3 | 3 |
| 04.08 | Record requirements rationale | 2 | 0 |
| 04.09 | Collect requirements from multiple viewpoints | 2 | 1 |
| 04.10 | Prototype poorly understood requirements | 1 | 0 |
| 04.11 | Use scenarios to elicit requirements | 1 | 3 |
| 04.12 | Define operational processes | 1 | 2 |
| 05.06 | Classify requirements using a multi-dimensional approach | 0 | 0 |
| 05.07 | Use interaction matrices to find conflicts and overlaps | 0 | 0 |
| 06.05 | Specify requirements quantitatively | 2 | 2 |
| 07.04 | Use structured methods for system modeling | 0 | 0 |
| 07.05 | Use a data dictionary | 0 | 3 |
| 07.06 | Document the links between stakeholder requirements and system models | 0 | 0 |
| 08.05 | Use prototyping to animate requirements | 0 | 0 |
| 08.06 | Write a draft user manual | 1 | 0 |
| 08.07 | Propose requirements test cases | 0 | 1 |
| 09.03 | Define traceability policies | 0 | 0 |
| 09.05 | Use a database to manage requirements | 0 | 1 |
| 09.06 | Define change management policies | 0 | 3 |
| 09.07 | Identify global system requirements | 1 | 0 |
| | Score | 14 | 19 |

Table 4: Scores for advanced guidelines

| ID | Guideline | Score (3/30/05) | Score (6/19/05) |
|-------|--------------------------------|--------------------|--------------------|
| 04.13 | Reuse requirements | 0 | 1 |
| 05.08 | Assess requirements risks | 0 | 0 |
| 08.08 | Paraphrase system models | 0 | 0 |
| 09.08 | Identify volatile requirements | 0 | 0 |
| 09.09 | Record rejected requirements | 0 | 0 |
| | Score | 0 | 1 |

Table 5: Assessment Summary (3/30/2005)

| | Basic | Intermediate | Advanced |
|--------------------|----------------|---------------------|-----------------|
| Guidelines Used | 19 | 9 | 0 |
| Weighted Score | 37 | 14 | 0 |
| Maximum Possible | 105 | 66 | 27 |
| Score % of Maximum | 35% | 21% | 0% |
| Level | Initial | | |

Table 6: Assessment Summary (6/19/2005)

| | Basic | Intermediate | Advanced |
|--------------------|-------------------|---------------------|-----------------|
| Guidelines Used | 27 | 9 | 1 |
| Weighted Score | 62 | 19 | 1 |
| Maximum Possible | 105 | 66 | 27 |
| Score % of Maximum | 59% | 29% | 4% |
| Level | Repeatable | | |

Assignment of maturity level used the following scale (Sommerville and Sawyer 1997):

- Initial: Less than 55 in the basic guidelines. May have implemented some intermediate guidelines
- Repeatable: Above 55 in the basic guidelines but less than 40 in the intermediate and advanced guidelines
- Defined: More than 65 in the basic guidelines and more than 40 in the intermediate and advanced guidelines

Table 7: Guideline Usage Summary (3/30/2005)

| | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 |
|------------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| Guideline counts | 7 | 9 | 2 | 4 | 2 | 2 | 2 | 0 |
| Maximum | 8 | 13 | 8 | 5 | 6 | 8 | 9 | 9 |
| % Usage | 88 | 69 | 25 | 80 | 33 | 25 | 22 | 0 |

Table 8: Guideline Usage Summary (6/19/2007)

| | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 |
|------------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| Guideline counts | 8 | 11 | 3 | 4 | 3 | 3 | 4 | 0 |
| Maximum | 8 | 13 | 8 | 5 | 6 | 8 | 9 | 9 |
| % Usage | 100 | 85 | 38 | 80 | 50 | 38 | 45 | 0 |

References

- Abrahamsson, P., O. Salo, et al. (2002). Agile Software Development Methods – Review and Analysis. Oulu, VTT Electronics.
- Baskerville, R. and T. Wood-Harper (1996). "A Critical Perspective on Action Research as a Method for Information Systems Research." Journal of Information Technology **11**: 235-246.
- Borjesson, A. and L. Mathiassen (2005). "Improving software organizations: agility challenges and implications." Information Technology & People **18**(4): 359-382.
- CMMI Product Team (2002). CMMI for Systems Engineering/Software Engineering/Integrated Product and Process Development/Supplier Sourcing, Software Engineering Institute.
- Davison, R. M., M. G. Martinsons, et al. (2004). "Principles of canonical action research." Information Systems Journal **14**(1): 65-86.
- Dove, R. (2001). Response Ability: the Language, Structure, and Culture of the Agile Enterprise. New York, Wiley.
- Gunneson, A. O. (1997). Transitioning to Agility – Creating the 21st Century Enterprise. Reading, MA, Addison-Wesley.
- Haeckel, S. (1995). "Adaptive enterprise design: the sense-and-respond model." Planning Review **23**(3): 6-13, 42.
- Haeckel, S. (1999). Adaptive Enterprise: Creating and Leading Sense-and-Respond Organizations. Boston, MA, Harvard Business School Press.
- Kock, N. (1997). "Negotiating mutually satisfying IS action research topics with organizations: an analysis of Rapoport's initiative dilemma." Journal of Workplace Learning **9**(7): 253-62.
- Mathiassen, L. (2002). "Collaborative practice research." Information Technology & People **15**(4): 321-345.
- McFeeley, B. (1996). IDEAL: A user's guide for software process improvement. Pittsburgh, PA, Software Engineering Institute.
- McKay, J. and P. Marshall (2001). "The dual imperatives of action research." Information Technology & People **14**(1): 46-59.
- Napier, N. P., J. Kim, et al. (under review). "Software Process Reengineering: A Model and Its application to an industrial case study." IEEE Transactions on Software Engineering.
- Napier, N. P., L. Mathiassen, et al. (2006). Perceptions and Processes in assessing software requirements practices. Proceedings of the Twelfth Americas Conference on Information Systems, Acapulco, Mexico.
- Overby, E., A. Bharadwaj, et al. (2006). "Enterprise agility and the enabling role of information technology." European Journal of Information Systems **15**(2): 120-131.
- Paulk, M., B. Curtis, et al. (1993). Capability Maturity Model for Software, Version 1.1. Pittsburgh, PA, Software Engineering Institute.
- Paulk, M., C. V. Weber, et al., Eds. (1995). The Capability maturity model: guidelines for improving the software process. SEI Series in Software Engineering. Boston, Addison-Wesley.
- Rapoport, R. (1970). "Three Dilemmas in Action Research." Human Relations **23**(6): 499-513.

- Sommerville, I. and J. Ransom (2005). "An empirical study of industrial requirements engineering process assessment and improvement." ACM Transactions on Software Engineering and Methodology **14**(1): 85-117.
- Sommerville, I. and P. Sawyer (1997). Requirements Engineering: A Good Practice Guide. New York, NY, John Wiley & Sons.
- Zaheer, A. and S. Zaheer (1997). "Catching the Wave: Alertness, Responsiveness and Market Influence in Global Electronic Networks." Management Science **43**(11): 1493-1509.