Association for Information Systems
AIS Electronic Library (AISeL)

ICIS 2006 Proceedings

International Conference on Information Systems (ICIS)

December 2006

Negotiating Response-ability and Repeat-ability in Requirements Engineering

Nannette Napier
Georgia State University

Lars Mathiassen
Georgia State University

Roy Johnson
University of Pretoria

NEGOTIATING RESPONSE-ABILITY AND REPEAT-ABILITY IN REQUIREMENTS ENGINEERING

Alternative Approaches to Information Systems Development

Nannette P. Napier
Computer Information Systems
Georgia State University
nnapier1@gsu.edu

Lars Mathiassen
Center for Process Innovation
Georgia State University
lmathiassen@gsu.edu

Roy D. Johnson
University of Pretoria
Department of Informatics
Roy@UP.ac.az

Abstract

Requirements engineering (RE) practices are critical to success during the 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 et al. ,2001). Despite RE-specific process descriptions and best practices (Beecham et al.,, 2005b; CMMI Product Team, 2002; Sommerville and Sawyer, 1997), RE remains one of the most challenging aspects of business software development (Beecham et al., 2005a). This is due in part to competitive

business environments characterized by frequent requirements changes, rapid technological advances, and time-to-market pressures (Ramesh et al., 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 that 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 et al.,, 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 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). 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

Assumption	*Repeat-ability*	*Response-ability*
1. Nature of requirements	▪ Requirements represent software capabilities ▪ Requirements are explicated as texts in documents	▪ Requirements are perceptions of software capabilities ▪ Requirements are tacitly embedded in social relationships
2. Requirements capture	▪ Requirements are derived through specification ▪ Interaction is formal	▪ Requirements are discovered through negotiation ▪ Interaction is informal
3. Requirements usage	▪ Requirements are baselined and predate development ▪ Requirements are stored with traceability to source code	▪ Requirements emerge through development ▪ Requirements are expressed through software solutions
4. Change management	▪ Requirements changes are exceptions and must be managed	▪ Requirements changes are expected and must be embraced
5. Improvement approach	▪ The goal is to reduce process variance through best practices	▪ 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).

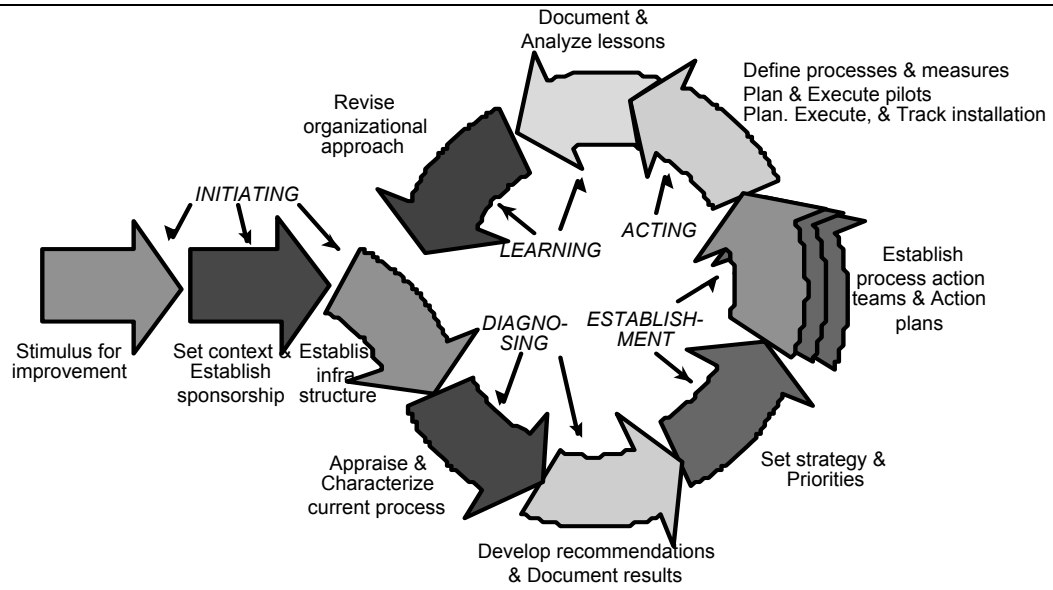

Figure 1: IDEAL Model (McFeeley 1996)

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.

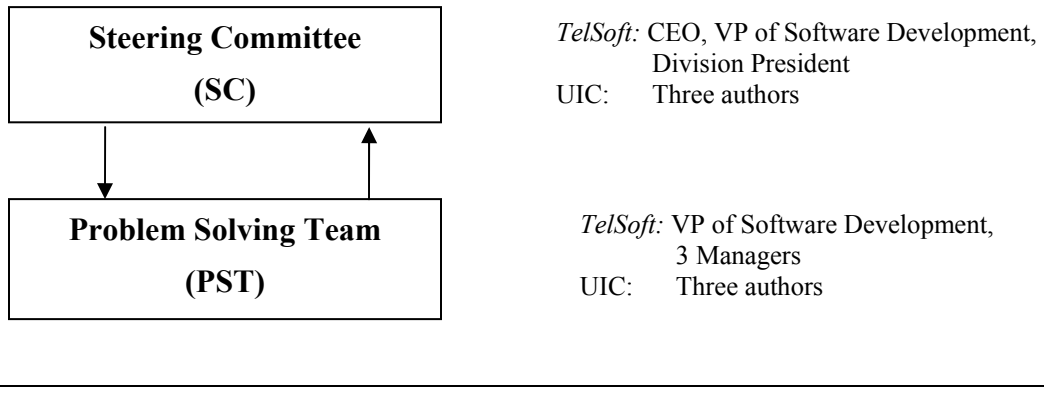

Figure 2: Managing Collaborative Practice Research (Mathiassen 2002)

Data Collection

Data collection and documentation are essential for successful action research and qualitative research in general (Avison et al., 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

Group Affiliation	***Count***	***Role***
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

Requirements Documents	Requirements Activities
• Which? • Inputs from whom? • Contributions? • Output to whom?	• Which? • Interactions? • Collaboration? • Resources?
Strengths, Weaknesses, and Opportunities	Strengths, Weaknesses, and 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*, *w*e 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 useful 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. 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)

	Repeat-ability	***Response-ability***
Problems	• 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)	• 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)
Recommendations	• 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	• 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 1. 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

Agile Alliance "Manifesto for Agile Software Development", http://www.agilemanifesto.org/, 2001.

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.

Avison, D., Lau, F., Myers, M., and Nielsen, P.A. "Action Research", *Communications of the ACM*, (42: 1), January 1999, pp. 94-97.

Baker, S. "Formalizing Agility: An Agile Organization's Journey toward CMMI Accreditation", in *Agile Conference*, July 24-29, 2005, pp. 185-192.

Baskerville, R. "Diversity in Information Systems Action Research Methods", *European Journal of Information Systems*, (7: 2), June 1998, pp. 90-107.

Beck, K. *Extreme Programming Explained: Embrace Change*, Addison-Wesley, Reading, MA, 1999.

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.

Beecham, S., Hall, T., and Rainer, A. "Defining a Requirements Process Improvement Model", *Software Quality Journal*, (13: 3), September 2005b, pp. 247-279.

Boehm, B.W. "The Economics of Software Maintenance", in *Proceedings of the Software Maintenance Workshop*, Washington, DC, 1983, pp. 9-37.

Boehm, B.W. "Get Ready for Agile Methods, with Care", *Computer*, (35: 1), January 2002, pp. 64-69.

Boehm, B.W., and Turner, R. *Balancing Agility and Discipline: A Guide for the Perplexed*, Addison-Wesley, Boston, 2004.

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.

Cockburn, A. *Writing Effective Use Cases*, Addison-Wesley, Reading, MA, 2000.

Dove, R. *Response Ability: The Language, Structure, and Culture of the Agile Enterprise*, Wiley, New York, 2001.

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.

Gunneson, A.O. *Transitioning to Agility -- Creating the 21st Century Enterprise*, Addison-Wesley, Reading, MA, 1997.

Haeckel, S. "Adaptive Enterprise Design: The Sense-and-Respond Model", *Planning Review*, (23: 3)1995, pp. 6-13, 42.

Haeckel, S. *Adaptive Enterprise: Creating and Leading Sense-and-Respond Organizations*, Harvard Business School Press, Boston, MA, 1999.

Highsmith, J. *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*, Dorset House Publishing, New York, NY, 2000.

Highsmith, J., and Cockburn, A. "Agile Software Development: The Business of Innovation", *Computer*, (34: 9), September 2001, pp. 120-127.

Humphrey, W.S. *Managing the Software Process*, Addison-Wesley, Boston, MA, 1989.

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.

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.

Kotonya, G., and Sommerville, I. *Requirements Engineering Processes and Techniques*, John Wiley & Sons, 1998.

Kuvaja, P., and Bicego, A. "Bootstrap - a European Assessment Methodology", *Software Quality Journal*, (3: 3), September 1994, pp. 117-127.

Langley, A. "Strategies for Theorizing from Process Data", *Academy of Management Review*, (24: 4), October 1999, pp. 691-710.

Markus, M.L. "Power, Politics, and MIS Implementation", *Communications of the ACM*, (26: 6), June 1983, pp. 430-444.

Mason, J. *Qualitative Researching*, (2nd edition ed.), Sage, London, 2002.

Mathiassen, L. "Collaborative Practice Research", *Information Technology & People*, (15: 4)2002, pp. 321-345.

McFeeley, B. "Ideal: A User's Guide for Software Process Improvement", CMU/SEI-96-HB-001, Software Engineering Institute, Pittsburgh, PA, 1996.

McKay, J., and Marshall, P. "The Dual Imperatives of Action Research", *Information Technology & People*, (14: 1)2001, pp. 46-59.

Miles, M.B., and Huberman, A.M. *Qualitative Data Analysis: An Expanded Sourcebook*, Sage Publications, Thousand Oaks, 1994.

Orr, K. "Agile Requirements: Opportunity or Oxymoron?" *IEEE Software*, (21: 3), May-June 2004, pp. 71-73.

Paulk, M. "Extreme Programming from a CMM Perspective", *IEEE Software*, (18: 6), November-December 2001, pp. 19-26.

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.

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.

Rapoport, R. "Three Dilemmas in Action Research", *Human Relations*, (23: 6)1970, pp. 499-513.

Rout, T.P. "Spice: A Framework for Software Process Assessment", *Software Process: Improvement and Practice*, (1: 1), August 1995, pp. 57-66.

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.

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.

Sommerville, I., and Sawyer, P. *Requirements Engineering: A Good Practice Guide*, John Wiley & Sons, New York, NY, 1997.

Susman, G., and Evered, R. "An Assessment of the Scientific Merits of Action Research", *Administrative Science Quarterly*, (23: 4)1978, pp. 582-603.

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.